CICS Transaction Server for z/OS



Diagnosis Reference

Version 5 Release 4

CICS Transaction Server for z/OS



Diagnosis Reference

Version 5 Release 4

Note
Before using this information and the product it supports, read the information in "Notices" on page 2085.

This edition applies to the IBM CICS Transaction Server for z/OS Version 5 Release 4 (product number 5655-Y04) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1997, 2017.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this PDF xiii

Chapter 1. CICS domains 1

CICS domains 1
Domain gates 5
Functions provided by gates 5
Specific gates, generic and call-back gates. . . . 5
Domain call formats 6
Ownership of formats 6
Tokens 6
Asynchronous services domain (AS) 7
Asynchronous services domain (AS) 8
Application manager domain (AP) 10
Application domain 10
Application manager domain's specific gates . . 11
Application manager domain's call-back gates 333
Application manager domain's generic gates 333
Application manager domain's generic formats 334
Business application manager domain (BA) . . . 335
Business application manager domain's specific gates 335
Business application manager domain's generic gates 363
Business application manager domain's call-back gates 364
Business application manager domain's generic formats 364
Modules 365
Exits 367
CICS catalog domain (CC) 367
CICS catalog domain's specific gates. 367
CICS catalog domain's generic gates. 372
Modules 373
Directory manager domain (DD) 373
Directory manager domain's specific gates. . . 374
Directory manager domain's generic gates. . . 384
Document handler domain (DH) 384
Document handler domain's specific gates. . . 384
Document handler domain's generic gates. . . 407
Document handler domain's call-back gates . . 407
Modules 408
Domain manager domain (DM) 409
Domain manager domain's specific gates . . . 409
Domain manager domain's generic gates . . . 416
Domain manager domain's generic formats . . 416
Domain manager domain's callback formats . . 418
Modules 419
Debugging profile domain (DP) 419
Debugging profile domain's specific gates . . . 419
Debugging profile domain's generic gates . . . 453
Dispatcher domain (DS) 454
Dispatcher domain's specific gates 454
Dispatcher domain's generic gates 489
Dispatcher domain's generic formats 490
Modules 492
Exits 492
Dump Domain (DU) 493
Dump Domain's specific gates. 493
Dump domain's generic gates 515
Modules 518
Exits 519
Enterprise Java domain (EJ) 519
Enterprise Java domain's specific gates 519
Enterprise Java domain's generic gates 520
Modules 520
Event manager domain (EM) 521
Event manager domain's specific gates 521
Event manager domain's generic gates 532
Modules 532
Event processing domain (EP). 533
Event processing domain's specific gates . . . 533
Event processing domain's generic gates . . . 543
Modules 543
IP ECI (IE) domain 543
IP ECI domain's specific gates. 543
IP ECI domain's generic gates 546
Modules 546
Inter-system (IS) domain 546
IS domain specific gates 546
IS domain modules 581
Kernel domain (KE) 582
Kernel domain's specific gates. 582
Kernel domain's generic formats 609
Modules 611
Loader domain (LD) 612
Loader domain's specific gates 612
Loader domain's generic gates. 643
Modules 644
Log manager domain (LG) 645
Log manager domain's specific gates 646
Logger manager domain's generic gates . . . 679
Log manager domain's call-back gates 679
Log manager domain's call-back formats . . . 680
Modules 680
Exits 682
Lock manager domain (LM) 682
Lock manager domain's specific gates 683
Lock manager domain's generic gates 685
Modules 685
Message domain (ME) 686
Message domain specific gates 686
Message domain generic gates. 701
Modules 702
Exits 703
Markup language domain (ML) 703
Markup language domain's specific gates . . . 703
Modules 716
Monitoring domain (MN) 716
Monitoring domain's specific gates 716
Monitoring domain's generic gates 737
Modules 739

© Copyright IBM Corp. 1997, 2017

Exits 740
Managed platform (MP) domain 740
Managed platform domain's specific gates. . . 740
Managed platform domain's generic gates. . . 751
Modules 752
Enqueue domain (NQ) 752
Enqueue domain's specific gates 752
Enqueue domain's generic gates 771
Enqueue domain's call-back gates 771
Modules 772
Exits 773
Object transaction service domain (OT). 773
Specific gates for the object transaction service domain 773
Modules 780
Parameter manager domain (PA) 781
Parameter manager domain's specific gates . . 781
Parameter manager domain's generic gates . . 783
Modules 784
Program manager domain (PG) 784
Program Manager domain's specific gates . . . 784
Program manager domain's generic gates . . . 866
Modules 867
Pipeline manager domain (PI) 868
Pipeline manager domain's specific gates . . . 868
Pipeline manager domain's generic gates . . . 916
Modules 916
Partner management domain (PT) 917
Partner management domain's specific gates 917
Modules 929
Resource life-cycle domain (RL) 930
Resource life-cycle domain's specific gates. . . 930
Resource life-cycle domain's generic gates. . . 941
Resource life-cycle domain's call-back formats 941
Modules 947
Recovery manager domain (RM) 948
Recovery manager domain's specific gates. . . 948
Recovery manager domain's callback formats 996
Modules. 1009
Region status domain (RS) 1012
Region status domains specific gates 1012
Region status domains generic gates 1019
Modules. 1020
RRMS domain (RX) 1020
RRMS domain's specific gates 1020
RRMS domain's call-back gates 1024
Modules. 1024
Request streams domain (RZ) 1024
Request streams domain's specific gates . . . 1024
Modules. 1033
Scheduler services domain (SH) 1034
Scheduler services domain's specific gates . . 1034
Scheduler services domain's generic gates . . 1038
Scheduler domain's call-back gates 1039
Modules. 1040
JVM domain (SJ) 1041
JVM domain's specific gates 1041
JVM domain's generic gates 1062
Modules. 1063
Storage manager domain (SM) 1063
Storage manager domain's specific gates . . . 1063
Storage manager domain's generic gates . . . 1096
Storage manager domain's generic formats 1096
Modules. 1101
Sockets domain (SO) 1103
Sockets domain's specific gates 1103
Sockets domain's generic gates 1163
Modules. 1163
Statistics domain (ST) 1165
Statistics domain's specific gates. 1165
Statistics domain's generic gates. 1171
Statistics domain's generic gates. 1171
Statistics domain's generic formats 1172
Modules. 1174
Timer domain (TI) 1174
Timer domain's specific gates. 1175
Timer domain's generic gates. 1183
Timer domain's generic formats 1183
Modules. 1184
Trace domain (TR) 1184
Trace domain's specific gates 1184
Trace domain's generic gates 1192
Modules. 1192
Temporary storage domain (TS domain) 1193
Temporary storage domain's specific gates . . 1193
Temporary storage domain's generic gates . . 1226
Temporary storage domain's callback formats 1226
Modules. 1227
Exits 1229
User domain (US) 1229
User domain's specific gates 1229
User domain's generic gates 1251
Modules. 1251
Web domain (WB) 1252
Web domain's specific gates 1252
Web domain's generic gates 1320
Web domain's call-back gates. 1321
Modules. 1321
Exits 1322
Web 2.0 domain (W2) 1322
Web 2.0 domain's specific gates 1322
Modules. 1334
Exits 1335
Transaction manager domain (XM) 1335
Transaction manager domain's specific gates 1335
Transaction manager domain's generic gates 1397
Transaction Manager domain's callback formats 1397
Transaction manager domain's generic formats 1400
Modules. 1402
Exits 1404
Security domain (XS) 1404
Security domain's specific gates 1404
Security manager domain's generic gates. . . 1450
Modules. 1450

Chapter 2. CICS Components 1453

Autoinstall for terminals, consoles and APPC connections. 1453
Design overview 1453
Modules. 1459
Diagnosing autoinstall problems 1460
Diagnosing console autoinstall problems . . . 1462

z/OS Communications Server exits. 1462
Trace 1462
Autoinstall terminal model manager 1463
Functions provided by the autoinstall terminal model manager 1463
Modules. 1467
Exits 1468
Trace 1468
Basic mapping support 1468
Design overview 1468
Control blocks. 1471
Modules. 1473
Copy books 1492
Exits 1493
Trace 1493
Builders 1493
Design overview 1494
Control blocks. 1511
Modules. 1512
Diagnosing problems with the builders . . . 1517
Exits 1517
Trace 1517
Messages 1518
Built-in functions 1519
Design overview 1519
Modules. 1519
Exits 1520
Trace 1520
CICS-DB2 Attachment Facility 1520
Design overview 1520
Control blocks. 1526
Modules. 1527
Exits 1528
Trace 1528
Statistics. 1528
CICS Web support and the CICS business logic interface. 1528
Command interpreter 1532
CSD utility program (DFHCSDUP) 1533
Database control (DBCTL). 1535
Data interchange program 1546
Design overview 1546
Modules. 1547
Exits 1547
Trace 1547
Distributed program link 1547
Modules. 1549
Exits 1549
Trace 1549
Distributed transaction processing 1549
Design overview 1550
Modules. 1551
Exits 1559
Trace 1559
DL/I database support 1559
Design overview 1559
Control blocks. 1561
Modules. 1562
Exits 1562
Trace 1563
Dump utility program (DFHDU710) 1563
Design overview 1563
Modules. 1564
Copy books 1565
Exits 1565
Trace 1565
Dynamic allocation sample program (IBM 3270 only) 1565
Design overview 1565
Control blocks. 1567
Modules. 1567
Exits 1567
Trace 1567
External interfaces 1567
ECI over TCP/IP 1567
Design overview 1568
EXEC interface 1570
Design overview 1570
Control blocks. 1571
Modules. 1573
Exits 1583
Trace 1583
Execution diagnostic facility (EDF) 1583
External CICS interface. 1584
Design overview 1584
Modules. 1586
Exits 1587
Trace 1587
Field engineering program 1588
Design overview 1588
Modules. 1588
Exits 1588
Trace 1588
File control 1588
Design overview 1589
Control blocks. 1600
Modules. 1612
Parameter lists 1651
File Control's call back gates 1695
Exits 1696
Trace 1696
Front end programming interface (FEPI) 1697
Design overview 1697
Control blocks. 1700
Dump 1701
FEPI and the z/OS Communications Server 1702
Modules. 1703
Function shipping 1706
Design overview 1707
Exits 1721
Trace 1721
Good morning message program 1722
Design overview 1722
Modules. 1722
Exits 1722
Trace 1722
Interregion communication (IRC) 1722
Design overview 1723
Control blocks. 1723
Modules. 1728
Exits 1732
Trace 1732

Intersystem communication (ISC) 1732
Interval control 1732
Design overview 1733
Control blocks. 1733
Modules. 1734
Exits 1734
Trace 1734
Language Environment interface 1734
Design overview 1734
Control blocks. 1738
Modules. 1738
Exits 1738
Trace 1738
External interfaces 1739
Master terminal program 1746
Design overview 1746
Modules. 1746
Exits 1746
Trace 1746
Message generation program 1746
Design overview 1747
Modules. 1747
Exits 1747
Trace 1747
Message switching 1747
Design overview 1748
Control blocks. 1749
Modules. 1749
Exits 1749
Trace 1749
External interfaces 1749
Multiregion operation (MRO) 1749
Node abnormal condition program 1750
Design overview 1750
Control blocks. 1753
Modules. 1753
Exits 1753
Trace 1753
Statistics. 1754
Node error program. 1754
Design overview 1754
Modules. 1754
Exits 1755
Trace 1755
Program control 1755
Design overview 1755
Modules. 1756
Exits 1757
Trace 1758
Program error program. 1758
Design overview 1758
Control blocks. 1758
Modules. 1758
Exits 1758
Trace 1759
Program preparation utilities 1759
Design overview 1759
Modules. 1759
Exits 1759
Trace 1760
Remote DL/I 1760
Design overview 1760
Control blocks. 1760
Resource definition online (RDO) 1761
Design overview 1761
Modules. 1762
Exits 1763
Trace 1764
SAA Communications and Resource Recovery interfaces 1764
Design overview 1765
Functions provided by the CPI component 1766
Modules. 1768
Exits 1768
Trace 1768
Statistics utility program (DFHSTUP) 1768
Design overview 1769
Modules. 1771
Storage control macro-compatibility interface . . 1772
Design overview 1772
Modules. 1773
Exits 1773
Trace 1773
Subsystem interface 1773
Functional overview. 1773
Design overview 1774
Control Blocks. 1775
Modules. 1776
Exits 1777
Trace 1777
External interfaces 1777
Subtask control 1777
Design overview 1777
Control blocks. 1780
Modules. 1780
Exits 1781
Trace 1781
External interfaces 1781
Syncpoint program 1781
Design overview 1781
Control blocks. 1782
Modules. 1783
Exits 1784
Trace 1784
System dump formatting program 1784
Design overview 1784
Modules. 1785
Exits 1787
Trace 1787
External interfaces 1787
System recovery program 1787
Design overview 1787
Modules. 1792
Exits 1792
Trace 1792
System spooler interface 1792
Design overview 1793
Modules. 1794
Exits 1794
Trace 1794
Table manager. 1794
Design overview 1794

Control blocks 1799
Modules 1800
Exits 1800
Trace 1800
Table Management Statistics 1800
Task-related user exit control 1800
Functional overview 1800
Design overview 1802
Control blocks 1805
Modules 1806
Exits 1807
Trace 1807
External interfaces 1807
Task-related user exit recovery 1807
Design overview 1807
Modules 1810
Exits 1810
Trace 1810
External interfaces 1811
Terminal abnormal condition program 1811
Design overview 1811
Modules 1813
Exits 1813
Trace 1814
Terminal control 1814
Design overview 1814
Control blocks 1831
Modules 1833
Exits 1835
Trace 1835
Terminal error program 1836
Design overview 1836
Modules 1836
Exits 1836
Trace 1836
Trace control macro-compatibility interface . . . 1836
Design overview 1837
Modules 1837
Exits 1837
Trace 1837
Trace formatting 1838
Design overview 1838
Control blocks 1841
Modules 1841
Exits 1842
Transaction Failure program 1842
Design overview 1842
Modules 1844
Exits 1844
Trace 1844
Transaction restart program 1844
Design overview 1844
Control blocks 1845
Modules 1845
Exits 1845
Trace 1845
Transaction Restart Statistics 1845
Transaction routing 1845
Design overview 1846
Control blocks 1863
Modules 1865
Exits 1866
Trace 1866
Transient data control 1866
Design overview 1866
Modules 1871
Exits 1871
Trace 1871
User exit control 1871
Design overview 1871
Control blocks 1875
Modules 1876
Exits 1877
Trace 1877
WTO and WTOR 1877
Design overview 1877
Modules 1878
Exits 1878
Trace 1878
z/OS Communications Server generic resource 1878
Design Overview 1878
Generic resource and LU6.1/LU6.2 1879
Ending affinities 1880
Generic resource and ATI 1881
Modules 1881
Problem solving for generic resource 1883
z/OS Communications Server LU6.2 1885
Design overview 1885
Modules 1888
Exits 1895
Trace 1895
z/OS Communications Server persistent sessions support 1895
Design overview 1895
Modules 1903
Diagnosing persistent sessions problems . . . 1905
Persistent sessions status byte (TCTE_PRSS) 1908
Bid status byte (TCTE_BID_STATUS) 1910
Summary of persistent session waits 1911
z/OS Communications Server exits 1911
Trace 1912
Statistics 1912

Chapter 3. CICS executable modules 1913
DFHACP 1913
DFHAICBP 1913
DFHALP 1914
DFHAMP 1914
DFHAPJC 1914
DFHAPSIP 1915
DFHAPRX 1915
DFHAPST 1916
DFHAPTD 1916
DFHAPTI 1917
DFHAPTIM 1917
DFHAPTIX 1917
DFHASV 1918
DFHBSIB3 1918
DFHBSIZ1 1918
DFHBSIZ3 1918
DFHBSMIR 1919
DFHBSMPP 1919

DFHBSM61. 1919
DFHBSM62. 1919
DFHBSS. 1920
DFHBSSA 1920
DFHBSSF 1920
DFHBSSS 1920
DFHBSSZ 1921
DFHBSSZB. 1921
DFHBSSZG 1921
DFHBSSZI 1922
DFHBSSZL. 1922
DFHBSSZM 1922
DFHBSSZP. 1922
DFHBSSZR. 1923
DFHBSSZS. 1923
DFHBSSZ6. 1923
DFHBST. 1923
DFHBSTB 1924
DFHBSTBL. 1924
DFHBSTB3. 1924
DFHBSTC 1924
DFHBSTD 1925
DFHBSTE 1925
DFHBSTH 1925
DFHBSTI 1925
DFHBSTM. 1926
DFHBSTO 1926
DFHBSTP3. 1926
DFHBSTS 1927
DFHBSTT 1927
DFHBSTZ 1927
DFHBSTZA 1927
DFHBSTZB. 1928
DFHBSTZC 1928
DFHBSTZE. 1928
DFHBSTZH 1928
DFHBSTZL. 1929
DFHBSTZO 1929
DFHBSTZP. 1929
DFHBSTZR. 1929
DFHBSTZS. 1930
DFHBSTZV 1930
DFHBSTZZ. 1930
DFHBSTZ1. 1931
DFHBSTZ2. 1931
DFHBSTZ3. 1931
DFHBSXGS 1931
DFHBSZZ 1932
DFHBSZZS. 1932
DFHBSZZV 1932
DFHCAPB 1932
DFHCCNV. 1933
DFHCMP 1933
DFHCPY 1933
DFHCRC 1934
DFHCRNP. 1934
DFHCRQ 1934
DFHCRR 1934
DFHCRS 1935
DFHCRSP 1935
DFHCRT 1935
DFHCSA 1936
DFHCSDUP 1936
DFHCSSC 1936
DFHCSVC 1937
DFHCSVCU 1937
DFHCUCAB 1938
DFHCUCB. 1938
DFHCUCCB 1938
DFHCUCDB 1939
DFHCWTO 1939
DFHDBAT. 1939
DFHDBCON 1940
DFHDBCR. 1940
DFHDBCT. 1940
DFHDBCTX 1941
DFHDBDI. 1941
DFHDBDSC 1941
DFHDBIQ 1942
DFHDBME. 1942
DFHDBMOX 1942
DFHDBP 1942
DFHDBREX 1943
DFHDBSPX 1943
DFHDBSSX 1943
DFHDBSTX 1944
DFHDBTOX 1944
DFHDBUEX 1944
DFHDCP 1945
DFHDES 1945
DFHDIP. 1945
DFHDLI. 1946
DFHDLIAI. 1946
DFHDLIDP 1946
DFHDLIRP. 1946
DFHDMP 1947
DFHDRPG. 1947
DFHDSBA$, DFHDSB1$ 1948
DFHDU710. 1948
DFHDXACH 1948
DFHDXSTM 1949
DFHDYP 1949
DFHEAG 1949
DFHEAG0. 1949
DFHEAI. 1950
DFHEAI0 1950
DFHEAP1$. 1951
DFHEBF 1951
DFHEBU 1951
DFHECI. 1952
DFHECID 1952
DFHECIP 1952
DFHECP1$. 1953
DFHEDAD. 1953
DFHEDAP. 1953
DFHEDC 1954
DFHEDFBR 1954
DFHEDFD. 1954
DFHEDFM. 1955
DFHEDFP. 1955
DFHEDFR. 1955
DFHEDFX. 1955

DFHEDI. 1956
DFHEDP 1956
DFHEDP1$. 1956
DFHEEI 1957
DFHEEX 1957
DFHEFRM 1957
DFHEGL 1958
DFHEIDTI 1958
DFHEIG. 1958
DFHEIIC 1958
DFHEIFC 1959
DFHEIP 1959
DFHEIPA 1959
DFHEISC 1960
DFHEISP 1960
DFHEISR 1960
DFHEJC. 1961
DFHEKC 1961
DFHELII 1961
DFHEMS 1961
DFHEMTA 1962
DFHEMTD. 1962
DFHEMTP 1962
DFHEOTP 1962
DFHEPC 1963
DFHEPI. 1963
DFHEPP1$ 1963
DFHEPS. 1964
DFHERM 1964
DFHESTP 1964
DFHETC 1964
DFHETD 1965
DFHETL 1965
DFHETR 1965
DFHETS. 1966
DFHEXI. 1966
DFHFCAT 1966
DFHFCBD 1966
DFHFCDN. 1967
DFHFCDTS 1967
DFHFCFR 1967
DFHFCFS 1968
DFHFCL 1968
DFHFCM 1968
DFHFCMT 1969
DFHFCN 1969
DFHFCRL 1969
DFHFCRP 1970
DFHFCSD 1970
DFHFCST 1970
DFHFCU 1970
DFHFCVR 1971
DFHFCVS 1971
DFHFDP 1971
DFHFEP 1972
DFHGMM 1972
DFHHPSVC 1972
DFHICP. 1973
DFHIIPA$, DFHIIP1$ 1973
DFHIRP. 1974
DFHIRW10. 1974
DFHISP 1974
DFHISFS 1975
DFHISXF 1975
DFHJCP. 1975
DFHJUP. 1976
DFHKCP 1976
DFHKCQ 1976
DFHKCRP 1977
DFHKCSC 1977
DFHKCSP 1977
DFHLUP 1978
DFHMCPA$, DFHMCPE$, DFHMCP1$ 1978
DFHMCX 1978
DFHMGP 1979
DFHMGT 1979
DFHMIRS 1979
DFHML1 1980
DFHMROQP 1981
DFHMSP 1981
DFHMXP 1982
DFHM32A$, DFHM321$ 1982
DFHPBPA$, DFHPBP1$ 1982
DFHPD710 1983
DFHPEP 1983
DFHPHP 1984
DFHPL1OI 1984
DFHPRK 1984
DFHPSP. 1985
DFHPSPDW 1985
DFHPSPSS 1985
DFHPSPST 1986
DFHPSSVC. 1986
DFHPUP 1986
DFHP3270 1986
DFHQRY 1987
DFHRCEX 1987
DFHRKB 1987
DFHREST 1988
DFHRLRA$, DFHRLR1$ 1988
DFHRMSY 1988
DFHRTC 1989
DFHRTE 1989
DFHSFP. 1989
DFHSIA1 1990
DFHSIB1 1990
DFHSIC1 1990
DFHSID1 1990
DFHSIF1 1991
DFHSIG1 1991
DFHSIH1 1991
DFHSII1. 1992
DFHSIJ1. 1992
DFHSIP 1993
DFHSKP 1993
DFHSMSCP 1994
DFHSNAT 1994
DFHSNNFY 1994
DFHSNP 1995
DFHSNSN 1995
DFHSNVCL 1995
DFHSNVID 1996

DFHSNVPR 1996
DFHSNVTO 1996
DFHSPP. 1996
DFHSRLI 1997
DFHSRP 1997
DFHSSEN 1998
DFHSSGC 1998
DFHSSIN 1998
DFHSSMGP 1999
DFHSSMGT 1999
DFHSSWT 1999
DFHSSWTF 1999
DFHSSWTO 2000
DFHSTDT 2000
DFHSTFC 2000
DFHSTIB 2001
DFHSTJC 2001
DFHSTLK 2001
DFHSTLS 2001
DFHSTP. 2002
DFHSTSZ 2002
DFHSTTD 2003
DFHSTTM 2003
DFHSTTR 2003
DFHSTTS 2004
DFHSUSN 2004
DFHSUSX 2004
DFHSUZX 2005
DFHTACP 2005
DFHTAJP 2006
DFHTBSB 2006
DFHTBSBP. 2006
DFHTBSD 2007
DFHTBSDP 2007
DFHTBSL 2007
DFHTBSLP. 2007
DFHTBSQ 2008
DFHTBSQP 2008
DFHTBSR 2008
DFHTBSRP. 2008
DFHTBSSP. 2009
DFHTBS00 2009
DFHTCBP 2009
DFHTCP 2009
DFHTCRP 2010
DFHTCRPC 2010
DFHTCRPL 2011
DFHTCRPS. 2011
DFHTCRPU 2011
DFHTDA 2011
DFHTDB 2012
DFHTDEXL 2012
DFHTDP 2013
DFHTDQ 2013
DFHTDRM. 2013
DFHTDRP 2014
DFHTDTM. 2014
DFHTDX 2014
DFHTEP 2015
DFHTMP 2015
DFHTON 2016
DFHTOR 2016
DFHTORP 2016
DFHTPPA$, DFHTPP1$ 2017
DFHTPQ 2017
DFHTPR 2017
DFHTPS. 2018
DFHTRAP 2019
DFHTR710 and AMDUSREF 2019
DFHTRP 2020
DFHTRZCP 2020
DFHTRZIP. 2020
DFHTRZPP 2020
DFHTRZXP 2021
DFHTRZYP 2021
DFHTRZZP 2021
DFHTSP. 2021
DFHTU710. 2022
DFHUCNV. 2022
DFHUEH 2022
DFHUEM 2023
DFHUSBP 2023
DFHWCCS. 2023
DFHWCGNT 2024
DFHWDATT 2024
DFHWDINA 2024
DFHWDISP 2025
DFHWDSRP 2025
DFHWDWAT 2025
DFHWKP 2026
DFHWLFRE 2026
DFHWLGET 2026
DFHWMG1 2026
DFHWMI 2027
DFHWMMT 2027
DFHWMPG 2027
DFHWMP1. 2028
DFHWMQG 2028
DFHWMQH 2028
DFHWMQP 2028
DFHWMQS 2029
DFHWMRD 2029
DFHWMS 2029
DFHWMS20 2030
DFHWMWR 2030
DFHWOS 2030
DFHWOSA. 2030
DFHWOSB. 2031
DFHWSRTR 2031
DFHWSSN1 2031
DFHWSSN2 2032
DFHWSSN3 2032
DFHWSSOF 2032
DFHWSSR 2033
DFHWSSW. 2033
DFHWSTI 2033
DFHWSTKV 2034
DFHWSXPI 2035
DFHWTI 2035
DFHWTRP. 2035
DFHXCP 2036
DFHXCPC 2036

DFHXCP1 2036
DFHXFP 2036
DFHXFQ 2037
DFHXFX 2037
DFHXRA 2037
DFHXRB 2038
DFHXRC 2038
DFHXRCP 2038
DFHXRE 2038
DFHXRP 2039
DFHXRSP 2039
DFHXSMN 2039
DFHXSMX 2040
DFHXSS. 2040
DFHXSSB 2040
DFHXSWM 2041
DFHXTCI 2041
DFHXTP 2041
DFHZABD 2041
DFHZACT 2042
DFHZAIT 2042
DFHZAND. 2042
DFHZARER 2043
DFHZARL 2043
DFHZARM. 2044
DFHZARQ 2044
DFHZARR 2044
DFHZARRA 2045
DFHZARRC 2045
DFHZARRF 2045
DFHZASX 2046
DFHZATA 2046
DFHZATD 2046
DFHZATDX 2047
DFHZATI 2047
DFHZATMD 2047
DFHZATMF 2048
DFHZATR 2048
DFHZATS 2048
DFHZATT 2049
DFHZBAN 2049
DFHZBKT 2049
DFHZBLX 2049
DFHZCA 2050
DFHZCB 2050
DFHZCC 2051
DFHZCHS 2052
DFHZCLS 2052
DFHZCLX 2052
DFHZCNA 2053
DFHZCNR 2053
DFHZCNT 2053
DFHZCP 2054
DFHZCQ 2054
DFHZCQDL 2055
DFHZCQIN 2055
DFHZCQIQ 2055
DFHZCQIS. 2055
DFHZCQIT 2056
DFHZCQRS 2056
DFHZCRQ 2056
DFHZCRT 2056
DFHZCUT 2057
DFHZCW 2057
DFHZCX 2057
DFHZCXR 2058
DFHZCY 2058
DFHZCZ 2059
DFHZDET 2059
DFHZDSP 2060
DFHZDST 2060
DFHZEMW 2060
DFHZERH 2061
DFHZEV1 2061
DFHZEV2 2061
DFHZFRE 2061
DFHZGET 2062
DFHZHPRX 2062
DFHZHPSR 2062
DFHZISP 2063
DFHZIS1 2063
DFHZIS2 2063
DFHZLEX 2064
DFHZLGX 2064
DFHZLOC 2064
DFHZLRP 2065
DFHZLTX 2065
DFHZLUS 2065
DFHZNAC. 2066
DFHZNEP 2066
DFHZNSP 2066
DFHZOPA 2067
DFHZOPN 2067
DFHZOPX 2067
DFHZQUE 2068
DFHZRAC 2068
DFHZRAQ 2068
DFHZRAR 2069
DFHZRAS 2069
DFHZRLG 2069
DFHZRLP 2069
DFHZRLX 2070
DFHZRRX 2070
DFHZRSP 2070
DFHZRST 2071
DFHZRSY 2071
DFHZRVL 2071
DFHZRVS 2071
DFHZRVX 2072
DFHZSAX 2072
DFHZSCX 2072
DFHZSDA 2073
DFHZSDL 2073
DFHZSDR 2073
DFHZSDS 2074
DFHZSDX 2074
DFHZSES 2074
DFHZSEX 2075
DFHZSHU 2075
DFHZSIM 2075
DFHZSIX 2076
DFHZSKR 2076

DFHZSLS 2076
DFHZSLX 2077
DFHZSSX 2077
DFHZSTAP 2077
DFHZSTU 2077
DFHZSUP 2078
DFHZSYN 2078
DFHZSYX 2078
DFHZTAX 2079
DFHZTPX 2079
DFHZTRA 2079
DFHZTSP 2080
DFHZUCT 2080
DFHZUIX 2080
DFHZUSR 2080
DFHZXCU 2081
DFHZXQO 2081
DFHZXRC 2081
DFHZXRE0 2082
DFHZXRL 2082
DFHZXRT 2082
DFHZXST 2083

Notices 2085

Index 2091

About this PDF

This PDF is a reference of the components that make up CICS. It also provides reference tables of CICS source modules and executable modules.

For details of the terms and notation used in this book, see Conventions and terminology used in the CICS documentation in IBM Knowledge Center.

Date of this PDF

This PDF was created on June 6th 2017.

© Copyright IBM Corp. 1997, 2017

Chapter 1. CICS domains

A description of the domains into which CICS is organized, and the functions within these domains.

CICS domains

At the top level, CICS® is organized into *domains*. With the exception of the application domain, which contains several components, each domain is a single major component of CICS.

Domains never communicate directly with each other. Calls between domains are routed through kernel linkage routines. Calls can be made only to official interfaces to the domains, and they must use the correct protocols. This structure is shown in Figure 1.

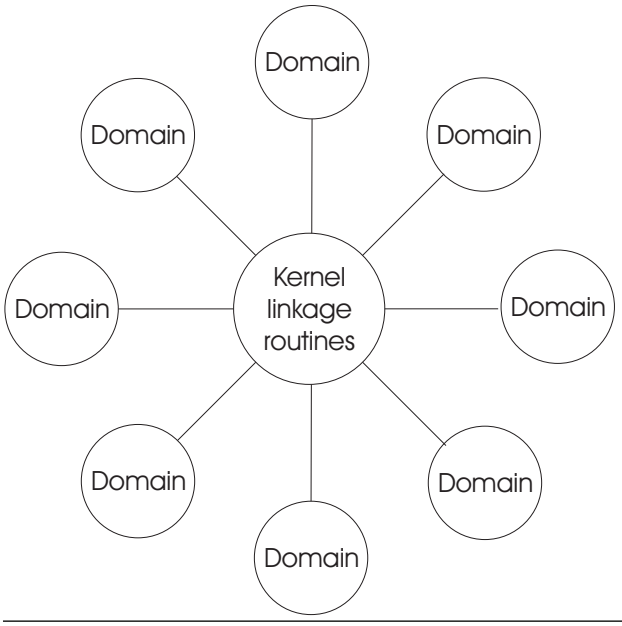


Figure 1. CICS organization—domains

Each domain manages its own data. No domain accesses another domain's data directly. If a domain needs data belonging to another domain, it must call that domain, and that domain then passes the data back in the caller's parameter area.

The CICS domains are shipped as object code only (OCO); no source code is available for the domains.

The following table lists the CICS domains alphabetically by domain identifier. For each domain, the table also shows whether or not the domain is OCO, and gives a reference to the section describing the interfaces to the domain.

© Copyright IBM Corp. 1997, 2017

Domain ID	Domain	OCO?	See topic
AP	Application	See note	"Application manager domain (AP)" on page 10
AS	Asynchronous Services Domain	Yes	"Asynchronous services domain (AS)" on page 7
BA	Business Application Manager	Yes	"Business application manager domain (BA)" on page 335
CC	Local catalog	Yes	"CICS catalog domain (CC)" on page 367
DD	Directory manager	Yes	"Directory manager domain (DD)" on page 373
DH	Document handler	Yes	"Document handler domain (DH)" on page 384
DM	Domain manager	Yes	"Domain manager domain (DM)" on page 409
DP	Debugging profile domain	Yes	"Debugging profile domain (DP)" on page 419
DS	Dispatcher	Yes	"Dispatcher domain (DS)" on page 454
DU	Dump	No	"Dump Domain (DU)" on page 493
EJ	Enterprise Java™	No	"Enterprise Java domain (EJ)" on page 519
EM	Event manager	Yes	"Event manager domain (EM)" on page 521
EP	Event processing	Yes	"Event processing domain (EP)" on page 533
GC	Global catalog	Yes	"CICS catalog domain (CC)" on page 367
IE	IP ECI	Yes	"IP ECI (IE) domain" on page 543
IS	Inter-system (IS) domain	Yes	"Inter-system (IS) domain" on page 546

Domain ID	Domain	OCO?	See topic
KE	Kernel	Yes	"Kernel domain (KE)" on page 582
LD	Loader	Yes	"Loader domain (LD)" on page 612
LG	Log manager	Yes	"Log manager domain (LG)" on page 645
LM	Lock manager	Yes	"Lock manager domain (LM)" on page 682
ME	Message	Yes	"Message domain (ME)" on page 686
ML	Markup language	Yes	"Markup language domain (ML)" on page 703
MN	Monitoring	Yes	"Monitoring domain (MN)" on page 716
MP	Managed Platform	Yes	"Managed platform (MP) domain" on page 740
NQ	Enqueue	Yes	"Enqueue domain (NQ)" on page 752
OT	Object transaction service	No	"Object transaction service domain (OT)" on page 773
PA	Parameter manager	Yes	"Parameter manager domain (PA)" on page 781
PG	Program manager	Yes	"Program manager domain (PG)" on page 784
PI	Pipeline manager	Yes	"Pipeline manager domain (PI)" on page 868
PT	Partner manager	Yes	"Partner management domain (PT)" on page 917
RL	Resource lifecycle	Yes	"Resource life-cycle domain (RL)" on page 930
RM	Recovery manager	Yes	"Recovery manager domain (RM)" on page 948

Domain ID	Domain	OCO?	See topic
RS	Region status	Yes	"Region status domain (RS)" on page 1012
RX	Resource recovery service	Yes	"RRMS domain (RX)" on page 1020
RZ	Request Stream	No	"Request streams domain (RZ)" on page 1024
SH	Scheduler services	Yes	"Scheduler services domain (SH)" on page 1034
SJ	JVM Domain	No	"JVM domain (SJ)" on page 1041
SM	Storage manager	Yes	"Storage manager domain (SM)" on page 1063
SO	Sockets Domain	No	"Sockets domain (SO)" on page 1103
ST	Statistics	Yes	"Statistics domain (ST)" on page 1165
TI	Timer	Yes	"Timer domain (TI)" on page 1174
TR	Trace	No	"Trace domain (TR)" on page 1184
TS	Temporary storage	Yes	"Temporary storage domain (TS domain)" on page 1193
US	User	Yes	"User domain (US)" on page 1229
WB	Web	Yes	"Web domain (WB)" on page 1252
W2	Web 2.0	Yes	"Web 2.0 domain (W2)" on page 1322
XM	Transaction manager	Yes	"Transaction manager domain (XM)" on page 1335
XS	Security manager	Yes	"Security domain (XS)" on page 1404

Note: The application domain is mainly non-OCO, but it contains these OCO components:

- CICS data table services

- RDO for VSAM files and LSR pools
- Some EXEC CICS system programming functions
- Autoinstall terminal model manager
- Partner resource manager
- SAA Communications and Resource Recovery
- Some of the file control functions
- Recovery manager connectors interfaces.

The offline statistics utility program (DFHSTUP) and the system dump formatting routines are also treated as OCO.

Domain gates

A *domain gate* is an entry point or interface to a domain. It can be called by any authorized caller who needs to use some function provided by the domain.

A number of domain functions are available through the exit programming interface (XPI).

In practice, every domain has several gates. Each gate has a 4-character identifier; the first two characters are the identifier of the owning domain, and the second two characters differentiate between the functions of the domain's gates. Here, for example, are two of the dispatcher (DS) domain's gates:

DSAT
DSSR

Functions provided by gates

An individual gate can provide many functions. The required function is determined by the parameters included on the call. The DSSR gate of the DS domain, for example, provides all these functions:

ADD_SUSPEND
DELETE_SUSPEND
INQUIRE_SUSPEND_TOKEN
RESUME
SUSPEND
WAIT_MVS™
WAIT_OLDC
WAIT_OLDW.

Specific gates, generic and call-back gates

It is useful to distinguish between **specific gates**, **generic gates** and **callback gates**:

- A specific gate gives access to a set of functions that are provided by that domain alone. The functions are likely to be requested by many different callers.

  DS domain, for example, has a specific gate (DSAT) that provides CHANGE_MODE and CHANGE_PRIORITY functions (among other functions). Only the DS domain provides those functions, but they can be requested by many different callers.
- A generic gate gives access to a set of functions that are provided by several domains.

  Most domains provide a QUIESCE_DOMAIN function, for example, so that they can be quiesced when CICS is shutting down normally. They each have a generic gate that provides this function. DM domain makes a **generic call** to that gate in any domain that is to be quiesced.

- A call-back gate also gives access to a set of functions that can be provided by several domains. Unlike a generic gate where the call is broadcast to all domains that have provided a gate a call-back is restricted to specific domains but uses a format owned by the calling domain.

  For example the Recovery Manager calls the domains that have registered an interest in syncpoint processing using the PERFORM_PREPARE function format that it owns.

Domain call formats

Any module calling a domain gate must use the correct **format** for the call. The format represents the parameter list structure. It describes the parameters that must be provided on the call (the **input** parameters), and the parameters that are returned to the caller when the request has been processed (the **output** parameters).

For example, Table 1 lists the input and output parameters for the ATTACH function of the DS domain's DSAT gate.

Table 1. Domain call formats

Input parameters	Output parameters
PRIORITY	TASK_TOKEN
USER_TOKEN	RESPONSE
[TIMEOUT]	[REASON]
TYPE	
[MODE]	
[TASK_REPLY_GATE_INDEX]	
[SPECIAL_TYPE]	

Parameters not shown in brackets are mandatory, and are always interpreted in trace entries. Parameters shown in brackets are optional, and are in trace entries only if values have been set. An exception to this rule is that, regardless of whether REASON is mandatory or optional for a particular function, its value is included in a trace entry only for a non-'OK' response.

The domain call formats described are in the sections dealing with the domains that own them, as discussed in "Ownership of formats."

Ownership of formats

Every format is 'owned' by a domain:

- The formats for specific calls are owned by the domain being called. DS domain, for example, owns the format for the CHANGE_MODE and CHANGE_PRIORITY calls. This book uses the term **specific format** to refer to such formats.
- The formats for generic calls and call-back calls are owned by the calling domain. DM domain, for example, owns the format for calls to (generic) gates providing the QUIESCE_DOMAIN function in other domains. This book uses the term **generic format** to refer to such formats.

Tokens

Tokens are passed as parameters on many domain calls. They identify uniquely objects that are operands of domain functions.

Here are some examples:

TASK_TOKEN
uniquely identifies a task to be used as the operand of a function.
DOMAIN_TOKEN
uniquely identifies a domain to be used as the operand of a function.
SUSPEND_TOKEN
uniquely identifies a task for the purpose of a suspend or resume dialog.

The BROWSE_TOKEN parameter on domain interfaces

Some domains provide functions that callers can use to browse through a set of objects in the domain. These functions normally use a *browse token* that encapsulates the state of the browse operation.

The browse token is represented in most cases by the **BROWSE_TOKEN** parameter, although some domains use a different name.

1. The called domain creates the token when the calling domain issues a START_BROWSE request, and returns it to the caller.
2. The calling domain passes the token to the called domain on GET_NEXT and similar requests. The called domain uses the token to distinguish concurrent browse operations from one another, and to maintain the state of the browse operation.
3. Finally the calling domain passes the token to the called domain on an END_BROWSE request, after which the token is invalid.

RESPONSE parameter on domain interfaces

All domain calls return the **RESPONSE** parameter to indicate whether the call was successful.

The **RESPONSE** parameter has the following values:

OK The requested function has been completed successfully.

EXCEPTION
Processing of the function could not be completed, and the domain state is unchanged. More information is given in the **REASON** parameter.

DISASTER
The domain could not complete the request because of some irrecoverable system problem. More information is given in the **REASON** parameter.

INVALID
The parameter list is not valid. More information is given in the **REASON** parameter.

KERNERROR
The kernel was unable to call the required function gate.

PURGED
A purge has been requested for the task making the domain call.

Asynchronous services domain (AS)

The asynchronous services domain provides the functions within asynchronous APIs to attach asynchronous child tasks, and then get responses back from the child tasks.

Asynchronous services domain (AS)

The asynchronous services domain provides the functions within asynchronous APIs to attach asynchronous child tasks, and then get responses back from the child tasks.

Asynchronous services domain's specific gates

The specific gates provide access for other domains to functions that are provided by the AS domain.

ASAS gate, RUN_TRANSACTION function:

The RUN_TRANSACTION function creates an asynchronous child.

Input Parameters

transaction_id
is the ID of the child transaction which is run asynchronously to the parent task.

channel
Optional Parameter

is a token referencing the channel that is used for passing data between parent and child tasks, if required.

Output Parameters

child_token
is a token representing the child task.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
tran_not_found
tran_disabled
tran_shutdown_disabled
not_authorized
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ASAS gate, FETCH_CHILD function:

The FETCH_CHILD function fetches the response from a named asynchronous child.

Input Parameters

child_token
is a token representing the child that is fetched.

Output Parameters

channel_name
Optional Parameter

is the name of the fetched channel.

completion_status
Optional Parameter

is the completion status of the child.

abend_code
Optional Parameter

is the abend code of the child.

REASON
The following values are returned when RESPONSE is EXCEPTION:
`invalid_child_token`
`channel_already_fetched`
`timed_out`
`notfinished`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ASAS gate, FETCH_ANY function:

The FETCH_ANY function fetches the response from any asynchronous child.

Output Parameters

child_token
is a token representing the child that is fetched.

channel_name
Optional Parameter

is the name of the fetched channel.

completion_status
Optional Parameter

is the completion status of the child.

abend_code
Optional Parameter

is the abend code of the child.

REASON
The following values are returned when RESPONSE is EXCEPTION:
`invalid_child_token`
`channel_already_fetched`
`timed_out`
`notfinished`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Asynchronous services domain's generic gates

Table 2 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 2. Asynchronous services domain's generic gates

Gate	Trace	Function	Format
DMDM	AS 0100 AS 0101	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
XMNT	AS 0400 AS 0401	MXT_NOTIFY MXT_CHANGE_NOTIFY	XMNT

Asynchronous services domain's call-back gates

Table 3 on page 10 summarizes the domain's call-back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 3. Asynchronous services domain's call-back gates

Gate	Trace	Function	Format
ASXM	AS 0200 AS 0201	INIT_XM_CLIENT BIND_XM_CLIENT TRANSACTION_HANG ABEND_TERMINATE RELEASE_XM_CLIENT	XMAC

Modules

Asynchronous services (AS) domain modules.

Module	Function
DFHASDM	Domain initialization and termination program
DFHASDUF	Dump formatting
DFHASAS	Supports the asynchronous APIs.
DFHASNT	Handles the following requests: MXT_NOTIFY MXT_CHANGE_NOTIFY
DFHASTRI	Trace interpretation
DFHASXM	Transaction attach and detach processing
DFHASRM	Handles the following requests: PERFORM_COMMIT PERFORM_SHUNT

Application manager domain (AP)

The principal components of the application domain are described in "Application domain."

Application domain

Application programs are run in the application (AP) domain, which contains several major components, as shown in Figure 2 on page 11.

Most application domain CICS functions are either provided by modules that are part of the CICS nucleus, that is to say they are an integral part of the system and are loaded at system initialization time, or they are system application programs, which are loaded as needed in the same way as user application programs.

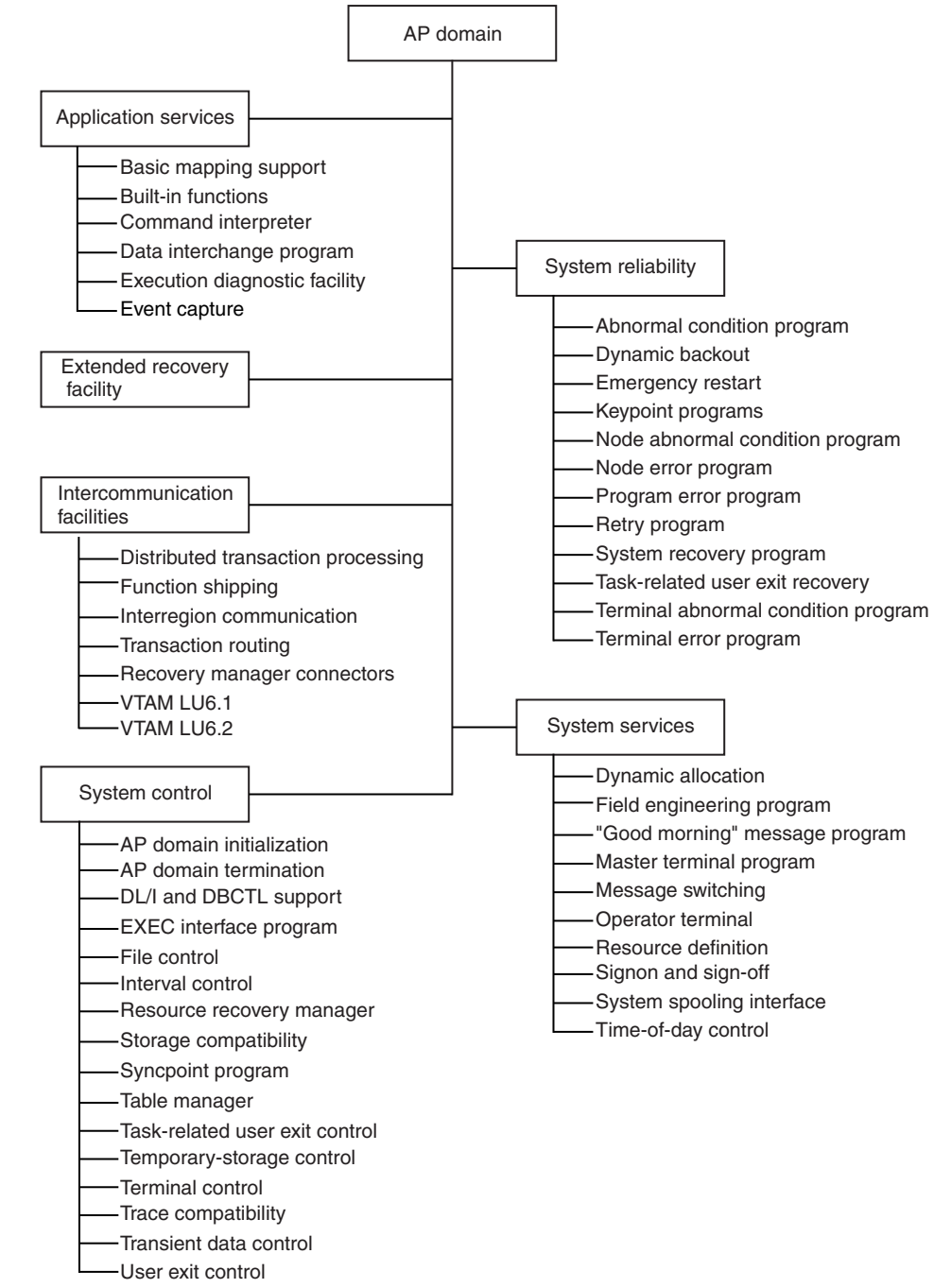


Figure 2. AP domain - major components

Application manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the AP domain.

ABAB gate, CREATE_ABEND_RECORD function

The CREATE_ABEND_RECORD function of the ABAB gate is used to create an abend record; that is, a trnasaction abend control block (TACB).

Input Parameters

ABEND_CODE
Optional parameter

The four-character transaction abend code.

ACCESS_REGISTERS
Optional parameter

The contents of the access registers at the time of a program check or operating system abend.

ALET
Optional parameter

The access list entry token (ALET) at the time of a program check or operating system abend.

ALL_FP_REGISTERS
Optional parameter

The contents of the floating point register values in the order 0 - 15 at the time of a program check or operating system abend.

BEAR
Optional parameter

The value of the breaking event address register (BEAR) at the time of a program check or operating system abend.

CURRENT_ACCESS_VALUES
Optional parameter

The current access register values are saved in the TACB.

CURRENT_FP_VALUES
Optional parameter

The current FP register values are saved in the TACB. If the task has not used the additional FP registers only the original FP registers are saved in the TACB. If any of the additional FP registers have been used by the task all the FP registers (0-15) and the FPC register are saved in the TACB.

CURRENT_VRR_VALUES
Optional parameter

The current vector register values are saved in the TACB.

ERROR_MESSAGE
Optional parameter

The error message sent from the remote system if the abend was raised by DFHZAND.

ERROR_OFFSET
Optional parameter

The offset of a program check or operating system abend in the failing application program or CICS(R) AP domain program.

EXECUTION_KEY
Optional parameter

A code that indicates the execution key at the time the abend was issued, or at the time the operating system abend or program check occurred.

FAILING_PROGRAM
Optional parameter

The name of the program in which the abend occurred.

FAILING_RESOURCE
Optional parameter

The name of the system TCTTE (the connection) if the abend was raised by DFHZAND.

FLOATING_POINT_REGISTERS
Optional parameter

The contents of the original floating point registers at the time of a program check or operating system abend.

FPC_REGISTER
Optional parameter

The value of the floating point control register at the time of a program check or operating system abend.

GENERAL_REGISTERS
Optional parameter

The contents of the general purpose registers at the time of a program check or operating system abend.

GENERAL64_REGISTERS
Optional parameter

The contents of the 64-bit general purpose registers at the time of a program check or operating system abend. This is an alternative parameter to GENERAL_REGISTERS. If GENERAL64_REGISTERS is specified, you cannot use the H64G_REGISTERS parameter.

GREG_ORDER
Optional parameter

An indication of the order of the registers passed in the GENERAL_REGISTERS and GENERAL64_REGISTERS parameters. DFHSRP saves the registers in the abend record in the order 0 - 15, and INQUIRE_ABEND_RECORD always returns them in this order.

Values for the parameter are as follows:

```
R0TOR15
R14TOR13
```

H64G_REGISTERS
Optional parameter

The contents of the high order words of the 64-bit general purpose registers at the time of a program check or operating system abend. If H64G_REGISTERS is specified, you cannot use the GENERAL64_REGISTERS parameter.

INTERRUPT_DATA
Optional parameter

The interrupt code, instruction length code, and other data at the time of a program check or operating system abend.

PSW|PSW16
Optional parameter

The contents of the program status word (PSW) or the 16 byte PSW at the time of a program check or operating system abend.

REMOTE_SYSTEM
Optional parameter

The name of the remote system if the abend was raised in the client transaction to reflect an abend occurring in the DPL server.

REQUEST_ID
Optional parameter

The request ID from the TCTTE for a terminal-oriented task.

SENSE_BYTES
Optional parameter

The SNA sense bytes if the abend was raised by DFHZAND.

SPACE
Optional parameter

An indication whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend.

Values for the parameter are as follows:

```
BASESPACE
NOSPACE
SUBSPACE
```

STATUS_FLAGS
Optional parameter

The status flags at the time of the abend.

STOKEN
Optional parameter

The subspace token (STOKEN) at the time of a program check or operating system abend.

STORAGE_TYPE
Optional parameter

A code that indicates the storage hit on an OC4 abend.

TEA
Optional parameter

The translation exception address at the time of a program check or operating system abend.

VECTOR_REGISTERS
Optional parameter

The contents of the vector registers at the time of a program check or operating system abend.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
```

ABEND_TOKEN
The token allocated by ABAB for this abend. The token must be passed on subsequent UPDATE_ABEND_RECORD and START_ABEND requests to ABAB. The token is no longer valid after a START_ABEND request.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ABAB gate, INQUIRE_ABEND_RECORD function

The INQUIRE_ABEND_RECORD function of the ABAB gate is used to inquire about an abend record (TACB).

Input Parameters

ABEND_TYPE
Optional parameter

Indicates which abend record the information is to be extracted from.

Values for the parameter are as follows:
FIRST
LASTASRA
LATEST

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is EXCEPTION:
NO_ABEND_RECORD

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ABEND_CODE

Optional parameter

The four-character transaction abend code.

ACCESS_REGISTERS

Optional parameter

The contents of the access registers at the time of a program check or operating system abend.

ALET

Optional parameter

The access list entry token (ALET) at the time of a program check or operating system abend.

ALL_FP_REGISTERS

Optional parameter

The contents of the floating point register values in the order 0 - 15 at the time of a program check or operating system abend.

BEAR

Optional parameter

The value of the breaking event address register (BEAR) at the time of a program check or operating system abend.

DUMP

Optional parameter

Indicates whether a dump was requested for this abend.

Values for the parameter are as follows:
NO
YES

ERROR_MESSAGE

Optional parameter

The error message sent from the remote system if the abend was raised by DFHZAND.

ERROR_OFFSET

Optional parameter

The offset of a program check or operating system abend in the failing application program or CICS(R) AP domain program.

EXECUTION_KEY

Optional parameter

A code that indicates the execution key at the time the abend was issued, or at the time the operating system abend or program check occurred.

FAILING_PROGRAM
Optional parameter

The name of the program in which the abend occurred.

FAILING_RESOURCE
Optional parameter

The name of the system TCTTE (the connection) if the abend was raised by DFHZAND.

FLOATING_POINT_REGISTERS
Optional parameter

The contents of the original floating point registers at the time of a program check or operating system abend.

FPC_REGISTER
Optional parameter

The value of the floating point control register at the time of a program check or operating system abend.

GENERAL_REGISTERS
Optional parameter

The contents of the general purpose registers at the time of a program check or operating system abend.

GENERAL64_REGISTERS
Optional parameter

The contents of the 64-bit general purpose registers at the time of a program check or operating system abend.

H64G_REGISTERS
Optional parameter

The contents of the high order words of the 64-bit general purpose registers at the time of a program check or operating system abend.

IGNORE_HANDLES
Optional parameter

Indicates whether this abend should be passed to any EXEC CICS HANDLE routines that are active. IGNORE_HANDLES(YES) results in EXEC CICS HANDLE being ignored at all levels of the program stack.

Values for the parameter are as follows:

```
NO
YES
```

INTERRUPT_DATA
Optional parameter

The interrupt code, instruction length code, and other data at the time of a program check or operating system abend.

PSW
Optional parameter

The contents of the program status word (PSW) at the time of a program check or operating system abend.

PSW16
Optional parameter

The contents of the 16 byte PSW at the time of a program check or operating system abend.

REMOTE_SYSTEM
Optional parameter

The name of the remote system if the abend was raised in the client transaction to reflect an abend occurring in the DPL server.

REQUEST_ID
Optional parameter

The request ID from the TCTTE for a terminal-oriented task.

SENSE_BYTES
Optional parameter

The SNA sense bytes if the abend was raised by DFHZAND.

SPACE
Optional parameter

An indication whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend.

Values for the parameter are as follows:
```
BASESPACE
NOSPACE
SUBSPACE
```

STATUS_FLAGS
Optional parameter

The status flags at the time of the abend.

STOKEN
Optional parameter

The subspace token (STOKEN) at the time of a program check or operating system abend.

STORAGE_TYPE
Optional parameter

A code that indicates the storage hit on an OC4 abend.

TEA
Optional parameter

The translation exception address at the time of a program check or operating system abend.

VECTOR_REGISTERS
Optional parameter

The contents of the vector registers at the time of a program check or operating system abend.

ABAB gate, START_ABEND function

The START_ABEND function of the ABAB gate is used to start transaction abend processing.

Input Parameters

ABEND_TOKEN
is the token allocated by ABAB for this abend (on a preceding CREATE_ABEND_RECORD request).

DUMP
Optional parameter

indicates whether a transaction dump should be produced for this abend.

Values for the parameter are:
```
NO
```

YES

IGNORE_HANDLES
Optional parameter

indicates whether this abend should be passed to any EXEC CICS HANDLE routines that are active. IGNORE_HANDLES(YES) results in EXEC CICS HANDLE being ignored at all levels of the program stack.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is EXCEPTION:
INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RETRY_ADDRESS
If an XPCTA exit requests retry, control returns to the point of invocation of start_abend, passing the retry address. This address includes the AMODE indicator in the first bit; it can be used as the target address in a DFHAM TYPE=BRANCH by the caller of START_ABEND GENERAL_REGISTERS is also set to point to the list of registers to be used for the retry, and SPACE to indicate the subspace. START_ABEND GENERAL64_REGISTERS and H64G_REGISTERS are also set to point to the list of registers to be used for the retry if this information is available.

GENERAL_REGISTERS
Optional parameter

The contents of the general purpose registers at the time of a program check or operating system abend.

GENERAL64_REGISTERS
Optional parameter

The contents of the 64-bit general purpose registers at the time of a program check or operating system abend.

H64G_REGISTERS
Optional parameter

The contents of the high order words of the 64-bit general purpose registers at the time of a program check or operating system abend.

SPACE
Optional parameter

An indication whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend.

Values for the parameter are:
BASESPACE
NOSPACE
SUBSPACE

ABAB gate, TAKE_TRANSACTION_DUMP function

The TAKE_TRANSACTION_DUMP function of the ABAB gate is used to take a transaction dump.

The TRANSACTION resource definition must specify dump and DUMP(YES) must be specified or defaulted on the associated START_ABEND call.

A transaction dump is not taken if any of the following is true:

- The application is going to handle the abend; that is, there is an active handle at this level and IGNORE_HANDLES(NO) is specified or defaulted on the associated START_ABEND call.
- The application is Language Environment/370 enabled, in which case the language interface deals with the abend.
- A transaction dump is currently in progress.

Input parameters

None

Output parameters

None

ABAB gate, UPDATE_ABEND_RECORD function

The UPDATE_ABEND_RECORD function of the ABAB gate is used to update an abend record (TACB).

Input Parameters

ABEND_TOKEN
is the token allocated by ABAB for this abend (on a preceding CREATE_ABEND_RECORD request).

ABEND_CODE
Optional parameter

The four-character transaction abend code.

ACCESS_REGISTERS
Optional parameter

The contents of the access registers at the time of a program check or operating system abend.

ALET
Optional parameter

The access list entry token (ALET) at the time of a program check or operating system abend.

ALL_FP_REGISTERS
Optional parameter

The contents of the floating point register values in the order 0 - 15 at the time of a program check or operating system abend.

BEAR
Optional parameter

The value of the breaking event address register (BEAR) at the time of a program check or operating system abend.

CURRENT_ACCESS_VALUES
Optional parameter

The current access register values are saved in the TACB.

CURRENT_FP_VALUES
Optional parameter

The current FP register values are saved in the TACB. If the task has not used the additional FP registers only the original FP registers are saved in the TACB. If any of the additional FP registers have been used by the task all the FP registers (0-15) and the FPC register are saved in the TACB.

CURRENT_VRR_VALUES
Optional parameter

The current vector register values are saved in the TACB.

ERROR_OFFSET
Optional parameter

The offset of a program check or operating system abend in the failing application program or CICS(R) AP domain program.

EXECUTION_KEY
Optional parameter

A code indicating the execution key at the time the abend was issued, or at the time the operating system abend or program check occurred.

FAILING_PROGRAM
Optional parameter

The name of the program in which the abend occurred.

FLOATING_POINT_REGISTERS
Optional parameter

The contents of the original floating point registers at the time of a program check or operating system abend.

FPC_REGISTER
Optional parameter

The value of the floating point control register at the time of a program check or operating system abend.

GENERAL_REGISTERS
Optional parameter

The contents of the general purpose registers at the time of a program check or operating system abend.

GENERAL64_REGISTERS
Optional parameter

The contents of the 64-bit general purpose registers at the time of a program check or operating system abend. This is an alternative parameter to GENERAL_REGISTERS. If GENERAL64_REGISTERS is specified, you cannot use the H64G_REGISTERS parameter.

GREG_ORDER
Optional parameter

A indication of the order of the registers passed in the GENERAL_REGISTERS GENERAL64_REGISTERS parameters. DFHSRP saves the registers in the abend record in the order 0-15, and INQUIRE_ABEND_RECORD will always return them in this order.

Values for the parameter are:

```
R0TOR15
R14TOR13
```

H64G_REGISTERS
Optional parameter

The contents of the high order words of the 64-bit general purpose registers at the time of a program check or operating system abend.
GENERAL64_REGISTERS cannot be used if H64G_REGISTERS is specified.

INTERRUPT_DATA
Optional parameter

The interrupt code and instruction length code etc, at the time of a program check or operating system abend.

PSW|PSW16
Optional parameter

The contents of the program status word (PSW) or the 16 byte PSW at the time of a program check or operating system abend.

REMOTE_SYSTEM
Optional parameter

The name of the remote system if the abend was raised in the client transaction to reflect an abend occurring in the DPL server.

REQUEST_ID
Optional parameter

The request ID from the TCTTE for a terminal-oriented task.

SPACE
Optional parameter

An indication whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend.

Values for the parameter are:

```
BASESPACE
NOSPACE
SUBSPACE
```

STATUS_FLAGS
Optional parameter

The status flags at the time of the abend.

STOKEN
Optional parameter

The subspace token (STOKEN) at the time of a program check or operating system abend.

STORAGE_TYPE
Optional parameter

A code indicating the storage hit on an OC4.

TEA
Optional parameter

The translation exception address at the time of a program check or operating system abend.

VECTOR_REGISTERS
Optional parameter

The contents of the vector registers at the time of a program check or operating system abend.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
```

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APAC gate, REPORT_CONDITION function

The REPORT_CONDITION function reports exception conditions that are encountered during transaction execution to the principal facility terminal, the CSMT destination, or both.

Input Parameters

CONDITION
Optional Parameter

The nature of the exception condition.

Values for the parameter are:

```
ROLLBACK
ROLLBACK_TERMINATE
ROLLBACK_NOT_SUPPORTED
LOCAL_NO_VOTE
REMOTE_NO_VOTE
REMOTE_NO_DECISION
INDOUBT_FAILURE
HEURISTIC_COMMIT
HEURISTIC_BACKOUT
COMMIT_FAILURE
BACKOUT_FAILURE
REMOTE_COMMIT_ABENDED
HEURISTIC_READONLY_COMMIT
HEURISTIC_READONLY_BACKOUT
LINKS_INVALID
```

CONTINUE
Optional Parameter

This parameter is not used.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
```

The following values are returned when RESPONSE is EXCEPTION:

```
TRANSACTION_ABEND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ABEND_CODE
Optional Parameter

The abend code issued for the condition specified.

APAP gate, TRANSFER_SIT function

The TRANSFER_SIT function of the APAP gate is used to transfer the address of DFHSIT to the AP domain after a GET_PARAMETERS call from this domain to the parameter manager domain.

Input Parameters

SIT

specifies the address and length of the system initialization table (DFHSIT).

Output Parameters

REASON

The values for the parameter are:

```
INCONSISTENT_RELEASE
INVALID_ADDRESS
INVALID_FUNCTION
INVALID_SIT_LENGTH
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APCR gate, ESTIMATE_ALL function

The ESTIMATE_ALL function of the APCR gate is used to estimate the size of terminal input/output area (TIOA) needed to ship a channel.

Input Parameters

CHANNEL_NAME

is the name of the channel.

CHANNEL_TOKEN

is a token referencing the channel.

COMMAND

is the type of API command that caused the channel to be shipped.

Values for the parameter are:

```
LINK
RETURN
START_ISC
START_MRO
```

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_ERROR
```

BYTES_NEEDED

is the total size, in bytes, of the exported channel, including channel and container headers and the overall length of the data in the containers. This total includes all bytes for all containers.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CHANNEL_TOKEN_OUT

Optional Parameter

contains, if CHANNEL_NAME was specified on input, a token referencing the channel.

APCR gate, ESTIMATE_CHANGED function

The ESTIMATE_CHANGED function of the APCR gate is used to obtain the size of the channel data structure that will be used to ship the containers that have been modified since the IMPORT_ALL call. Only new, modified, or deleted containers are shipped, with deleted containers being shipped as container headers only.

Input Parameters

CHANNEL_TOKEN
is a token referencing the channel.

COMMAND
is the type of API command that caused the channel to be shipped.

Values for the parameter are:
LINK
RETURN
START_ISC
START_MRO

CONTAINER_LIST
is a list of all the containers in the channel, obtained from an earlier IMPORT_ALL call.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CHANNEL_ERROR

BYTES_NEEDED
is the total size, in bytes, of the exported channel, including channel and container headers and the overall length of the data in the containers. This total includes all bytes for all containers.

NEW_CONTAINER_LIST
is a list of all the containers in the channel that have been created, modified, or deleted since the last IMPORT_ALL call. This list must be passed to a subsequent EXPORT_CHANGED call.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APCR gate, EXPORT_ALL function

The EXPORT_ALL function of the APCR gate is used to export the complete contents of a channel.

Input Parameters

CHANNEL_TOKEN
is a token referencing the channel.

COMMAND
is the type of API command that caused the channel to be shipped.

Values for the parameter are:
LINK
RETURN
SIBUS
START_ISC
START_MRO

CORRELATION_ID
Optional Parameter

If CORRELATION_ID is specified, the channel is exported from an AOR by request streams. (RZTA SEND_REPLY is used.)

TERMINAL_TOKEN
Optional Parameter

is a token referencing the terminal with which the channel is associated. If TERMINAL_TOKEN is specified, CICS terminal control is used to export the channel.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_ERROR
TERMINAL_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TC_ABEND
Optional Parameter

is the terminal control abend code.

TC_RESPONSE
Optional Parameter

is the terminal control response code.

TC_SENSE
Optional Parameter

is the terminal sense code.

APCR gate, EXPORT_CHANGED function

The EXPORT_CHANGED function of the APCR gate is used to return only those parts of a channel that have changed since IMPORT_ALL was issued.

Input Parameters

CHANNEL_TOKEN
is a token referencing the channel.

COMMAND
is the type of API command that caused the channel to be shipped.

Values for the parameter are:

```
LINK
```

CONTAINER_LIST
is a list of all the containers in the channel, obtained from an earlier IMPORT_ALL call.

TERMINAL_TOKEN
is a token referencing the terminal with which the channel is associated. If TERMINAL_TOKEN is specified, CICS terminal control is used to export the channel.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_ERROR
DATA_ERROR
TERMINAL_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TC_ABEND
is the terminal control abend code.
TC_RESPONSE
is the terminal control response code.
TC_SENSE
is the terminal sense code.

APCR gate, IMPORT_ALL function

The IMPORT_ALL function of the APCR gate is used to import the complete contents of a channel.

Input Parameters

COMMAND
is the type of API command that caused the channel to be shipped.

Values for the parameter are:
```
LINK
RETURN
SIBUS
START_ISC
START_MRO
```
CHANNEL_TOKEN_IN
Optional Parameter

is a token referencing an existing channel into which the channel data is to be imported.
DATA_START
Optional Parameter

is the position of the beginning of the channel data in the inbound TIOA.
RS_TOKEN
Optional Parameter

is a token referencing the request stream with which the channel is associated. If RS_TOKEN is specified, the channel is exported from a listener region by request streams. (RZSO SEND_REQUEST is used).
TERMINAL_TOKEN
Optional Parameter

is a token referencing the terminal with which the channel is associated. If TERMINAL_TOKEN is specified, CICS terminal control is used to export the channel.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
DATA_ERROR
TERMINAL_ERROR
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
CHANNEL_NAME
Optional Parameter

is the name of the channel that has been created.
CHANNEL_TOKEN
Optional Parameter

is a token referencing the channel that has been created.

CONTAINER_LIST
Optional Parameter

is the address of a control block that identifies the initial state of the channel. It can be passed to a subsequent EXPORT_CHANGED call, when it is used to identify what changes have been made by comparing the initial state of the channel to the current state. This allows CICS to re-export only the changed containers.

CORRELATION_ID
Optional Parameter

DATA_END
Optional Parameter

SIZE
Optional Parameter

TC_ABEND
Optional Parameter

is the terminal control abend code.

TC_RESPONSE
Optional Parameter

is the terminal control response code.

TC_SENSE
Optional Parameter

is the terminal sense code.

APCR gate, IMPORT_CHANGED function

The IMPORT_CHANGED function of the APCR gate is used to import those parts of a channel that have been modified since an EXPORT_ALL call. Any modified containers are either replaced or deleted. New containers are added. Unchanged containers are not received on the connection.

Input Parameters

CHANNEL_TOKEN
is a token referencing the channel.

COMMAND
is the type of API command that caused the channel to be shipped.

Values for the parameter are:
`LINK`

DATA_START
is the position of the beginning of the channel data in the inbound TIOA.

TERMINAL_TOKEN
is a token referencing the terminal with which the channel is associated. If TERMINAL_TOKEN is specified, CICS terminal control is used to export the channel.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`CHANNEL_ERROR`
`DATA_ERROR`
`TERMINAL_ERROR`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TC_ABEND
is the terminal control abend code.

TC_RESPONSE
is the terminal control response code.
TC_SENSE
is the terminal sense code.
DATA_END
Optional Parameter
SIZE
Optional Parameter

APEX gate, INVOKE_USER_EXIT function

The INVOKE_USER_EXIT function of the APEX gate is used to invoke the user exit at a specified exit point.

Input Parameters

EXIT_POINT
is the name of the exit.
TRACE
indicates whether or not user exits are to be traced.

Values for the parameter are:
```
NO
YES
```
EXIT_PARAMETER_*n*
Optional Parameter

is the parameter (number *n*) required by the exit. The nature of the parameter varies from one exit to another.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
```
The following values are returned when RESPONSE is EXCEPTION:
```
CHANGE_MODE_FAILURE
EXIT_PROGRAM_FAILURE
```
The following values are returned when RESPONSE is INVALID:
```
INVALID_EXIT_POINT
INVALID_FUNCTION
```
EXIT_RETURN_CODE
is the return code, if any, issued by the exit.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APID gate, PROFILE function

The PROFILE function of the APID gate extracts information from the AP domain profile for timeout.

Input Parameters

NAME
Optional Parameter

is the name of the profile

Output Parameters

REASON

The values for the parameter are:

```
NOT_FOUND
TM_LOCATE_FAILED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RTIMEOUT

Optional Parameter

is the read timeout value.

APID gate, QUERY_NETNAME function

The PROFILE function of the APID gate extracts information from the AP domain profile for timeout.

Input Parameters

SYSID

is the name of the sysid

Output Parameters

REASON

The values for the parameter are:

```
NOT_FOUND
TM_LOCATE_FAILED
```

NETNAME

is the value of the netname for the given sysid.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APIQ gate, INQ_APPLICATION_DATA function

The INQ_APPLICATION_DATA function of the APIQ gate is used to inquire about application data owned by the application domain.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INQ_FAILED
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
DPL_PROGRAM
NO_TRANSACTION_ENVIRONMENT
TRANSACTION_DOMAIN_ERROR
USXM_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACEE

Optional Parameter

is the address of the access control environment element (ACEE)

DSA

Optional Parameter

is the address of the head of the chain of dynamic storage for reentrant programs.

EIB
Optional Parameter

is the address of the EXEC Interface Block.

RSA
Optional Parameter

is the address of the apllication's register save area.

SYSEIB
Optional Parameter

is the address of the System EXEC Interface Block.

TCTUA
Optional Parameter

is the address of the Task Control Table User Area.

TCTUASIZE
Optional Parameter

is the length (in bytes) of the Task Control Table User Area.

TWA
Optional Parameter

is the address of the Task Work Area.

TWASIZE
Optional Parameter

is the length (in bytes) of the Task Work Area.

APIQ gate, INQ_SIT_PARM function

Return the value of a system initialization parameter.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

INFOCENTER
Optional Parameter

The value of the INFOCENTER system initialization parameter.

APJC gate, WRITE_JOURNAL_DATA function

The WRITE_JOURNAL_DATA function of the APJC gate is used to write a single record into a named journal.

Input Parameters

FROM
is the address of the record.

JOURNAL_RECORD_ID
is the system type record identifier.

JOURNALNAME
is the journal identifier name.

WAIT
specifies whether or not CICS is to wait until the record is written to auxiliary storage before returning control to the exit program.

Values for the parameter are:

NO

YES

RECORD_PREFIX

Optional Parameter

is the journal record user prefix.

Output Parameters

REASON

The values for the parameter are:

INVALID_FORMAT

INVALID_FUNCTION

IO_ERROR

JOURNAL_NOT_FOUND

JOURNAL_NOT_OPEN

LENGTH_ERROR

STATUS_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APLI gate, ESTABLISH_LANGUAGE function

The ESTABLISH_LANGUAGE function of the APLI gate is used to establish the language of a conventional compiled program.

Input Parameters

DATA_LOCATION

defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line).

Values for the parameter are:

ANY

BELOW

DEFINED_LANGUAGE

is the language defined for the program.

Values for the parameter are:

ASSEMBLER

COBOL

C370

LE370

NOT_DEFINED

PLI

ENTRY_POINT

is the entry point address of the program.

EXECUTION_KEY

is a code indicating the execution key at the time the abend was issued, or at the time the operating system abend or program check occurred.

Values for the parameter are:

CICS

USER

LANGUAGE_BLOCK

is a token identifying the current language block for the program.

LOAD_POINT

is the load point address of the program.

PROGRAM
is the 8-character name of the program whose language is to be determined

PROGRAM_LENGTH
is the length of the program.

REQUEST_TYPE
identifies the call of establish language. If the caller has a request type of link and establish language fails, then abend. Do not abend for a request type of load.

Values for the parameter are:
LINK
LOAD

THREADSAFE
indicates whether whether the program is quasi-reentrant (and must execute on the QR TCB) or threadsafe (and can execute on the QR TCB or an OPEN TCB).

Values for the parameter are:
NO
OPENAPI
YES

JVM_CLASS_PTR
Optional Parameter

is a token addressing the JVM class name length and value.

JVM_DEBUG
Optional Parameter

An enumerated type indicating whether JVM debug is to be used

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
TRANSACTION_ABEND

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

CICSVAR_THREADSAFE
is the threadsafe value established for the program.

Values for the parameter are:
CICSVAR_NO
CICSVAR_OPENAPI
CICSVAR_YES
NOT_DEFINED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ABEND_CODE
Optional Parameter

is the four-character transaction abend code.

LANGUAGE_ESTABLISHED
Optional Parameter

is the language established for the program.

Values for the parameter are:
```
ASSEMBLER
ASSEMBLER_CICS
COBOL
COBOL2
C370
JVM
LE370
MVSLE370
NOT_DEFINED
PLI
```
NEW_BLOCK
Optional Parameter

is a new token identifying the new language block for the program.

RUNTIME_ENVIRONMENT
Optional Parameter

is the runtime environment established for the program.

Values for the parameter are:
```
JVM_RUNTIME
LE370_RUNTIME
NON_LE370_RUNTIME
XPLINK_RUNTIME
```
APLI gate, START_PROGRAM function
The START_PROGRAM function of the APLI gate is used to start a program.

Input Parameters
CEDF_STATUS
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF).

Values for the parameter are:
```
CEDF
NOCEDF
```
COMMAREA
is an optional token identifying the communications area for the program.

EXECUTION_SET
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program).

Values for the parameter are:
```
DPLSUBSET
FULLAPI
```
LANGUAGE_BLOCK
is a token identifying the current language block for the program.

LINK_LEVEL
is the 16-bit value indicating the link-level of the program.

PROGRAM
is the 8-character name of the program whose language is to be determined

DEFERRED_ABEND_FOR_XCTL
Optional Parameter

indicates whether a Runaway type abend should be started on completion of the current START_PROGRAM.

Values for the parameter are:
NO
YES

ENVIRONMENT_TYPE
Optional Parameter

is the environment type of the program.

Values for the parameter are:
EXEC
GLUE
PLT
SYSTEM
TRUE
URM

JVM_PROG
Optional Parameter

indicates whether the request is for establish language for a JVM program.

Values for the parameter are:
NO
YES

PARMLIST_PTR
Optional Parameter

is an optional token identifying the parameter list for the program.

SYNCONRETURN
Optional Parameter

defines whether or not a syncpoint is to be taken on return from the linked program.

Values for the parameter are:
NO
YES

SYSEIB_REQUEST
Optional Parameter

indicates whether or not an EXEC CICS LINK or EXEC CICS XCTL had the SYSEIB translator option specified.

Values for the parameter are:
NO
YES

TRANSACTION_BUILD
Optional Parameter

Values for the parameter are:
BUILD
TERMINATE

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

AUTOSTART_DISABLED
JVM_PROFILE_NOT_FOUND
JVM_PROFILE_NOT_VALID
JVMPOOL_DISABLED
SYSTEM_PROPERTIES_NOT_FND
TRANSACTION_ABEND
USER_CLASS_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ABEND_CODE
Optional Parameter

is the four-character transaction abend code.

IGNORE_PENDING_XCTL
Optional Parameter

indicates whether or not a pending XCTL should be ignored by program manager.

Values for the parameter are:
NO
YES

APLJ gate, PIPI_CALL_SUB function

Provides an interface to the Language Environment preinitialization programming interface (PIPI) call_sub function.

Input Parameters

EXECUTION_KEY
The execution key used when a program runs in this PIPI environment.

Values for the parameter are:
CICS
USER

PIPI_CALL_PARAMETERS
The address of the parameter list to be passed to the called program.

PIPI_TABLE_INDEX
The row number in the PIPI table of the program to be called.

PIPI_TOKEN
A token returned by Language Environment's init_sub_dp function. The token identifies the PIPI environment, and is used on the PIPI call_sub and term functions.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
TRANSACTION_ABEND

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPI_RETURN_CODE
Optional Parameter

The return code from the Language Environment function.

PIPI_SUB_FEEDBACK
Optional Parameter

The Language Environment feedback code

PIPI_SUB_RETURN_CODE
Optional Parameter

The Language Environment subroutine return code

APLI gate, PIPI_INIT_SUB_DP function

Provides an interface to the Language Environment preinitialization programming interface (PIPI) init_sub_dp function.

Input Parameters

EXECUTION_KEY
The execution key used when a program runs in this PIPI environment.

Values for the parameter are:

```
CICS
USER
```

PIPI_RUNTIME_OPTIONS
Address of the Language Environment runtime options to be used for the pre-initialized environment.

PIPI_SERVICE_RTNS
Address of the vector of service routines which CICS provides for the PIPI environment (LOAD, DELETE, GETSTORE, FREESTORE).

PIPI_TABLE_ADDRESS
Address of the PIPI table of routines to be executed in the PIPI environment.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
TRANSACTION_ABEND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

PIPI_TOKEN
A token returned by Language Environment's init_sub_dp function. The token identifies the PIPI environment, and is used on the PIPI call_sub and term functions.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPI_RETURN_CODE
Optional Parameter

The return code from the Language Environment function.

APLI gate, PIPI_TERM function

Provides an interface to the Language Environment preinitialization programming interface (PIPI) term function.

Input Parameters

EXECUTION_KEY

The execution key used when a program runs in this PIPI environment.

Values for the parameter are:

```
CICS
USER
```

PIPI_TOKEN

A token returned by Language Environment's init_sub_dp function. The token identifies the PIPI environment, and is used on the PIPI call_sub and term functions.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
TRANSACTION_ABEND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPI_RETURN_CODE

Optional Parameter

The return code from the Language Environment function.

APLX gate, NOTIFY_REFRESH function

Notify AP domain that a program has been replaced by a new copy. AP domain cleans us some of its resources.

Input Parameters

PROGRAM

The 8-character name of the program that has been refreshed.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
TRANSACTION_ABEND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ABEND_CODE

Optional Parameter

The four-character abend code which is to be issued by CICS when an exception response is given and the cause of the exception is a transaction abend.

APRA gate, RELAY_TERMINAL_REQUEST function

The RELAY_TERMINAL_REQUEST function of the APRA gate relays an API request, which has a surrogate TCTTE in use as the principal facility, to the routing region.

Input Parameters

MESSAGE_DATA
Contains the inbound message.

Output Parameters

SURROGATE
A token containing a pointer to the surrogate TCTTE.

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRA gate, REMOTE_ATTACH function

The REMOTE_ATTACH function of the APRA gate attaches a transaction for a transaction routing session in the application region.

Input Parameters

MESSAGE_DATA
Contains the inbound message.

Output Parameters

SURROGATE
A token containing a pointer to the surrogate TCTTE.

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRA gate, REMOTE_DETACH function

The REMOTE_DETACH function of the APRA gate detaches a transaction for a transaction routing session in the application region.

Input Parameters

SURROGATE
A token containing a pointer to the surrogate TCTTE.

Output Parameters

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRD gate, END_ATOMS function

Commit outstanding atoms of recovery.

Input Parameters

DIRECTION

Indicates whether the atoms of recovery are committed or backed out.

Values for the parameter are:

BACKWARD
FORWARD

LOG

A bjnary value that indicates whether changes are to be logged.

Values for the parameter are:

NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

DISASTER_PERCOLATION

The following values are returned when RESPONSE is EXCEPTION:

PERCOLATE_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RESULT

The result of the commit request.

Values for the parameter are:

NO
READ_ONLY
YES

APRD gate, INITIALISE function

Perform the second stage of initialization of resource definition recovery.

Input Parameters

START

The type of CICS startup.

Values for the parameter are:

COLD
EMER
WARM

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

DISASTER_PERCOLATION

The following values are returned when RESPONSE is EXCEPTION:

PERCOLATE_ERROR
RECOVER_FAILED

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRD gate, PRE_INITIALISE function

Perform the first stage of initialization of resource definition recovery.

- Build the resource definition anchor block (RDAB)
- Load TBSS and TONR
- Initialize the suspend tokens
- Tell RM about the APRD recovery gate address

Input Parameters

STORE_TOKEN
A token that identifies the storage subpool in which the anchor block is created.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
`DISASTER_PERCOLATION`

The following values are returned when RESPONSE is EXCEPTION:
`PERCOLATE_ERROR`

The following values are returned when RESPONSE is INVALID:
`INVALID_FUNCTION`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRL gate, CREATE_BUNDLE_BY_ID_VER function

Create a unique BUNDLE resource name by using the bundle ID and version information.

Input parameters

ATTRIBUTE_STRING
The block for attribute data.

BUNDLE_ID
The bundle ID that is defined in the bundle manifest.

BUNDLE_MAJOR_VER
The major version of the bundle that is defined in the bundle manifest.

BUNDLE_MICRO_VER
The micro version of the bundle that is defined in the bundle manifest.

BUNDLE_MINOR_VER
The minor version of the bundle that is defined in the bundle manifest.

MGMTPART
The name of the management part that is associated with the application for this bundle.

USERID
The user ID that requested the bundle creation.

Output parameters

BUNDLE_NAME
A dynamically generated name for the BUNDLE resource.

REASON
The following values are returned when RESPONSE is EXCEPTION:

INSTALL_FAILED

NOT_AUTHORISED

TERMINATING

VALIDATION_FAILED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT

INVALID_FUNCTION

INVALID_LENGTH

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRL gate, CREATE_RESOURCE function

The CREATE_RESOURCE function creates a resource when a BUNDLE resource is installed.

Input parameters

APPLICATION_ID
Optional parameter

The 64-character name of the application.

APPL_MAJOR_VER
Optional parameter

The fullword binary value that is the major version of the application.

APPL_MICRO_VER
Optional parameter

The fullword binary value that is the micro version of the application.

APPL_MINOR_VER
Optional parameter

The fullword binary value that is the minor version of the application.

ATTRIBUTE_STRING
The block for attribute data.

BUNDLE_TOKEN
The token of the BUNDLE resource that is creating the resource.

PLATFORM_ID
Optional parameter

The 64-character name of the platform.

RESOURCE_NAME
The 8-character name of the resource.

RESOURCE_SIGNATURE
The block for the resource signature.

RESOURCE_TOKEN
The token that represents the resource.

RESOURCE_TYPE
The 12-character resource type.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INSTALL_FAILED
TERMINATING
VALIDATION_FAILED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
INVALID_LENGTH
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRL gate, DISCARD_BUNDLE_BY_ID_VER function

Discard a BUNDLE resource using the bundle ID and version information.

Input parameters

BUNDLE_ID
The bundle ID that is defined in the bundle manifest.

BUNDLE_MAJOR_VER
The major version number of the CICS bundle.

BUNDLE_MICRO_VER
The micro version number of the CICS bundle.

BUNDLE_MINOR_VER
The minor version number of the CICS bundle.

MGMTPART
The name of the management part that is associated with the application for this bundle.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
BUNDLE_DISCARD_FAILED
NOT_FOUND
TERMINATING
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
INVALID_LENGTH
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRL gate, SET_BUNDLE_BY_ID_VER function

Modify the state of a BUNDLE resource using the bundle ID and version information.

Input parameters

BUNDLE_ID
The bundle ID that is defined in the bundle manifest.

BUNDLE_MAJOR_VER
The major version number of the CICS bundle.

BUNDLE_MICRO_VER
The micro version number of the CICS bundle.

BUNDLE_MINOR_VER
The minor version number of the CICS bundle.

MGMTPART
The name of the management part that is associated with the application for this bundle.

STATUS
Indicates whether the BUNDLE resource is to be discarded immediately.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
AVAILABLE_FAILED
AVAIL_NONE
BUNDLE_ENABLE_INCOMPLETE
BUNDLE_SET_FAILED
NOT_ENABLED
NOT_FOUND
NOT_UNAVAILABLE
TERMINATING
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
INVALID_IMMEDIATE_DISCARD
INVALID_LENGTH
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRR gate, IPIC_ROUTE_TRANSACTION function

The IPIC_ROUTE_TRANSACTION function of the APRR gate routes a transaction for a transaction routing session in the routing region.

Input Parameters

IPCONN
Is the name of the IPCONN resource.

TRANS_REMOTENAME
Is the REMOTENAME attribute of the TRANSACTION resource

Output Parameters

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRS gate, ACQUIRE_SURROGATE function

The ACQUIRE_SURROGATE function of the APRS gate acquires a surrogate TCTTE for a remote terminal definition.

Input Parameters

OWNER_NETNAME
The NETNAME resource attribute of the terminal-owning region (TOR).

TERMID_IN_OWNER
The TERMID resource attribute of the terminal-owning region (TOR).

Output Parameters

SURROGATE
A token containing a pointer to the surrogate TCTTE.

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRS gate, RELEASE_SURROGATE function

The RELEASE_SURROGATE function of the APRS gate releases a surrogate TCTTE for a remote terminal definition.

Input Parameters

SURROGATE
A token containing a pointer to the surrogate TCTTE.

Output Parameters

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRT gate, ROUTE_TRANSACTION function

The ROUTE_TRANSACTION function of the APRT gate is used to dynamically route transactions (which are defined to be dynamic and not automatically initiated) based on decisions made by the dynamic transaction routing program. For transactions which are automatically initiated or are defined to be remote and not dynamic, DFHAPRT will statically route such transactions.

Input Parameters

DTRTRAN
indicates whether or not dynamic transaction routing is available.

Values for the parameter are:
NO

YES

DYNAMIC
indicates whether or not the transaction is defined as dynamic.

Values for the parameter are:
NO
YES

REMOTE
indicates whether or not the transaction is defined as remote.

Values for the parameter are:
NO
YES

REMOTE_NAME
is the four-character transaction identifier by which this transaction is to be known on the remote CICS region.

REMOTE_SYSTEM
is the name of the remote system if the abend was raised in the client transaction to reflect an abend occurring in the DPL server.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
ALL_SESSIONS_BUSY
DTRTRAN_REJECTED
ISC_DISABLED
NOTAUTH
PROGRAM_NOT_FOUND
REMOTE_CONN_OOS
REMOTE_CONN_OOS_SYS_CHGD
ROUTE_FAILED
TRANSACTION_ABEND

ABEND_CODE
is the four-character transaction abend code.

RAN_LOCALLY
indicates whether or not the transaction ran on the local CICS region (that is, was not routed to a remote CICS region).

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRX gate, FLATTEN_REQUEST function

The FLATTEN_REQUEST function of the APRX gate flattens a transaction routing request message that is transmitted from a routing region to an application region.

Input Parameters

XTSTG
Token containing a pointer to the transformer parameter list, DFHXTSTG.

FLAT_DATA
Buffer for flattened message data.

Output Parameters

REASON

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRX gate, FLATTEN_RESPONSE function

The FLATTEN_RESPONSE function of the APRX gate flattens a transaction routing response message that is transmitted from an application region to a routing region.

Input Parameters

XTSTG

Token containing a pointer to the transformer parameter list, DFHXTSTG.

FLAT_DATA

Buffer for flattened message data.

Output Parameters

REASON

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRX gate, UNFLATTEN_REQUEST function

The UNFLATTEN_REQUEST function of the APRX gate unflattens a transaction routing request message that is transmitted from a routing region to an application region.

Input Parameters

XTSTG

Token containing a pointer to the transformer parameter list, DFHXTSTG.

FLAT_DATA

Buffer for flattened message data.

Output Parameters

REASON

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APRX gate, UNFLATTEN_RESPONSE function

The UNFLATTEN_RESPONSE function of the APRX gate unflattens a transaction routing response message that is transmitted from an application region to a routing region.

Input Parameters

XTSTG

Token containing a pointer to the transformer parameter list, DFHXTSTG.

FLAT_DATA

Buffer for flattened message data.

Output Parameters

REASON

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTC gate, CANCEL function

The CANCEL function of the APTC gate invalidates the listening function.

Input Parameters

TOKEN
is the token for the session TCTTE

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
TC_ERROR
TOKEN_UNKNOWN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTC gate, CLOSE function

The CLOSE function of the APTC gate is used in clean up.

Input Parameters

TOKEN
is the token for the session TCTTE

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
TC_ERROR
TOKEN_UNKNOWN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTC gate, EXTRACT_PROCESS function

The EXTRACT_PROCESS function of the APTC gate extracts information for the request.

Input Parameters

TOKEN
Optional Parameter

is the token for the session TCTTE

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
TC_ERROR
TOKEN_UNKNOWN

CONVID
is the conversation id (which is the session tctte termid).

PIPDATA
Applicable only for LU6.2 conversations

PIPDATA_LEN
Applicable only for LU6.2 conversations

PROCESS_NAME
is the name of the process to be invoked

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SYNCLEVEL
is the synclevel of the conversation

APTC gate, LISTEN function

The LISTEN function of the APTC gate is used to update the TCTTE with the user token.

Input Parameters

TOKEN
The token for the session TCTTE

USER_TOKEN
The token supplied to the person who is to be notified.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
TC_ERROR
TOKEN_UNKNOWN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTC gate, OPEN function

The OPEN function of the APTC gate is used to allocate a session to the specified AOR.

Input Parameters

SYSID
is the name of the sysid

TRANID
is the transaction name to be attached in the AOR.

NETNAME
Optional Parameter

specifies the netname or applid of the AOR.

QUEUE
Optional Parameter

is the queue option specified by the routing program.

Values for the parameter are:
```
NO
YES
```

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
OPEN_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TOKEN

ERROR_CODE
Optional Parameter

The code passed back from the allocate procedure.

APTC gate, RECEIVE function

The RECEIVE function of the APTC gate is used to receive data.

Input Parameters

RECEIVE_BUFFER
is the buffer into which the reply is to be placed.

TOKEN
is the token for the session TCTTE

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NO_TCTTE
RECEIVE_BUFFER_TOO_SMALL
TC_ERROR
TOKEN_UNKNOWN
```

LAST
is an indicator to indicate if this is the last flow.

Values for the parameter are:

```
NO
YES
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTC gate, SEND function

The SEND function of the APTC gate is used to send the request to the AOR.

Input Parameters

LAST
is an indicator to indicate if this is the last flow.

Values for the parameter are:

```
NO
YES
```

SEND_BLOCK
is the block data with the length and send data pointer.

TOKEN
is the token for the session TCTTE

PREFIX_AREA
Optional Parameter

specifies the requeststreams information.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NO_TCTTE
TC_ERROR
TOKEN_UNKNOWN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTC gate, SET_SESSION function

The SET_SESSION function of the APTC gate is used to send the request to the AOR.

Input Parameters

RECOVERY_STATUS

indicates if recovery is necessary.

Values for the parameter are:

NECESSARY
UNNECESSARY

TOKEN

is the token for the session TCTTE

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

TC_ERROR
TOKEN_UNKNOWN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTD gate, DELETE_TRANSIENT_DATA function

The DELETE_TRANSIENT_DATA function of the APTD gate is used to delete the specified transient data queue.

Input Parameters

QUEUE

is the queue option specified by the routing program.

DISCARDING_DEFINITION

Optional Parameter

states whether this DELETEQ request is part of an attempt by Transient Data to discard a transient data queue definition.

Values for the parameter are:

NO
YES

RSL_CHECK

Optional Parameter

states whether resource-level checking is to be carried out.

Values for the parameter are:

NO
YES

Output Parameters

REASON

The values for the parameter are:

CSM_ERROR
DCT_ERROR
DIRECTORY_MGR_ERROR
INVALID_RSL_CHECK
IO_ERROR
JCP_ERROR
LOCKED

LOGIC_ERROR
NO_RECOVERY_TABLE
QUEUE_DISABLED
QUEUE_EXTRA
QUEUE_NOT_AUTH
QUEUE_NOT_FOUND
QUEUE_OMITTED
QUEUE_REMOTE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTD gate, INITIALISE_TRANSIENT_DATA function

The INITIALISE_TRANSIENT_DATA function of the APTD gate is invoked as part of the initialization process for the transient data facility.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
CSM_ERROR
DCT_ERROR
DIRECTORY_MGR_ERROR
JCP_ERROR
LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:

LOCKED
NO_RECOVERY_TABLE

The following values are returned when RESPONSE is EXCEPTION:

IO_ERROR
LENGTH_ERROR
NO_SPACE
QUEUE_BUSY
QUEUE_DISABLED
QUEUE_EMPTY
QUEUE_EXTRA
QUEUE_FULL
QUEUE_INDIRECT
QUEUE_INTRA
QUEUE_NOT_AUTH
QUEUE_NOT_FOUND
QUEUE_NOT_INPUT
QUEUE_NOT_OPEN
QUEUE_NOT_OUTPUT
QUEUE_REMOTE

The following values are returned when RESPONSE is INVALID:

FROM_LIST_OMITTED
INTO_OMITTED
INVALID_DATA_LOC
INVALID_FORMAT
INVALID_FROM_LIST_N
INVALID_FROM_LIST_P
INVALID_FROM_N
INVALID_FROM_P
INVALID_FUNCTION

INVALID_INTO_N
INVALID_INTO_P
INVALID_RSL_CHECK
INVALID_SUSPEND
QUEUE_OMITTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTD gate, READ_TRANSIENT_DATA function

The READ_TRANSIENT_DATA function of the APTD gate is used to read a single record from a named transient data queue.

Input Parameters

INTO
specifies a piece of storage into which the record is placed.

QUEUE
is the queue option specified by the routing program.

SUSPEND
specifies whether the caller wants to wait if the record to be read has not been committed to the queue yet.

Values for the parameter are:
NO
YES

DATA_KEY
Optional Parameter

if this is a READ TD SET rather than an INTO, DATA_KEY specifies whether Transient Data should obtain the required SET storage from CICS key or user key storage.

Values for the parameter are:
CICS
USER

DATA_LOC
Optional Parameter

if this is a READ TD SET rather than an INTO, DATA_LOC specifies whether Transient Data should obtain the required SET storage from above or below the 16MB line.

Values for the parameter are:
ANY
BELOW

RSL_CHECK
Optional Parameter

states whether resource-level checking is to be carried out.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
CSM_ERROR
DCT_ERROR
DIRECTORY_MGR_ERROR

INTO_OMITTED
INVALID_DATA_LOC
INVALID_INTO_N
INVALID_INTO_P
INVALID_RSL_CHECK
INVALID_SUSPEND
IO_ERROR
JCP_ERROR
LENGTH_ERROR
LOCKED
LOGIC_ERROR
NO_RECOVERY_TABLE
QUEUE_BUSY
QUEUE_DISABLED
QUEUE_EMPTY
QUEUE_NOT_AUTH
QUEUE_NOT_FOUND
QUEUE_NOT_INPUT
QUEUE_NOT_OPEN
QUEUE_OMITTED
QUEUE_REMOTE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTD gate, RESET_TRIGGER_LEVEL function

The RESET_TRIGGER_LEVEL function of the APTD gate is used to reset a transient data queue so that another trigger transaction can be attached. Sometimes it is necessary to include the RESET_TRIGGER_LEVEL function if a trigger transaction abends.

Input Parameters

QUEUE
is the queue option specified by the routing program.

Output Parameters

REASON
The values for the parameter are:
ABEND
QUEUE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APTD gate, WRITE_TRANSIENT_DATA function

The WRITE_TRANSIENT_DATA function of the APTD gate is used to write a single record (or multiple records) to a named transient data queue.

Input Parameters

FROM_LIST
is a list specifying the address and the length of each record that is to be written to the specified queue.

QUEUE
is the queue option specified by the routing program.

RSL_CHECK
Optional Parameter

states whether resource-level checking is to be carried out.

Values for the parameter are:

NO
YES

Output Parameters

REASON

The values for the parameter are:

CSM_ERROR
DCT_ERROR
DIRECTORY_MGR_ERROR
FROM_LIST_OMITTED
INVALID_FROM_LIST_N
INVALID_FROM_LIST_P
INVALID_FROM_N
INVALID_FROM_P
INVALID_RSL_CHECK
IO_ERROR
JCP_ERROR
LENGTH_ERROR
LOCKED
LOGIC_ERROR
NO_RECOVERY_TABLE
NO_SPACE
QUEUE_DISABLED
QUEUE_FULL
QUEUE_NOT_AUTH
QUEUE_NOT_FOUND
QUEUE_NOT_OPEN
QUEUE_NOT_OUTPUT
QUEUE_OMITTED
QUEUE_REMOTE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TD_MAX_LENGTH

Optional Parameter

indicates the maximum allowable length of a transient data record if a RESPONSE of EXCEPTION, and a REASON of LENGTH_ERROR is returned.

TD_MIN_LENGTH

Optional Parameter

indicates the minimum allowable length of a transient data record if a RESPONSE of EXCEPTION, and a REASON of LENGTH_ERROR is returned.

TD_RECORD

Optional Parameter

indicates the number of records that were successfully written to the transient data queue.

APXM gate, BIND_XM_CLIENT function

This function is called from the transaction manager domain during transaction initialization. The AP domain sets its recovery manager token to a non-zero value to ensure it will be invoked at syncpoint.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APXM gate, INIT_XM_CLIENT function

Called from the transaction manager domain during transaction initialization. The AP domain allocates the AP domain transaction lifetime control blocks, and anchors them with the AP domain's transaction token.

Input Parameters

LOCATE_PROFILE

Indicates whether the TCA should be initialized with values from the transaction's profile, if it exists.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON

The values for the parameter are:

```
GETMAIN_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APXM gate, RELEASE_XM_CLIENT function

Called from the transaction manager domain during transaction termination. AP domain transaction lifetime resources are released.

Output Parameters

REASON

The values for the parameter are:

```
FREEMAIN_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APXM gate, RMI_START_OF_TASK function

The RMI_START_OF_TASK function of the APXM gate is called from transaction manager domain to the AP Domain during transaction initialization. The AP domain invokes any task-related user exits enabled for start of task.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BRAT gate, ATTACH function

The ATTACH function of the BRAT gate is called to attach a transaction with a bridge primary client.

Input Parameters

FACILITYTOKEN

Facility token which references the BFB.

MESSAGE_TYPE
An indication that the bridge mechanism will use an architected message type. A CICS subroutine is used in place of the bridge exit.

Values for the parameter are:
BRIH

STATE_TOKEN
The message state token passed between the caller and the bridge subroutines responsible for the architected message.

TRANSACTION_ID
The 4 byte transaction id of the user transaction to be attached.

BRDATA
Optional Parameter

The address and length of a block of storage containing data to be passed to the bridge exit. This is used as part of the primary client data.

BREXIT
Optional Parameter

The name of the program to be used as the bridge exit. If this is not specified, DFHBRAT will get the default value from transaction manager. If there is no default bridge exit, an error is returned.

PRIORITY
Optional Parameter

Transaction manager priority of the transaction.

USERID
Optional Parameter

The USERID that should be signed-on to the terminal. This is only set when no facility token is passed.

Output Parameters

REASON
The values for the parameter are:
DISABLED
GETMAIN_FAILED
NO_BREXIT
NO_STORAGE
NO_XM_STORAGE
NOT_ENABLED_FOR_SHUTDOWN
NOT_FOUND
STATE_SYSTEM_ATTACH
USERID_NOT_AUTH_BREXIT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BRIQ gate, INQUIRE_CONTEXT function

The INQUIRE_CONTEXT of the BRIQ gate is called to inquire on bridge state data.

Input Parameters

TRANSACTION_TOKEN
Optional Parameter

The XM transaction token for the task to be inquired upon.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND

The following values are returned when RESPONSE is EXCEPTION:

BAD_TOKEN
NO_TRANSACTION_ENVIRONMENT

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BFB_TOKEN

Optional Parameter

The address of the BFB that was constructed or is to be re-used to satisfy this allocate.

BRDATA

Optional Parameter

Data passed to the bridge exit during attach.

BRIDGE_ENVIRONMENT

Optional Parameter

Indicates whether the task was started with a bridge facility.

Values for the parameter are:

NO
YES

BRIDGE_EXIT_PROGRAM

Optional Parameter

The name of the bridge exit program (if CONTEXT is BRIDGE or BREXIT).

BRIDGE_FORMATTER_PROGRAM

Optional Parameter

If CONTEXT(BREXIT) or CONTEXT(BRIDGE) is specified, the name of the bridge formatter user-replaceable program which is used to handle API commands emulated by the bridge.

BRIDGE_TRANSACTION_ID

Optional Parameter

The transaction that started the task running in a bridge environment.

CALL_EXIT_FOR_SYNC

Optional Parameter

Indicates if the bridge exit will be called for processing an explicit or implicit syncpoint

Values for the parameter are:

NO
YES

CONTEXT

Optional Parameter

The current program link level

Values for the parameter are:

BREXIT: a bridge exit or formatter is in control

BRIDGE: a task with a bridge exit is in control
NORMAL: the task is not running in a bridge environment.

FACILITYTOKEN
Optional Parameter

The 8 byte token used to represent the bridge session

IDENTIFIER
Optional Parameter

Data created by the bridge exit for problem determination purposes.

START_CODE
Optional Parameter

The emulated startcode of the user transaction

START_TYPE
Optional Parameter

Indicates how the task was started in the bridge environment.

Values for the parameter are:
LINK: the task was started using the Link3270 bridge.
START: the task was started using the START BREXIT mechanism.

CCNV gate, CONVERT_ADS function

Convert an application data structure (ADS) between a client and server code page.

Input Parameters

ADS_1
The application data structure to be converted.

RESOURCE_NAME
The name of the resource to be converted.

RESOURCE_TYPE
The type of resource to be converted.

Values for the parameter are:
FC
IC
PC
TD
TS

TARGET
The target code page for the data conversion.

Values for the parameter are:
ASCII
EBCDIC

ADS_2
Optional Parameter

A second application data structure to be converted, used only when RESOURCE_TYPE(FC) is specified.

BINARY_FORMAT
Optional Parameter

The binary format in which numeric data is represented.

Values for the parameter are:
BIG_ENDIAN
LITTLE_ENDIAN

CLIENT_CCSID
Optional Parameter

The Coded Character Set Identifier (CCSID) of the code page used by the client.

CLIENT_INDEX
Optional Parameter

Specifies the conversion table associated with the **CLIENT_CCSID** parameter.

CNV_ENTRY_TOKEN
Optional Parameter

A pointer to a DFHCNV TYPE=ENTRY record.

CNV_TABLE_TOKEN
Optional Parameter

The address at which DFHCNV is loaded.

SERVER_CCSID
Optional Parameter

The Coded Character Set Identifier (CCSID) of the code page used by the server.

SERVER_INDEX
Optional Parameter

Specifies the conversion table associated with the **SERVER_CCSID** parameter.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
```

SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:

BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCNV gate, CONVERT_DATA function

Convert a block of data between a client and server code page.

Input Parameters

SEGMENTED
A binary value that indicates whether the data to be converted is segmented or in a single buffer.

Values for the parameter are:

NO
YES

CONVERSION_TOKEN
Optional Parameter

A token that represents the server and client code page conversion tables.

SOURCE_BUFFER
Optional Parameter

A 31-bit pointer to the buffer that contains the data to be converted.

SOURCE_BUFFER64
Optional Parameter

A 64-bit pointer to the buffer that contains the data to be converted.

SOURCE_CCSID
Optional Parameter

The Coded Character Set Identifier (CCSID) of the code page used to encode the source data.

TARGET_BUFFER
Optional Parameter

A 31-bit pointer to the buffer that will contain the converted data.

TARGET_BUFFER64
Optional Parameter

A 64-bit pointer to the buffer that will contain the converted data.

TARGET_CCSID
Optional Parameter

The Coded Character Set Identifier (CCSID) of the code page used to encode the target data.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:

BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONVERSION_TOKEN_OUT
Optional Parameter

A token that represents the server and client code page conversion tables.

SUBSTITUTION
Optional Parameter

A binary value that indicates whether substitution characters were present in the input data.

Values for the parameter are:
NO
YES

CCNV gate, CREATE_CONVERSION_TOKEN function

Create a conversion token that represents the Coded Character Set Identifier (CCSID) of the source data and of the target data.

Input Parameters

SOURCE_CCSID
The CCSID of the source data.

TARGET_CCSID
The CCSID of the target data.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR

The following values are returned when RESPONSE is EXCEPTION:
ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED

TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:
BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

CONVERSION_TOKEN
A token that represents the CCSIDs of both source and target data.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCNV gate, EXTRACT_ADS function

Obtain an application data structure (ADS) for data conversion.

Input Parameters

ADS_1
The application data structure to be converted.

RESOURCE_NAME
The name of the resource to be converted.

RESOURCE_TYPE
The type of resource to be converted.

Values for the parameter are:
FC
IC
PC
TD
TS

TARGET
The target code page for the data conversion.

Values for the parameter are:
ASCII
EBCDIC

ADS_2
Optional Parameter

A second application data structure to be converted, used only when RESOURCE_TYPE(FC) is specified.

BINARY_FORMAT
Optional Parameter

The binary format in which numeric data is represented.

Values for the parameter are:
BIG_ENDIAN
LITTLE_ENDIAN

CNV_ENTRY_TOKEN
Optional Parameter

A pointer to a DFHCNV TYPE=ENTRY record.

CNV_TABLE_TOKEN
Optional Parameter

The address at which DFHCNV is loaded.

SERVER_INDEX
Optional Parameter

Specifies the conversion table associated with the **SERVER_CCSID** parameter.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
```

RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CLIENT_CCSID
Optional Parameter

The Coded Character Set Identifier (CCSID) of the code page used by the client.

SERVER_CCSID
Optional Parameter

The Coded Character Set Identifier (CCSID) of the code page used by the server.

CCNV gate, FREE_CONVERSION_TOKEN function

Free a conversion token

Input Parameters

C32_TOKEN
The 3270 data conversion token to be freed.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID

SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:

BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCNV gate, GET_CONVERSION_TOKEN function

Retrieve a conversion token.

Input Parameters

C32_TOKEN

The 3270 data conversion token.

CGCSGID_CP

Optional Parameter

The server code page

CGCSGID_CS

Optional Parameter

The server character set.

CICS_CCSID

Optional Parameter

The CICS code page.

CLIENT_INDEX

Optional Parameter

The client conversion table to use.

IBM_CCSID

Optional Parameter

The IBM-assigned number of a Coded Character Set Identifier (CCSID).

SERVER_INDEX

Optional Parameter

The server conversion table to use.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE

LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:

BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCNV gate, INITIALISE function

Initialize code page conversion services.

Input parameters

None.

Output parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:

BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCNV gate, INQUIRE_CONVERSION_SIZE function

Determine the size of the buffer that is required to receive the output from a data conversion operation.

Input Parameters

SEGMENTED

A binary value that indicates whether the data to be converted is segmented or in a single buffer.

Values for the parameter are:

NO
YES

CONVERSION_TOKEN

Optional Parameter

A token that represents the server and client code page conversion tables.

SOURCE_BUFFER

Optional Parameter

A 31-bit pointer to the buffer that contains the data to be converted.

SOURCE_BUFFER64

Optional Parameter

A 64-bit pointer to the buffer that contains the data to be converted.

SOURCE_CCSID

Optional Parameter

The Coded Character Set Identifier (CCSID) of the code page used to encode the source data.

TARGET_CCSID

Optional Parameter

The Coded Character Set Identifier (CCSID) of the code page used to encode the target data.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN

SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:

BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SIZE
The size of the buffer that is required to receive the output from a data conversion operation.

CONVERSION_TOKEN_OUT
Optional Parameter

A token that represents the server and client code page conversion tables.

CCNV gate, VERIFY_CGCSGID function

Verify that server code page and character set identifiers are valid.

Input Parameters

CGCSGID_CP
Optional Parameter

The server code page

CGCSGID_CS
Optional Parameter

The server character set.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:

BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CLIENT_INDEX
Optional Parameter

The client conversion table to use.

IBM_CCSID
Optional Parameter

The IBM-assigned number of a Coded Character Set Identifier (CCSID).

SERVER_INDEX
Optional Parameter

The server conversion table to use.

CCNV gate, VERIFY_CICS_CCSID function

Verify that a CICS Coded Character Set Identifier (CCSID) is valid.

Input Parameters

CICS_CCSID

Optional Parameter

The CICS code page.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:

BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CLIENT_INDEX
Optional Parameter

The client conversion table to use.

IBM_CCSID
Optional Parameter

The IBM-assigned number of a Coded Character Set Identifier (CCSID).

SERVER_INDEX
Optional Parameter

The server conversion table to use.

CCNV gate, VERIFY_IANA_CCSID function

Verify that an IANA Coded Character Set Identifier (CCSID) is valid.

Input Parameters

IANA_CCSID
The IANA CCSID to be verified.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP
COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
```

ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:

BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CLIENT_INDEX
Optional Parameter

The client conversion table to use.

IBM_CCSID
Optional Parameter

The IBM-assigned number of a Coded Character Set Identifier (CCSID).

SERVER_INDEX
Optional Parameter

The server conversion table to use.

CCNV gate, VERIFY_IBM_CCSID function

Verify that an IBM Coded Character Set Identifier (CCSID) is valid.

Input Parameters

IBM_CCSID
Optional Parameter

The IBM-assigned number of a Coded Character Set Identifier (CCSID).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
KEDD_ERROR
LMLM_ERROR
LOCK_FAILURE
LOOP
MULTI_ERROR
SMAD_ERROR
SMGF_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ADS_1_OMITTED
ADS_2_NOT_SUPP
CGCSGID_NOT_SUPP
CICS_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_KNOWN
CLIENT_CCSID_NOT_SUPP

COMBINATION_UNSUPPORTED
CONVERSION_NOT_REQUIRED
CONVERSION_NOT_SUPP
IANA_CCSID_NOT_KNOWN
IANA_CCSID_NOT_SUPP
IBM_CCSID_NOT_KNOWN
INSUFFICIENT_STORAGE
INTERNAL_CONVERSION_ERROR
SERVER_CCSID_NOT_KNOWN
SERVER_CCSID_NOT_SUPP
SERVER_UNSUPPORTED
SERVICE_NOT_AVAILABLE
SOURCE_CCSID_INVALID
SOURCE_DATA_INCOMPLETE
TARGET_BUFFER_EXHAUSTED
TARGET_CCSID_INVALID
ZOS_CONVERSION_ERROR

The following values are returned when RESPONSE is INVALID:

BINARY_FORMAT_INVALID
CNV_ENTRY_TOKEN_INVALID
CNV_TABLE_NOT_LOADED
CNV_TABLE_NOT_VALID
CNV_TABLE_TOKEN_INVALID
INVALID_FORMAT
INVALID_FUNCTION
RESOURCE_TYPE_INVALID
CONV_TOKEN_OMITTED
SOURCE_CCSID_OMITTED
TARGET_CCSID_OMITTED
TARGET_INVALID

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CLIENT_INDEX

Optional Parameter

The client conversion table to use.

DBCS_CODE

Optional Parameter

A binary value indicating whether the CCSID represent a double byte character set.

Values for the parameter are:

NO
YES

SERVER_INDEX

Optional Parameter

The server conversion table to use.

CQCQ gate, CLOSE_MVS_CIB_QUEUE function

Close the MVS console interface block (CIB) queue.

Input Parameters

CLOSE

Specifies whether the queue should be closed immediately.

Values for the parameter are:
IMMEDIATE
NORMAL

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CQCQ gate, DEFER_CIB function

This function moves the first CICS console interface block (CIB) from the QR TCB processed_n CIB queue to the QR TCB deferred CIB queue.

The function is invoked if a definition for the console has to be autoinstalled and the definition for another console is currently being autoinstalled.

CICS CIBs on the QR TCB deferred CIB queue will be returned to the QR TCB processed_n CIB queue at a time of the caller's choosing.

Input Parameters

CIB_TOKEN
The address of the first CICS CIB on the QR TCB processed_n queue.
MVS_CIB
The address of the MVS CIB embedded in the first CICS CIB on the QR TCB processed_n queue.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CIB_TOKEN_INVALID
MVS_CIB_INVALID
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CQCQ gate, GET_CIB function

This function returns a pointer to the MVS console interface block (CIB) embedded in the first CICS CIB on the QR TCB processed_n CIB queue.

If the queue is empty then any CICS CIBs on the CQ TCB processed_n CIB queue are moved to the QR TCB processed_n CIB queue. If the queue is still empty then an exception response, either reason CIB_QUEUE_EMPTY or reason CIB_QUEUE_CLOSED is returned.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CIB_QUEUE_CLOSED
CIB_QUEUE_EMPTY
CIB_TOKEN
The address of the first CICS CIB on the QR TCB processed_n queue.
MVS_CIB
The address of the MVS CIB embedded in the first CICS CIB on the QR TCB processed_n queue.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CQCQ gate, GET_PROCESSED_CIB function

Return a pointer to the MVS console interface block (CIB) embedded in the first CICS CIB on the CQ TCB processed CIB queue.

If the queue is empty then any CICS CIBs on the QR TCB processed CIB queue are moved to the CQ TCB processed CIB queue.

If the queue is still empty then an exception response, either CIB_QUEUE_EMPTY or CIB_QUEUE_CLOSED, is returned.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CIB_QUEUE_EMPTY

MVS_CIB
The address of the MVS CIB embedded in the first CICS CIB on the QR TCB processed_n queue.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CQCQ gate, INITIALIZE function

This function initializes the CQ component.

Initialization consists of the following steps:

- Allocate storage for the anchor block for the CQ component
- Set the address of the anchor block in the CSA optional features list
- Allocate storage for 254 CICS console interface blocks (CIBs); MVS supports a maximum of 255 concurrent CIBS, however one CIB is effectively reserved for CEKL
- Attach the CQ TCB
- Attach the CQ system task, progam DFHCQSY

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
SMAD_ERROR
SMGF_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CQCQ gate, MERGE_CIB_QUEUES function

Concatenates the QR TCB deferred console interface block (CIB) queue and the QR TCB processed_n CIB queue to form an updated QR TCB processed_n CIB queue.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CQCQ gate, PUT_CIB function

Removes the first CICS console interface block (CIB) from the CQ TCB free CIB queue, create the CICS CIB from the MVS CIB, and add the CICS CIB to the head of the CQ TCB processed_n queue.

Input Parameters

MVS_CIB
The address of the MVS CIB embedded in the first CICS CIB on the QR TCB processed_n queue.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`CICS_BUSY`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CQCQ gate, PUT_PROCESSED_CIB function

Move the first CICS console interface block (CIB) from the QR TCB processed_n CIB queue to the QR TCB processed_y CIB queue.

Input Parameters

CIB_TOKEN
The address of the first CICS CIB on the QR TCB processed_n queue.

MVS_CIB
The address of the MVS CIB embedded in the first CICS CIB on the QR TCB processed_n queue.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`CIB_TOKEN_INVALID`
`MVS_CIB_INVALID`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CQCQ gate, TRACE_PUT_CQ function

Makes an entry in the console queue trace table. Console queue trace entries are fixed length. Each trace entry can contain up to 128 bytes, the current limit, of data.

Input Parameters

MVS_CIB
The address of the MVS CIB embedded in the first CICS CIB on the QR TCB processed_n queue.

POINT_ID
The trace point identifier.

DATA1
Optional Parameter

The data to be traced.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, END_BROWSE_CAPTURESPEC function

END_BROWSE_CAPTURESPEC ends a browse of capture specifications.

Input Parameters

CS_BROWSE_TOKEN

The token that identifies the browse operation.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, END_BROWSE_DATA_PRED function

END_BROWSE_DATA_PRED ends the browse object.

Input Parameters

BROWSE_TOKEN

The token that identifies the current browse object.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, END_BROWSE_EVENTBINDING function

END_BROWSE_EVENTBINDING of the ECIS gate ends a browse of event bindings.

Input Parameters

EB_BROWSE_TOKEN

The token that identifies the browse operation.

Output Parameters

REASON

The following value is returned when RESPONSE is DISASTER:

```
UNKNOWN_DIRECTORY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, END_BROWSE_INFO_SOURCE function

END_BROWSE_INFO_SOURCE ends the browse object.

Input Parameters

BROWSE_TOKEN

The token that identifies the current browse object.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, END_BROWSE_OPTION_PRED function

END_BROWSE_OPTION_PRED ends the browse object.

Input Parameters

BROWSE_TOKEN

The token that identifies the current browse object.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, GET_NEXT_CAPTURESPEC function

GET_NEXT_CAPTURESPEC returns information about the next capture specification in the browse.

Input Parameters

CS_BROWSE_TOKEN

The token that identifies the current browse operation.

Output Parameters

CS_NAME

The name of the capture specification.

CAPTURE_PTYPE

Optional parameter

The capture point type.

Possible values of this parameter are:

```
PRECOMMAND
POSTCOMMAND
PROGRAMINIT
```

CAPTURE_POINT

Optional parameter

The verb or adverb associated with this command or blank.

CURRENT_PROGRAM
Optional parameter

The name of the program that is in the Current program application context predicate.

CURRENT_PROGRAM_OP
Optional parameter

The operator that is in the Current program application context predicate.

Possible values of this parameter are:
- ALLVALUES
- DOESNOTEQUAL
- DOESNOTSTART
- EQUALS
- GREATERTHAN
- ISNOTGREATER
- ISNOTLESS
- LESSTHAN
- STARTSWITH

CURRENT_TRANID
Optional parameter

The name of the transaction that is in the Transaction ID application context predicate.

CURRENT_TRANID_OP
Optional parameter

The operator that is in the Transacation ID application context predicate.

Possible values of this parameter are:
- ALLVALUES
- DOESNOTEQUAL
- DOESNOTSTART
- EQUALS
- GREATERTHAN
- ISNOTGREATER
- ISNOTLESS
- LESSTHAN
- STARTSWITH

CURRENT_USERID
Optional parameter

The name of the user that is in the User ID application context predicate.

CURRENT_USERID_OP
Optional parameter

The operator that is in the User ID application context predicate.

Possible values of this parameter are:
- ALLVALUES
- DOESNOTEQUAL
- DOESNOTSTART
- EQUALS
- GREATERTHAN
- ISNOTGREATER
- ISNOTLESS
- LESSTHAN
- STARTSWITH

EVENT_NAME
Optional parameter

The name of the event binding.

NUM_APPL_DATA_PRED
Optional parameter

The value of the application data predicate.

NUM_INFO_SOURCE
Optional parameter

The value of the information source.

NUM_OPTION_PRED
Optional parameter

The value of the application command option predicate.

PRIMARY_PREDICATE
Optional parameter

The value of the primary predicate.

PRIMARY_PREDICATE_OP
Optional parameter

The operator that is in the primary predicate.

Possible values of this parameter are:

```
ALLVALUES
DOESNOTEQUAL
DOESNOTSTART
EQUALS
GREATERTHAN
ISNOTGREATER
ISNOTLESS
LESSTHAN
STARTSWITH
```

PRIMARY_PREDICATE_TYPE
Optional parameter

The type of the primary predicate.

Possible values of this parameter are:

```
CONTAINER
CURRENTPGM
EVENT
FILE
MAP
PROGRAM
SERVICE
TDQUEUE
TRANSACTION
TSQUEUE
```

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
BROWSE_END_EARLY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, GET_NEXT_DATA_PRED function

GET_NEXT_DATA_PRED returns information about the next data predicate in the browse.

Input Parameters

BROWSE_TOKEN
The token that identifies the current browse object.

Output Parameters

CONTAINERNAME
Optional parameter

The name of the container when LOCATION is CHANNEL or FROMCHANNEL; otherwise, this field contains all blanks.

FIELDLENGTH
Optional parameter

The length, in bytes, of the data to be tested by this predicate.

FIELDOFFSET
Optional parameter

The offset into the source of the data to be tested by this predicate.

FILENAME
Optional parameter

The first 32 characters of the name of the variable in the imported language structure that is used to define this predicate.

FILTERVALUE
Optional parameter

The value of the application data predicate.

LOCATION
Optional parameter

The name of the location of the data to be tested.

STRUCTNAME
Optional parameter

The first 32 characters of the name of the imported structure that is used to define this predicate.

VARIABLENAME
Optional parameter

The first 32 characters of the name of the variable in the imported structure that is used to define this predicate.

PREDICATE_OPERATOR
Optional parameter

The operator that is used with the value in the FILTERVALUE option when the predicate is being evaluated.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
BROWSE_END_EARLY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
```

INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, GET_NEXT_EVENTBINDING function

GET_NEXT_EVENTBINDING returns information about the next event binding in the browse.

Input Parameters

EB_BROWSE_TOKEN
The token that identifies the current browse object.

Output Parameters

EB_NAME
The name of the event binding.

EB_STATUS
Optional parameter

The status of the event binding.

The values of this parameter are:
DISABLED
ENABLED

EB_USERTAG
Optional parameter

The current usertag of the event binding.

REASON
The following value is returned when RESPONSE is DISASTER:
UNKNOWN_DIRECTORY

The following value is returned when RESPONSE is EXCEPTION:
BROWSE_END

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, GET_NEXT_INFO_SOURCE function

GET_NEXT_INFO_SOURCE returns information about the information source in the browse.

Input Parameters

BROWSE_TOKEN
The token that identifies the current browse object.

Output Parameters

CONTAINERNAME
Optional parameter

The name of the container when LOCATION is CHANNEL or FROMCHANNEL; otherwise, this field contains all blanks.

FIELDLENGTH
Optional parameter

The length, in bytes, of the data specified in the LOCATION field.

FIELDOFFSET
Optional parameter

The offset into the source of the data to be captured by this capture specification.

FILENAME
Optional parameter

The first 32 characters of the name of the file that contains the imported language structure that is used to define this predicate.

ITEMNAME
Optional parameter

The name that is specified for this item of capture data.

LOCATION
Optional parameter

The name of the data source that contains the data to be captured.

STRUCTNAME
Optional parameter

The first 32 characters of the name of the imported structure that is used to define this information source.

VARIABLENAME
Optional parameter

The first 32 characters of the name of the variable in the imported structure that is used to define this information source.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
BROWSE_END_EARLY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, GET_NEXT_OPTION_PRED function

GET_NEXT_OPTION_PRED returns information about the next application command option predicate in the browse.

Input Parameters

BROWSE_TOKEN
The token that identifies the current browse object.

Output Parameters

FILTERVALUE
The mask value for this predicate.

OPTIONNAME
Optional parameter

The name of the application command option.

PREDICATE_OPERATOR
Optional parameter

The operator that is used with the value in the OPT_FILTERVALUE option when the predicate is being evaluated.

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
BROWSE_END_EARLY

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, INQ_CAPTURESPEC function

INQ_CAPTURESPEC retrieves information about a specified capture specification.

Input Parameters

CS_NAME
The name of the capture specification.

EB_NAME
The name of the event binding to be browsed for the associated capture specifications.

Output Parameters

CAPTURE_PTYPE
Optional parameter

The capture point type.

The values of this parameter are:
PRECOMMAND
POSTCOMMAND
PROGRAMINIT

CAPTURE_POINT
Optional parameter

The verb or adverb associated with this command or blank.

CURRENT_PROGRAM
Optional parameter

The name of the program in the current program application context predicate.

CURRENT_PROGRAM_OP
Optional parameter

The operator in the current program application context predicate.

The possible values of this parameter are:
ALLVALUES
DOESNOTEQUAL
DOESNOTSTART
EQUALS
GREATERTHAN
ISNOTGREATER
ISNOTLESS
LESSTHAN
STARTSWITH

CURRENT_TRANSID
Optional parameter

The name of the transaction in the Transaction ID application context predicate.

CURRENT_TRANSID_OP
Optional parameter

The operator in the Transaction ID application context predicate.

The possible values of this parameter are:
ALLVALUES
DOESNOTEQUAL
DOESNOTSTART
EQUALS
GREATERTHAN
ISNOTGREATER
ISNOTLESS
LESSTHAN
STARTSWITH

CURRENT_USERID
Optional parameter

The name of the user in the User ID application context predicate.

CURRENT_USERID_OP
Optional parameter

The operator in the User ID application context predicate.

The possible values of this parameter are:
ALLVALUES
DOESNOTEQUAL
DOESNOTSTART
EQUALS
GREATERTHAN
ISNOTGREATER
ISNOTLESS
LESSTHAN
STARTSWITH

EVENT_NAME
Optional parameter

The name of the event binding.

NUM_APPL_DATA_PRED
Optional parameter

The number of application data predicates.

NUM_INFO_SOURCE
Optional parameter

The number of information sources.

NUM_OPTION_PRED
Optional parameter

The number of application command option predicates.

PRIMARY_PREDICATE
Optional parameter

The value of the primary predicate.

PRIMARY_PREDICATE_OP
Optional parameter

The operator in the primary predicate.

The possible values of this parameter are:
ALLVALUES
DOESNOTEQUAL

DOESNOTSTART
EQUALS
GREATERTHAN
ISNOTGREATER
ISNOTLESS
LESSTHAN
STARTSWITH

PRIMARY_PREDICATE_TYPE
Optional parameter

The type of the primary predicate.

The possible values of this parameter are:
CONTAINER
CURRENTPGM
EVENT
FILE
MAP
PROGRAM
SERVICE
TDQUEUE
TRANSACTION
TDQUEUE

REASON
The following values are returned when RESPONSE is EXCEPTION:
CS_NOT_FOUND
EB_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, INQ_EVENTBINDING function

INQ_EVENTBINDING retrieves information about a specified event binding.

Input Parameters

EB_NAME
The name of the event binding.

Output Parameters

EB_STATUS
Optional parameter

The status of the event binding.

The values of this parameter are:
ENABLED
DISABLED

EB_USERTAG
Optional parameter

The user tag of the event binding.

EPADAPTER_NAME
Optional parameter

The name of the EP adapter for the event binding.

REASON
The following value is returned when RESPONSE is EXCEPTION:
NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, INQ_EVENTPROCESS function

INQ_EVENTPROCESS retrieves the status of event processing.

Output Parameters

EP_STATUS
The current status of event processing.

The values of this parameter are:
DRAINING
STARTED
STOPPED

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, SET_EVENTPROCESS function

SET_EVENTPROCESS sets the status of event processing.

Input Parameters

EP_STATUS
The new status of event processing.

The values of this parameter are:
DRAIN
DRAINEND
START
STOP

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
ALREADY_DRAINING

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, SET_EVENTBINDING function

SET_EVENTBINDING sets the status of the specified event binding.

Input Parameters

EB_NAME
The name of the event binding.

EB_STATUS
The new status of the event binding.

The values of this parameter are:
ENABLED
DISABLED

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, START_BROWSE_CAPTURESPEC function

START_BROWSE_CAPTURESPEC starts a browse of capture specifications.

Input Parameters

EB_NAME
The name of the event binding to be browsed for the associated capture specifications.

Output Parameters

CS_BROWSE_TOKEN
The token that identifies the browse operation.

REASON
The following value is returned when RESPONSE is EXCEPTION:
EB_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, START_BROWSE_DATA_PRED function

START_BROWSE_DATA_PRED starts a browse of the application data predicates for a capture resource.

Input Parameters

CS_NAME
The name of the CAPTURESPEC resource to be browsed for the associated application data predicates.

EB_NAME
The name of the event binding to be browsed for the associated capture specifications.

Output Parameters

BROWSE_TOKEN
The token that identifies the browse operation that is starting.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CS_NOT_FOUND
EB_NOT_FOUND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, START_BROWSE_EVENTBINDING function

The START_BROWSE function of ECIS gate starts a browse of event bindings.

Input Parameters

Output Parameters

EB_BROWSE_TOKEN
The token that identifies the browse operation.

REASON
The following value is returned when RESPONSE is DISASTER:

```
UNKNOWN_DIRECTORY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, START_BROWSE_INFO_SOURCE function

START_BROWSE_INFO_SOURCE starts a browse of the information sources for a specified CAPTURESPEC resource.

Input Parameters

CS_NAME
The name of the CAPTURESPEC resource to be browsed for the associated option predicates.

EB_NAME
The name of the event binding that is associated with the specified capture specification.

Output Parameters

BROWSE_TOKEN
The token that identifies the browse operation.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CS_NOT_FOUND
```

EB_NOT_FOUND

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECIS gate, START_BROWSE_OPTION_PRED function

START_BROWSE_OPTION_PRED starts a browse of the option predicates, such as the application command option predicates, for a capture resource.

Input Parameters

CS_NAME
The name of the CAPTURESPEC resource to be browsed for the associated option predicates.

EB_NAME
The name of the event binding that is associated with the specified capture specifications.

Output Parameters

BROWSE_TOKEN
The token that identifies the browse operation.

REASON
The following values are returned when RESPONSE is EXCEPTION:

CS_NOT_FOUND
EB_NOT_FOUND

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECSC gate, GET_EVENT_POINT_TOKEN function

GET_EVENT_POINT_TOKEN gets a token for a given capture point. This token can be used in the DFHECTST macro to check that event processing is active at the given capture point. The token can also give direct access to the capture point when the system event functions for event filter and capture are starting.

Input Parameters

CAPTURE_POINT
The event point at which a token can be generated. A token can be generated for the following events:

DB2_CONNECTION_STATUS
FILE_ENABLE_STATUS
FILE_OPEN_STATUS
TASK_THRESHOLD
TRANCLASS_TASK_THRESHOLD
TRANSACTION_ABEND

Output Parameters

EVENT_POINT_TOKEN
The returned event point token.

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECSC gate, PROCESS_DEFERRED function

PROCESS_DEFERRED starts the main loop that processes the contents of the deferred filtering queue.

Output Parameters

EVENT_POINT_TOKEN
The returned event point token.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECSC gate, SYSTEM_EVENT_GENERIC function

SYSTEM_EVENT_GENERIC returns information about a generic event compared to installed event capture specifications. This function can also capture and queue events for emission.

Input Parameters

ABSTIME
Optional parameter

The current absolute time and date.
ACTION
The name of the event that happened.
APPLID
Optional parameter

The APPLID of the system.
CAPTURE_POINT
The type of generic event that can be captured:
TRANSACTION_ABEND
MESSAGE
DATA1
Optional parameter

The addition piece of captured data identified by the NAME1 parameter.
DATA2
Optional parameter

The addition piece of captured data identified by the NAME2 parameter.
DATA3
Optional parameter

The addition piece of captured data identified by the NAME3 parameter.
DATA4
Optional parameter

The addition piece of captured data identified by the NAME4 parameter.
DATA5
Optional parameter

The addition piece of captured data identified by the NAME5 parameter.

DATA_LIST
Optional parameter

A list of additional pieces of filter or capture data identified by associated entry in NAME_LIST parameter.

EVENT_POINT_TOKEN
A token representing the capture point. This token is obtained from the GET_EVENT_POINT_TOKEN function .

NAME1
Optional parameter

A name by which an additional piece of capture data can be identified.

NAME2
Optional parameter

A name by which an additional piece of capture data can be identified.

NAME3
Optional parameter

A name by which an additional piece of capture data can be identified.

NAME4
Optional parameter

A name by which an additional piece of capture data can be identified.

NAME5
Optional parameter

A name by which an additional piece of capture data can be identified.

NAME_LIST
Optional parameter

A list of names by which the additional piece of filter or capture data specified by DATA_LIST parameter can be identified.

NETQUAL
Optional parameter

The network qualifier ID of the system.

TARGET
The name of the object to which the event happened.

TRANSID
Optional parameter

The current transaction ID.

USERID
Optional parameter

The current user ID.

Output Parameters

EVENTS
The number of events captured.

REASON
The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECSC gate, SYSTEM_EVENT_STATUS function

SYSTEM_EVENT_STATUS returns information about the resource state changes compared to installed event capture specifications. This function can also capture and queue events for emission.

Input Parameters

ABSTIME
Optional parameter

The current absolute time and date.

APPLID
Optional parameter

The APPLID of the system.

CAPTURE_POINT
The type of resource state change that can be captured:

```
DB2_CONNECTION_STATUS
FILE_ENABLE_STATUS
FILE_OPEN_STATUS
```

CONN_STATUS
Optional parameter

The current connection status of the resource, if applicable.

DATA1
Optional parameter

The addition piece of captured data identified by the NAME1 parameter.

DATA2
Optional parameter

The addition piece of captured data identified by the NAME2 parameter.

DATA3
Optional parameter

The addition piece of captured data identified by the NAME3 parameter.

DATA4
Optional parameter

The addition piece of captured data identified by the NAME4 parameter.

DATA5
Optional parameter

The addition piece of captured data identified by the NAME5 parameter.

ENABLE_STATUS
Optional parameter

The current enable status of the resource, if applicable.

EVENT_POINT_TOKEN
A token representing the capture point. This token is obtained from the GET_EVENT_POINT_TOKEN function.

NAME1
Optional parameter

A name by which an additional piece of capture data can be identified.

NAME2
Optional parameter

A name by which an additional piece of capture data can be identified.

NAME3
Optional parameter

A name by which an additional piece of capture data can be identified.

NAME4
Optional parameter

A name by which an additional piece of capture data can be identified.

NAME5
Optional parameter

A name by which an additional piece of capture data can be identified.

NETQUAL
Optional parameter

The network qualifier ID of the system.

OPEN_STATUS
Optional parameter

The current open status of the resource, if applicable.

RESOURCE_NAME
The name of the resource that has changed state.

STATUS_FROM
The state the resource has changed from.

STATUS_TO
The state the resource has changed to.

TRANSID
Optional parameter

The current transaction ID.

USERID
Optional parameter

The current user ID.

Output Parameters

EVENTS
The number of events captured.

REASON
The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECSC gate, SYSTEM_EVENT_THRESHOLD function

SYSTEM_EVENT_THRESHOLD returns information about the scalar value changes compared to installed event capture specifications. This function can also capture and queue events for emission.

Input Parameters

ABSTIME
Optional parameter

The current absolute time and date.

APPLID
Optional parameter

The APPLID of the system.

CAPTURE_POINT
The type of scalar value change that can be captured:
```
TASK_THRESHOLD
TRANCLASS_TASK_THRESHOLD
```

DEFERRED_FILTER_ID
Optional parameter

An ID assigned when the initial filter request is deferred. The same ID is specified when the request is reissued by the deferred filtering task CEPF.

EVENT_POINT_TOKEN
A token representing the capture point. This token is obtained from the GET_EVENT_POINT_TOKEN function.

MAX_VALUE
The maximum for the scalar value.

NETQUAL
Optional parameter

The network qualifier ID of the system.

SUSPEND
Defines whether the function can suspend. Values for the parameter are as follows:

```
YES
NO
```

The default value is `YES`.

THRESHOLD
The threshold value not to be crossed. This value can be specified in a threshold event filter predicate.

TRANSID
Optional parameter

The current transaction ID.

USERID
Optional parameter

The current user ID.

VALUE_FROM
The value the scalar value has changed from.

VALUE_TO
The value the scalar value has changed to.

Output Parameters

EVENTS
The number of events captured.

REASON
The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
INSUFFICIENT_STORAGE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ECSE gate, SIGNAL_EVENT function

SIGNAL_EVENT identifies a place in an application program where one or more events can be emitted.

Input Parameters

EVENT
The name of the event.

CHANNEL
Optional parameter

A channel name containing the source of the event data. It is optional and must not be used with the data parameter.

DATA
Optional parameter

An address and a length of the area containing the source of the event data. It is optional and must not be used with the channel parameter.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
EVENT_ERROR
CHANNEL_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCAT gate, INQ_BASEDSNAME function

This function is used only when the DSNB has not yet been validated.

Input Parameters

DSNAME
The 44-character name of the data set.

Output Parameters

BASEDSNAME
The 44–character name of the base data set.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
DATASET_NOT_KNOWN
DATASET_NOT_VSAM
SHOWCAT_ERROR
SHOWCAT_AIX_ERROR
ASSOC_NOT_FOUND
UNKNOWN_PATH_TYPE
LOCATE_ERROR
BASE_DATASET_NOT_KNOWN
DATASET_MIGRATED
```

The following value is returned when RESPONSE is DISASTER:

```
RECOVERY_ENTERED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCAT gate, INQ_CATALOG_QUIESCESTATE function

This function returns the quiesce state of the data set.

Input Parameters

DSNAME

The 44-character name of the data set.

Output Parameters

QUIESCESTATE

The quiesce state of the data set.

Values for the parameter are:

```
QUIESCED
UNQUIESCED
```

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
DATASET_NOT_KNOWN
BDAM_OR_PATH
IOERR
SYSTEM_BACK_LEVEL
```

The following value is returned when RESPONSE is DISASTER:

```
RECOVERY_ENTERED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCAT gate, INQ_CATALOG_RECOV_REQD function

This function inquires on the catalog recovery required flag.

Input Parameters

DSNAME

The 44-character name of the data set.

Output Parameters

RECOV_REQD

The state of the catalog recovery required flag.

Values for the parameter are:

```
YES
NO
```

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
DATASET_NOT_KNOWN
BDAM_OR_PATH
IOERR
SYSTEM_BACK_LEVEL
```

The following value is returned when RESPONSE is DISASTER:

```
RECOVERY_ENTERED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCAT gate, INQ_DATASET_STATE function

This function returns the state of the backup-while-open (BWO) bits for a named data set; the state is either fuzzy or sharp.

Input Parameters

DSNAME
The 44-character name of the data set.

Output Parameters

STATE
The state of the backup-while-open (BWO) bits for the data set.

Values for the parameter are:
FUZZY
SHARP

REASON
The following values are returned when RESPONSE is EXCEPTION:
FORWARD_RECOVERY_NEEDED
RESTORE_AND_FRECOV_NEEDED

The following values are returned when RESPONSE is DISASTER:
RECOVERY_ENTERED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCAT gate, SET_BWO_BITS_DISABLED function

This function sets the backup-while-open (BWO) bits to indicate that a data set is no longer eligible for fuzzy image copy.

Input Parameters

DSNAME
The 44-character name of the data set.

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
SYSTEM_BACK_LEVEL

The following values are returned when RESPONSE is DISASTER:

RECOVERY_ENTERED
SET_BWO_DISABLED_FAILED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCAT gate, SET_BWO_BITS_ENABLED function

This function sets the backout-while-open (BWO) bits to indicate that a data set is eligible for fuzzy image copy.

Input Parameters

DSNAME

The 44-character name of the data set.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

DATASET_NOT_KNOWN
FORWARD_RECOVERY_NEEDED
RESTORE_AND_FRECOV_NEEDED
SYSTEM_BACK_LEVEL
HSMDSS_BACK_LEVEL

The following value is returned when RESPONSE is DISASTER:

INQ_BWO_ENABLED_FAILED
RECOVERY_ENTERED
SET_BWO_ENABLED_FAILED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCAT gate, SET_CATALOG_RECOV_POINT function

This function updates the recovery point in the catalog for a named data set.

Input Parameters

DSNAME

The 44-character name of the data set.

RECOVERY_POINT

The 8-character recovery point.

Output Parameters

REASON

The following value is returned when RESPONSE is EXCEPTION:

```
SYSTEM_BACK_LEVEL
```

The following values are returned when RESPONSE is DISASTER:

```
RECOVERY_ENTERED
SET_CATALOG_RECOV_FAILED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCAT gate, SET_CATALOG_RECOV_REQD function

This function sets the recovery required flag in the catalog.

Input Parameters

DSNAME

The 44-character name of the data set.

RECOV_REQD

The catalog recovery required flag.

Values for the parameter are:

```
YES
NO
```

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
IOERR
SYSTEM_BACK_LEVEL
```

The following value is returned when RESPONSE is DISASTER:

```
RECOVERY_ENTERED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCAT gate, SET_CATALOG_RECOVERED function

This function sets the backup-while-open (BWO) bits of the catalog to a forward recovered state for a named data set.

Input Parameters

DSNAME

The 44-character name of the data set.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
DATASET_NOT_KNOWN
SYSTEM_BACK_LEVEL
```

The following values are returned when RESPONSE is DISASTER:

```
INQ_SMS_MANAGED_FAILED
RECOVERY_ENTERED
SET_CATALOG_RECOV_FAILED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCBU gate, TIDY_UP_FILE function

This function tidies up orphaned files during a CICS restart.

Input Parameters

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
INVALID_FUNCTION
INVALID_FORMAT
BROWSE_FAILED
RESOURCE_ERROR
```

No values are returned when RESPONSE is EXCEPTION.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCCA gate, CHECK function

This function returns the results of the previous operation.

Input Parameters

CHECK_TOKEN

The token that was returned on the previous request for which the results are being checked.

Output Parameters

ACCMETH_RETURN_CODE

A 2-byte code returned by SMSVSAM.

CONFLICTING_QUIESCE

Indicates the type of quiesce that conflicts with this request. Values for the parameter are:

QUIESCE
UNQUIESCE
NONBWO_END
BWO_END
NONBWO_START
BWO_START

REASON
The following values are returned when RESPONSE is EXCEPTION:
VSAM_REQUEST_ERROR
RLS_FAILURE

The following value is returned when RESPONSE is DISASTER:
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCA gate, COLD_START_RLS function

This function performs a cold start for the control access method control block (ACB).

This request is issued as part of CICS cold start processing. CICS issues an IDARECOV TYPE=COLDSTART call to SMSVSAM to release all record-level sharing (RLS) locks owned by this CICS and to clear the lost locks status and the non-RLS update-permitted state, for all data sets in this CICS region.

Input Parameters

SUBSYSNM
A pointer to an IFGSYSNM structure.

Output Parameters

ACCMETH_RETURN_CODE
A 2-byte code returned by SMSVSAM.

REASON
The following values are returned when RESPONSE is EXCEPTION:
VSAM_REQUEST_ERROR
RLS_FAILURE

The following value is returned when RESPONSE is DISASTER:
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR

PURGED

FCCA gate, DRAIN_CONTROL_ACB function

This function drains the control access method control block (ACB) when file control detects that an instance of the SMSVSAM server has failed.

DFHFCCA sets an indicator in file control static storage so that no other record-level sharing (RLS) activity can proceed and then DFHFCCA drains all existing RLS access. The server sequence number in file control static storage is incremented, all RLS ACBs are closed, and the control ACB is unregistered.

Input Parameters

None.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

DISASTER_PERCOLATION

ABEND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK

EXCEPTION

DISASTER

INVALID

KERNERROR

PURGED

FCCA gate, INQUIRE_RECOVERY function

This function inquires on the record-level sharing (RLS) recovery state; it is issued as part of CICS startup processing. CICS makes an IDAINQRC request to VSAM to obtain the information necessary to determine the RLS recovery actions that are required by CICS.

Input Parameters

AREA_PTR

A fullword pointer to the address of the area where the IFGINQRC information is to be returned.

AREA_LENGTH

A fullword binary field indicating the length of the supplied area.

Output Parameters

ACCMETH_RETURN_CODE

A 2-byte code returned by SMSVSAM.

REQUIRED_LENGTH

A fullword binary field containing the length of the IFGINQRC area to be returned, if its length exceeds the length of the supplied area.

REASON

The following values are returned when RESPONSE is EXCEPTION:

AREA_TOO_SMALL

VSAM_REQUEST_ERROR
RLS_FAILURE

The following value is returned when RESPONSE is DISASTER:
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCA gate, LOST_LOCKS_COMPLETE function

This function informs VSAM that lost locks (LL) recovery is complete.

CICS issues an IDARECOV TYPE=LL request to SMSVSAM when it has completed recovery processing for a data set that is in lost locks status. SMSVSAM resets the state of the data set in the sharing control data set to indicate that the data set is no longer in lost locks state with respect to this CICS.

Input Parameters

DATASET
The 44-character name of the base data set for which CICS has completed lost locks recovery.

RESTART
Optional Parameter

Indicates whether the call was issued by file control restart. Values for the parameter are:
YES
NO

Output Parameters

ACCMETH_RETURN_CODE
A 2-byte code returned by SMSVSAM.

REASON
The following values are returned when RESPONSE is EXCEPTION:
VSAM_REQUEST_ERROR
RLS_FAILURE

The following value is returned when RESPONSE is DISASTER:
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCA gate, QUIESCE_COMPLETE function

Quiesce processing is complete. When CICS has completed the processing required for a quiesce request from SMSVSAM, it issues an IDAQUIES call to SMSVSAM with a quiesce type of QUICMP.

Input Parameters

DATASET
The 44-character name of the base data set that has completed quiesce processing.

VSAM_QUIESCE_TOKEN
A token used to relate quiesce completion to the quiesce request that has been completed. This token is supplied by SMSVSAM when the quiesce request is received by CICS.

Output Parameters

ACCMETH_RETURN_CODE
A 2-byte code returned by SMSVSAM.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
VSAM_REQUEST_ERROR
RLS_FAILURE
```

The following value is returned when RESPONSE is DISASTER:

```
ABEND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCCA gate, QUIESCE_REQUEST function

This function issues a record-level sharing (RLS) quiesce request.

DFHFCCA issues quiesce requests to SMSVSAM on behalf of the quiesce component of CICS; it issues IDAQUIES calls of the following types:

- QUICLOSE to request SMSVSAM to notify all CICS systems that have ACBs open against this data set that these ACBs are to be closed. In addition, the data set is marked in the VSAM catalog as being quiesced after these ACBs have been closed.
- QUIOPEN to request SMSVSAM to mark the data set as no longer quiesced; that is, it is unquiesced. In addition, QUIOPEN will cancel a QUICLOSE that is in progress.
- QUIBEND to request SMSVSAM to cancel a BWO backup of a data set that is in progress.
- QUICEND to request SMSVSAM to cancel a non-BWO backup of a data set that is in progress.

Input Parameters

DATASET
The 44-character name of the base data set to be quiesced.

IMMEDIATE
Optional Parameter

This parameter applies only when the **QUIESCE_TYPE** parameter is set to QUIESCE. This parameter indicates whether the quiesce will force files to close immediately, or will allow inflight units of work to reach sync point. Values for the parameter are:

YES
NO

QUIESCE_TYPE
The type of quiesce. Values for the parameter are:

QUIESCE
UNQUIESCE
NONBWO_END
BWO_END

Output Parameters

ACCMETH_RETURN_CODE
A 2-byte code returned by SMSVSAM.

CHECK_TOKEN
A token that will be used on the CHECK request.

CONFLICTING_QUIESCE
Indicates the type of quiesce that conflicts with this request. Values for the parameter are:

QUIESCE
UNQUIESCE
NONBWO_END
BWO_END
NONBWO_START
BWO_START

REASON
The following values are returned when RESPONSE is EXCEPTION:

VSAM_REQUEST_ERROR
RLS_FAILURE

The following value is returned when RESPONSE is DISASTER:

ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCA gate, REGISTER_CONTROL_ACB function

This function registers the control access method control block (ACB). The control ACB is opened using an IDAREGP request to SMSVSAM. The control ACB must be registered before CICS can open any other ACBs for record-level sharing (RLS) access.

Input Parameters

None.

Output Parameters

VSAM_RETURN_CODE
A fullword return code from VSAM.

VSAM_REASON_CODE
A fullword 32-bit reason code from VSAM.

VSAM_ERROR_DATA
An 8-byte field containing error data returned by VSAM.

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
VSAM_REQUEST_ERROR
RLS_FAILURE
```

The following values are returned when RESPONSE is DISASTER:
```
DISASTER_PERCOLATION
ABEND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCCA gate, RELEASE_LOCKS function

This function releases all locks for the unit of work (UOW). CICS issues an IDALKREL request to SMSVSAM as part of commit processing at the end of every UOW. This request causes VSAM to release all locks owned by that UOW.

Input Parameters

LUWID
A fullword pointer to an IFGLUWID structure containing the ID for the unit of work.

RESTART
Optional Parameter

Indicates whether the call was issued by file control restart. Values for the parameter are:
```
YES
NO
```

Output Parameters

ACCMETH_RETURN_CODE
A 2-byte code returned by SMSVSAM.

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
VSAM_REQUEST_ERROR
RLS_FAILURE
```

The following value is returned when RESPONSE is DISASTER:
```
ABEND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCA gate, RESET_NONRLS_BATCH function

Resets the state of the data set in the sharing control data set to indicate that the batch override, or non-RLS update permitted, state no longer needs to be reported to CICS when it opens the data set.

Input Parameters

DATASET

The 44-character name of the base data set that is going to have its state cleared.

Output Parameters

ACCMETH_RETURN_CODE

A 2-byte code returned by SMSVSAM.

REASON

The following values are returned when RESPONSE is EXCEPTION:

VSAM_REQUEST_ERROR
RLS_FAILURE

The following value is returned when RESPONSE is DISASTER:

ABEND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCA gate, RETAIN_DATASET_LOCKS function

Retains all the locks for the data set in this unit of work (UOW).

CICS issues an IDARETLK TYPE=SS call to SMSVSAM when a UOW has suffered a backout failure on a data set. This call requests SMSVSAM to mark all locks against the data set owned by the UOW for conversion into retained locks on a subsequent IDALKREL call.

Input Parameters

LUWID

A fullword pointer to an IFGLUWID structure containing the ID for the unit of work.

DATASET

The 44-character name of the base data set that has had a backout failure.

Output Parameters

ACCMETH_RETURN_CODE

A 2-byte code returned by SMSVSAM.

REASON
The following values are returned when RESPONSE is EXCEPTION:
VSAM_REQUEST_ERROR
RLS_FAILURE

The following value is returned when RESPONSE is DISASTER:
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCA gate, RETAIN_UOW_LOCKS function

Retains all the locks in this unit of work (UOW).

CICS issues an IDARETLK TYPE=IND call to SMSVSAM when a UOW has encountered an indoubt failure. This call requests VSAM to mark all locks owned by the UOW for conversion into retained locks on a subsequent IDALKREL call.

Input Parameters

LUWID
A pointer to an IFGLUWID structure containing the ID for the unit of work.

Output Parameters

ACCMETH_RETURN_CODE
A 2-byte code returned by SMSVSAM.

REASON
The following values are returned when RESPONSE is EXCEPTION:
VSAM_REQUEST_ERROR
RLS_FAILURE

The following value is returned when RESPONSE is DISASTER:
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCA gate, UNREGISTER_CONTROL_ACB function

This function is used to unregister the control access method control block (ACB). The record-level sharing (RLS) control ACB is closed using an IDAUNRP request to SMSVSAM. The control ACB cannot be unregistered while any other ACBs are open for RLS access.

Input Parameters

None.

Output Parameters

VSAM_RETURN_CODE
A fullword return code from VSAM.

VSAM_REASON_CODE
A fullword reason code from VSAM.

REASON
The following values are returned when RESPONSE is EXCEPTION:
VSAM_REQUEST_ERROR
RLS_FAILURE

The following values are returned when RESPONSE is DISASTER:
DISASTER_PERCOLATION
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCI gate, INQUIRE function

FCCI is the parameter list used by file control to communicate with the coupling facility data table (CFDT) cross-memory server, DFHCFMN, for the table inquire function.

Input Parameters

BROWSE
Optional Parameter

This parameter specifies whether the inquire is for a single table or for the first or next table in a browse. If this parameter is omitted, a single table inquire is performed. The FIRST option indicates a search for a table greater than or equal to the specified name, and NEXT indicates a search for a table greater than the specified name.

Values for the parameter are:
FIRST
NEXT

TABLE NAME
16-character table name; this name is typically the CICS file name padded with trailing blanks.

TRANSACTION_NUMBER
Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

Output Parameters

ACCESS_MODE
Returned as EXCLUSIVE if the table is open for exclusive access; otherwise, SHARED.

This parameter can take the following values:
EXCLUSIVE
SHARED

AVAILABLE
Indicates whether new opens are currently allowed.

Values for the parameter are:

YES

NO

CURRENT_RECORDS
This fullword binary field indicates the number of records in the table the last time the current server accessed the table.

CURRENT_USERS
This fullword binary field indicates the number of user opens that are currently active against the table.

INITIAL_LOAD
Specifies whether initial load is required. If not, the first open creates an empty table.

Values for the parameter are:

YES

NO

KEY_LENGTH
This fullword binary field specifies the table key length in bytes, in the range 1 - 16.

LOADED
Indicates whether the table has been loaded. If the table was created as empty this is set to YES as if loading had already been done. If not, the value is set to YES using the SET function when loading is complete.

Values for the parameter are:

YES

NO

MAXIMUM_RECORDS
This fullword binary field specifies the maximum number of records that can be stored in the table. If no maximum limit is required, the maximum positive number (hex 7FFFFFFF) can be specified.

OPEN_MODE
Indicates whether the table is currently open and, if so, whether it is open for read-only or read/write access.

This parameter can take the following values:
NONE
READ_ONLY
READ_WRITE

REASON
The following values are returned when RESPONSE is EXCEPTION:

SERVER_CONNECTION_FAILED
TABLE_NOT_FOUND
CF_ACCESS_ERROR

RECORD_LENGTH
This fullword binary field specifies the table maximum record length, in the range 1 - 32767.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHARED ACCESS
If the table is currently open for exclusive access, this parameter indicates the level of shared access permitted by the exclusive user. If the table is not open for exclusive access, this parameter normally indicates that read and write sharing is allowed.

Values for the parameter are:
NONE
READ_ONLY
READ_WRITE

TABLE_NAME
The 16-character table name; this name is typically the CICS file name padded with trailing blanks.

UPDATE_MODEL
Specifies the method to be used for updating the table. This parameter takes one of the following values:
CONTENTION
Indicates that version compare and swap is used for updating the table.
LOCKING
Indicates that normal update locking is used for updating the table.
RECOVERABLE
Indicates that backout support is included with normal update locking.

FCCR gate, DELETE function

This function deletes a record from a coupling facility data table (CFDT) following a read for update.

Input Parameters

KEY
The 16-byte key of the record to be deleted.
KEY_COMPARISON
The comparison condition; this parameter can take the following values:
LT
LTEQ
EQ
GTEQ
GT
KEY_MATCH_LENGTH
The key match length for generic key operations.
SUSPEND
Specifies whether to wait if the requested record is locked by an active lock.
Values for the parameter are:
YES
NO

TABLE_NAME
This 16-character field contains the 8-character name of the CFDT and is padded with trailing blanks.

TABLE_TOKEN
The token returned by the OPEN function, which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
This 4-character string identifies the requesting task in the debug trace, if used.

UOW_ID
This 8-character string specifies the unit of work ID. The unit of work ID is required when updating using the locking model.

UPDATE_TOKEN
The token returned by the preceding read for update.

Output Parameters

KEY
The 16-byte key of the deleted record.

LOCK_OWNER_SYSTEM
This 8-character string identifies the MVS system from which the record lock was acquired for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
This 8-character string identifies the applid of the region that owns the record lock for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
This 8-character string identifies the unit of work that owns the record lock for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
RECORD_NOT_FOUND
RECORD_CHANGED
RECORD_BUSY
RECORD_LOCKED
TABLE_LOADING
INVALID_REQUEST
UPDATE_TOKEN_INVALID
INCOMPLETE_UPDATE
TABLE_TOKEN_INVALID
TABLE_DESTROYED
UOW_FAILED
UOW_NOT_IN_FLIGHT
UOW_TOO_LARGE
POOL_STATE_ERROR
CF_ACCESS_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
```

KERNERROR
PURGED

FCCR gate, DELETE_MULTIPLE function

This function deletes records from a coupling facility data table, subject to key match conditions, until no more records match or an exception occurs.

Input Parameters

KEY
The 16-byte key of the record to be deleted.

KEY_COMPARISON
The comparison condition; this parameter can take the following values:
LT
LTEQ
EQ
GTEQ
GT

KEY_MATCH_LENGTH
The key match length for generic key operations.

SUSPEND
Specifies whether to wait if the requested record is locked by an active lock. Values for the parameter are:
YES
NO

TABLE_NAME
This 16-character field contains the 8-character name of the CFDT and is padded with trailing blanks.

TABLE_TOKEN
The token returned by the OPEN function, which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
This 4-character string identifies the requesting task in the debug trace, if used.

UOW_ID
This 8-character string specifies the unit of work ID. The unit of work ID is required when updating using the locking model.

Output Parameters

DELETED_RECORD_COUNT
The number of records successfully deleted by the DELETE_MULTIPLE function.

KEY
The 16-byte key of the last record deleted.

LOCK_OWNER_APPLID
This 8-character string identifies the applid of the region that owns the record lock for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

LOCK_OWNER_SYSTEM
This 8-character string identifies the MVS system from which the record lock was acquired for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
This 8-character string identifies the unit of work that owns the record lock for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

REASON
The following values are returned when RESPONSE is EXCEPTION:

SERVER_CONNECTION_FAILED
RECORD_NOT_FOUND
RECORD_CHANGED
RECORD_BUSY
RECORD_LOCKED
TABLE_LOADING
INVALID_REQUEST
UPDATE_TOKEN_INVALID
INCOMPLETE_UPDATE
TABLE_TOKEN_INVALID
TABLE_DESTROYED
UOW_FAILED
UOW_NOT_IN_FLIGHT
UOW_TOO_LARGE
POOL_STATE_ERROR
CF_ACCESS_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCR gate, HIGHEST function

This function returns the highest key in a coupling facility data table (CFDT), if there is one.

Input Parameters

TABLE_NAME
This 16-character field contains the 8-character name of the CFDT and is padded with trailing blanks.

TABLE_TOKEN
The token returned by the OPEN function, which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
This 4-character string identifies the requesting task in the debug trace, if used.

Output Parameters

KEY
Returns the 16-byte key of the highest record.

REASON
The following values are returned when RESPONSE is EXCEPTION:
SERVER_CONNECTION_FAILED
RECORD_NOT_FOUND
TABLE_LOADING
TABLE_TOKEN_INVALID
TABLE_DESTROYED
POOL_STATE_ERROR
CF_ACCESS_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCR gate, LOAD function

This function adds a record to a coupling facility data table (CFDT) during loading.

Input Parameters

TABLE_NAME

This 16-character field contains the 8-character name of the CFDT and is padded with trailing blanks.

TABLE_TOKEN

The token returned by the OPEN function, which must be passed on all subsequent requests against that open table.

KEY

The 16-byte key of the record to be loaded.

DATA

The address and length of the record data to be loaded.

TRANSACTION_NUMBER

This 4-character string identifies the requesting task in the debug trace, if used.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

SERVER_CONNECTION_FAILED
DUPLICATE_RECORD
MAXIMUM_RECORDS_REACHED
NO_SPACE_IN_POOL
INVALID_REQUEST
INVALID_LENGTH
RECORD_NOT_FOUND
TABLE_LOADING
TABLE_TOKEN_INVALID
TABLE_DESTROYED
POOL_STATE_ERROR
CF_ACCESS_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCR gate, POINT function

This function locates a record in a coupling facility data table (CFDT).

Input Parameters

KEY
The 16-byte key of the record to be accessed. For approximate key operations, this parameter specifies the start key and is updated on successful completion to contain the key of the record accessed.

KEY_COMPARISON
The comparison condition; this parameter can take the following values:
LT
LTEQ
EQ
GTEQ
GT

KEY_MATCH_LENGTH
The key match length for generic key operations.

TABLE_NAME
This 16-character field contains the 8-character name of the CFDT and is padded with trailing blanks.

TABLE_TOKEN
The token returned by the OPEN function, which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
This 4-character string identifies the requesting task in the debug trace, if used.

UOW_ID
This 8-character string specifies the unit of work ID. The unit of work ID is required when updating using the locking model.

Output Parameters

KEY
Returns the 16-byte key of the located record.

REASON
The following values are returned when RESPONSE is EXCEPTION:
SERVER_CONNECTION_FAILED
RECORD_NOT_FOUND
TABLE_LOADING
TABLE_TOKEN_INVALID
TABLE_DESTROYED
POOL_STATE_ERROR
CF_ACCESS_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCR gate, READ function

This function reads a record in a coupling facility data table (CFDT) and, optionally, updates it.

Input Parameters

BUFFER
The input buffer for read requests.

KEY
The 16-byte key of the record to be accessed. For approximate key operations, this parameter specifies the start key and is updated on successful completion to contain the key of the record accessed.

KEY_COMPARISON
The comparison condition; this parameter can take the following values:
LT
LTEQ
EQ
GTEQ
GT

KEY_MATCH_LENGTH
The key match length for generic key operations.

SUSPEND
Specifies whether to wait if the requested record is locked by an active lock. Values for the parameter are:
YES
NO

TABLE_NAME
This 16-character field contains the 8-character name of the CFDT and is padded with trailing blanks.

TABLE_TOKEN
The token returned by the OPEN function, which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
This 4-character string identifies the requesting task in the debug trace, if used.

UOW_ID
This 8-character string specifies the unit of work ID. The unit of work ID is required when updating using the locking model.

Output Parameters

KEY
Returns the 16-byte key of the record.

LOCK_OWNER_APPLID
This 8-character string identifies the applid of the region that owns the record lock for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

LOCK_OWNER_SYSTEM
This 8-character string identifies the MVS system from which the record lock was acquired for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
This 8-character string identifies the unit of work that owns the record lock for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

REASON
The following values are returned when RESPONSE is EXCEPTION:
SERVER_CONNECTION_FAILED
RECORD_NOT_FOUND
RECORD_BUSY
RECORD_LOCKED
TABLE_LOADING
INVALID_REQUEST
INCOMPLETE_UPDATE
TABLE_TOKEN_INVALID
TABLE_DESTROYED

UOW_FAILED
UOW_NOT_IN_FLIGHT
UOW_TOO_LARGE
POOL_STATE_ERROR
CF_ACCESS_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

UPDATE_TOKEN
Returns a token on a read for update.

FCCR gate, READ_DELETE function

The READ_DELETE function reads and deletes a record from a coupling facility data table. It is not used by CICS.

FCCR gate, REWRITE function

This function rewrites an existing record in a coupling facility data table (CFDT), following a read for update.

Input Parameters

DATA
The address and length of the record data to be rewritten.

KEY
The 16-byte key of the record to be rewritten.

SUSPEND
Specifies whether to wait if the requested record is locked by an active lock.
Values for the parameter are:
YES
NO

TABLE_NAME
This 16-character field contains the 8-character name of the CFDT and is padded with trailing blanks.

TABLE_TOKEN
The token returned by the OPEN function, which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
This 4-character string identifies the requesting task in the debug trace, if used.

UOW_ID
This 8-character string specifies the unit of work ID. The unit of work ID is required when updating using the locking model.

UPDATE_TOKEN
The token returned by the preceding read for update.

Output Parameters

LOCK_OWNER_APPLID
This 8-character string identifies the applid of the region that owns the record lock for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

LOCK_OWNER_SYSTEM
This 8-character string identifies the MVS system from which the record lock was acquired for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
This 8-character string identifies the unit of work that owns the record lock for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
RECORD_NOT_FOUND
RECORD_CHANGED
RECORD_BUSY
RECORD_LOCKED
MAXIMUM_RECORDS_REACHED
NO_SPACE_IN_POOL
TABLE_LOADING
INVALID_REQUEST
INVALID_LENGTH
UPDATE_TOKEN_INVALID
INCOMPLETE_UPDATE
TABLE_TOKEN_INVALID
TABLE_DESTROYED
UOW_FAILED
UOW_NOT_IN_FLIGHT
UOW_TOO_LARGE
POOL_STATE_ERROR
CF_ACCESS_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCCR gate, UNLOCK function

This function unlocks a record previously read for update in a coupling facility data table (CFDT).

Input Parameters

BUFFER
The input buffer for read requests.

KEY
The 16-byte key of the record to be unlocked.

TABLE_NAME
This 16-character field contains the 8-character name of the CFDT and is padded with trailing blanks.

TABLE_TOKEN
The token returned by the OPEN function, which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
This 4-character string identifies the requesting task in the debug trace, if used.

UOW_ID
This 8-character string specifies the unit of work ID. The unit of work ID is required when updating using the locking model.

UPDATE_TOKEN
The token returned by the preceding read for update.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
RECORD_NOT_FOUND
RECORD_CHANGED
TABLE_LOADING
INVALID_REQUEST
UPDATE_TOKEN_INVALID
TABLE_TOKEN_INVALID
TABLE_DESTROYED
UOW_NOT_IN_FLIGHT
POOL_STATE_ERROR
CF_ACCESS_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCCR gate, WRITE function

This function writes a new record to a coupling facility data table (CFDT).

Input Parameters

DATA
The address and length of the record data to be added.

KEY
The 16-byte key of the record to be added.

SUSPEND
Specifies whether to wait if the requested record is locked by an active lock.
Values for the parameter are:

```
YES
NO
```

TABLE_NAME
This 16-character field contains the 8-character name of the CFDT and is padded with trailing blanks.

TABLE_TOKEN
The token returned by the OPEN function, which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
This 4-character string identifies the requesting task in the debug trace, if used.

UOW_ID
This 8-character string specifies the unit of work ID. The unit of work ID is required when updating using the locking model.

Output Parameters

LOCK_OWNER_APPLID
This 8-character string identifies the applid of the region that owns the record lock for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

LOCK_OWNER_SYSTEM
This 8-character string identifies the MVS system from which the record lock was acquired for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
This 8-character string identifies the unit of work that owns the record lock for a RECORD_BUSY or RECORD_LOCKED condition. This parameter is also set when the wait exit is taken for a lock wait.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
DUPLICATE_RECORD
RECORD_BUSY
RECORD_LOCKED
MAXIMUM_RECORDS_REACHED
NO_SPACE_IN_POOL
TABLE_LOADING
INVALID_REQUEST
INVALID_LENGTH
UPDATE_TOKEN_INVALID
INCOMPLETE_UPDATE
TABLE_TOKEN_INVALID
TABLE_DESTROYED
UOW_FAILED
UOW_NOT_IN_FLIGHT
UOW_TOO_LARGE
POOL_STATE_ERROR
CF_ACCESS_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCCT gate, CLOSE function

Ends the connection to the specified table.

Input Parameters

TABLE_NAME
16-character table name. This name is typically the CICS file name padded with trailing blanks.

TABLE_TOKEN
The token returned by the OPEN function, which must be passed on all subsequent requests against that table.

TRANSACTION_NUMBER
Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
TABLE_TOKEN_INVALID
TABLE_DESTROYED
POOL_STATE_ERROR
CF_ACCESS_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCCT gate, DELETE function

This function deletes a table if the table is not currently open. A security check for table access is performed.

Input Parameters

TABLE_NAME
16-character table name. This name is typically the CICS file name padded with trailing blanks.

TRANSACTION_NUMBER
Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
ACCESS_NOT_ALLOWED
TABLE_NOT_FOUND
EXCLUSIVE_ACCESS_CONFLICT
TABLE_DESTROYED
CF_ACCESS_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCT gate, EXTRACT_STATISTICS function

This function returns information about a table that is currently open, with the option to reset the statistics.

Input Parameters

RESET_STATISTICS
Optional Parameter

Specifies whether to reset the statistics. Values for the parameter are:
YES
NO

TABLE_NAME
16-character table name. This name is typically the CICS file name padded with trailing blanks.

TABLE_TOKEN
The token returned by the OPEN function, which must be passed on all subsequent requests against that table.

TRANSACTION_NUMBER
Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

Output Parameters

CONTENTION_COUNT
Optional Parameter

This fullword binary field indicates the number of times a rewrite or delete failed because of a mismatched version (for the contention model) or the number of times that a lock was found to be unavailable (for the locking or recoverable models) since the last statistics reset.

CURRENT_RECORDS
This fullword binary field indicates the number of records in the table the last time that the current server accessed the table.

CURRENT_USERS
This fullword binary field indicates the number of explicit opens that are currently active against the table, not including internal recoverable opens issued by the server.

HIGHEST_RECORDS
Optional Parameter

This fullword binary field indicates the highest number of records in the table as seen by the current server at any time since the last statistics reset.

MAXIMUM_RECORDS
Optional Parameter

This fullword binary field specifies the maximum number of records that can be stored in the table. If no maximum limit is required, the maximum positive number (hex 7FFFFFFF) can be specified.

REASON
The following values are returned when RESPONSE is EXCEPTION:
SERVER_CONNECTION_FAILED
TABLE_TOKEN_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCT gate, OPEN function

This function defines a table and establishes a connection to it. A security check is performed for access to the table name. If the table does not exist, it is implicitly created.

Input Parameters

TABLE_NAME
16-character table name. This name is typically the CICS file name padded with trailing blanks.

RECORD_LENGTH
This fullword binary field specifies the table maximum record length, in the range 1 - 32767.

KEY_LENGTH
This fullword binary field specifies the table key length in bytes, in the range 1 - 16.

MAXIMUM_RECORDS
Optional Parameter

This fullword binary field specifies the maximum number of records that can be stored in the table. If no maximum limit is required, the maximum positive number (hex 7FFFFFFF) can be specified.

UPDATE_MODEL
Specifies the method to be used for updating the table. Values for the parameter are:

CONTENTION
Indicates that version compare and swap is used for updating the table.

LOCKING
Indicates that normal update locking is used for updating the table.

RECOVERABLE
Indicates that backout support is included with normal update locking.

INITIAL_LOAD
Optional Parameter

Specifies whether initial load is required. If not, the first open creates an empty table. Values for the parameter are:
YES
NO

OPEN_MODE
Optional Parameter

Specifies the mode in which the file is opened. Values for the parameter are:

READ_ONLY
READ_WRITE

The default value for this parameter is READ_WRITE.

ACCESS_MODE

Optional Parameter

Specifies whether the table is being opened for exclusive or shared use. Values for the parameter are:

EXCLUSIVE
SHARED
PREFER_SHARED

Only one user at a time can have an exclusive open active. If the table requires loading and is not yet being loaded, it can be opened only in exclusive mode. The PREFER_SHARED option means that the table will be opened in exclusive mode if loading is required; otherwise, it will be opened in shared mode. The default value for this parameter is SHARED.

SHARED_ACCESS

Optional Parameter

Specifies for an exclusive mode open whether other users are allowed shared access to the file at the same time. Values for the parameter are:

NONE
READ_ONLY
READ_WRITE

The default value for this parameter is READ_WRITE.

TRANSACTION_NUMBER

Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

Output Parameters

ACCESS_MODE

Optional Parameter

Specifies whether the table is being opened for exclusive or shared use. Values for the parameter are:

EXCLUSIVE

SHARED

PREFER_SHARED

Only one user at a time can have an exclusive open active. If the table requires loading and is not yet being loaded, it can be opened only in exclusive mode. The PREFER_SHARED option means that the table will be opened in exclusive mode if loading is required; otherwise, it will be opened in shared mode. The default value for this parameter is SHARED.

CURRENT_RECORDS

This fullword binary field indicates the number of records in the table the last time that the current server accessed the table.

CURRENT_HIGH_KEY

Optional Parameter

This 16-character string indicates the key of the last record in the table at the time of the request.

CURRENT_USERS
This fullword binary field indicates the number of user opens that are currently active against the table.

INITIAL_LOAD
Specifies whether initial load is required. If not, the first open creates an empty table.

Values for the parameter are:

YES

NO

KEY_LENGTH
This fullword binary field specifies the table key length in bytes, in the range 1 - 16.

LOADED
Optional Parameter

Indicates whether the table has been loaded. If the table was created as empty, this parameter is set to YES as if loading had already taken place. If not, this parameter is set to YES using the SET function when loading is complete.

This parameter takes one of the following values:

YES

NO

MAXIMUM_RECORDS
Optional Parameter

This fullword binary field specifies the maximum number of records that can be stored in the table. If no maximum limit is required, the maximum positive number (hex 7FFFFFFF) can be specified.

REASON
The following values are returned when RESPONSE is EXCEPTION:

SERVER_CONNECTION_FAILED
TABLE_NOT_FOUND
CF_ACCESS_ERROR

RECORD_LENGTH
This fullword binary field specifies the table maximum record length, in the range 1 - 32767.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK

EXCEPTION

DISASTER

INVALID

KERNERROR

PURGED

TABLE_TOKEN
Token returned by the OPEN function which must be passed on all subsequent requests against that open table.

UPDATE_MODEL
Specifies the method to be used for updating the table. Values for the parameter are:
CONTENTION
Indicates that version compare and swap is used for updating the table.
LOCKING
Indicates that normal update locking is used for updating the table.
RECOVERABLE
Indicates that backout support is included with normal update locking.

FCCT gate, SET function

This function is used to change the attributes of a table. The maximum number of records can be changed, the open mode can be changed to indicate that loading is no longer taking place, and the access mode can be changed from exclusive to shared.

Input Parameters

ACCESS_MODE
Optional Parameter

Specifies whether the table is being opened for exclusive or shared use. Values for the parameter are:
EXCLUSIVE
SHARED
PREFER_SHARED

Only one user at a time can have an exclusive open active. If the table requires loading and is not yet being loaded, it can be opened only in exclusive mode. The PREFER_SHARED option means that the table will be opened in exclusive mode if loading is required; otherwise, it will be opened in shared mode. The default value for this parameter is SHARED.

AVAILABLE
Optional Parameter

Indicates whether new open requests are currently allowed for this table. Values for the parameter are:
YES
NO

LOADED
Optional Parameter

Indicates whether the table has been loaded. If the table was created as empty this parameter is set to YES as if loading had already taken place. Values for the parameter are:
YES
NO

MAXIMUM_RECORDS
Optional Parameter

This fullword binary field specifies the maximum number of records that can be stored in the table. If no maximum limit is required, the maximum positive number (hex 7FFFFFFF) can be specified.

SHARED_ACCESS
Optional Parameter

Specifies for an exclusive open mode whether other users are allowed shared access to the file at the same time. Values for the parameter are:
NONE
READ_ONLY

READ_WRITE

The default value for this parameter is READ_WRITE.

TABLE_NAME

16-character table name. This name is typically the CICS file name padded with trailing blanks.

TABLE_TOKEN

Optional Parameter

Token returned by the OPEN function, which must be passed on all subsequent requests against that open table. If the table is currently open, the table token must be specified. If no table token is specified, a security check for table access is performed.

TRANSACTION_NUMBER

Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

SERVER_CONNECTION_FAILED
ACCESS_NOT_ALLOWED
TABLE_NOT_FOUND
SHARED_ACCESS_CONFLICT
EXCLUSIVE_ACCESS_CONFLICT
ALREADY_SET
INCORRECT_STATE
OPTION_NOT_SUPPORTED
TABLE_TOKEN_INVALID
TABLE_DESTROYED
POOL_STATE_ERROR
CF_ACCESS_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCU gate, BACKOUT function

This function backs out the changes made by an active unit of work and releases the locks before returning control to the caller.

Input Parameters

TRANSACTION_NUMBER

Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

UOW_ID
This 8-character string combines the subsystem name with the unit of work identification in the client region to form the fully qualified unit of work identifier.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
RECOVERY_NOT_ENABLED
UOW_NOT_FOUND
UOW_MADE_NO_CHANGES
POOL_STATE_ERROR
CF_ACCESS_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCCU gate, COMMIT function

This function commits the changes made by a unit of work and releases all locks before returning control to the caller.

Input Parameters

TRANSACTION_NUMBER
Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

UOW_ID
This 8-character string combines the subsystem name with the unit of work identification in the client region to form the fully qualified unit of work identifier.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
RECOVERY_NOT_ENABLED
UOW_NOT_FOUND
UOW_MADE_NO_CHANGES
UOW_FAILED
NO_SPACE_IN_POOL
POOL_STATE_ERROR
CF_ACCESS_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
```

DISASTER
INVALID
KERNERROR
PURGED

FCCU gate, INQUIRE function

This function returns information about the status of an active unit of work.

Input Parameters

BROWSE

Optional Parameter

Specifies whether the inquire is for a single unit of work or for the first or next unit of work in a browse. If this parameter is omitted, the inquire is assumed to be a single unit of work inquire. Values for the parameter are:

FIRST
NEXT

The FIRST option indicates a search for a UOW ID greater than or equal to the specified UOW ID, and NEXT indicates a search for a UOW ID greater than the specified UOW ID.

TRANSACTION_NUMBER

Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

UOW_ID

This 8-character string combines the subsystem name with the unit of work identification in the client region to form the fully qualified unit of work identifier.

UOW_RESTARTED

Optional Parameter

Specifies that the function must select only units of work that have or have not been through restart processing. Values for the parameter are:

YES
NO

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

SERVER_CONNECTION_FAILED
RECOVERY_NOT_ENABLED
UOW_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

UOW_ID

The 8-character unit of work identification.

UOW_RESTARTED

Indicates whether the unit of work has been through restart. Values for the parameter are:

YES
NO

UOW_RETAINED
Indicates whether the locks for the unit of work have been marked as retained, either explicitly in the current connection or implicitly by a restart. Values for the parameter are:
YES
NO

UOW_STATE
Indicates the state of an active unit of work. Values for the parameter are:

IN_FLIGHT
The unit of work has made changes but has not yet reached the stage of prepare to commit.

IN_DOUBT
The unit of work has been prepared but not committed or backed out.

IN_COMMIT
Commit processing has started.

IN_BACKOUT
Backout processing has started.

When commit or backout processing completes, the unit of work is deleted.

FCCU gate, PREPARE function

This function marks a unit of work as prepared to be committed. The PREPARE function is required to support 2-phase commit protocols and is ignored if the unit of work is already in a prepared or retained state.

Input Parameters

TRANSACTION_NUMBER
Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

UOW_ID
This 8-character string combines the subsystem name with the unit of work identification in the client region to form the fully qualified unit of work identifier.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
SERVER_CONNECTION_FAILED
RECOVERY_NOT_ENABLED
UOW_NOT_FOUND
UOW_MADE_NO_CHANGES
UOW_FAILED
NO_SPACE_IN_POOL
POOL_STATE_ERROR
CF_ACCESS_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID

KERNERROR
PURGED

FCCU gate, RESTART function

This function establishes recovery status at startup. Recoverable operations for the client region are enabled and state information relating to any unresolved units of work is rebuilt.

Input Parameters

TRANSACTION_NUMBER

Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

UOW_SUBSYSTEM_NAME

Optional Parameter

The 8-character subsystem name to be used at the first part of the unit of work identifier for units of work relating to the client region. For a CICS client region, this parameter is ignored and the CICS applid is used. For a non-CICS client region, if this parameter is omitted, or specified as spaces, the MVS job name is used instead.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

SERVER_CONNECTION_FAILED
SUBSYSTEM_ALREADY_ACTIVE
RESTART_ALREADY_ACTIVE
TABLE_OPEN_FAILED
NO_SPACE_IN_POOL
POOL_STATE_ERROR
CF_ACCESS_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCCU gate, RETAIN function

This function marks any locks relating to the named unit of work as retained.

Input Parameters

TRANSACTION_NUMBER

Optional Parameter

This 4-character string identifies the requesting task in the debug trace if used.

UOW_ID

This 8-character string combines the subsystem name with the unit of work identification in the client region to form the fully qualified unit of work identifier.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
RECOVERY_NOT_ENABLED
UOW_NOT_FOUND
UOW_MADE_NO_CHANGES
UOW_FAILED
NO_SPACE_IN_POOL
POOL_STATE_ERROR
CF_ACCESS_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCDN gate, CATALOG_DSNB function

This function catalogs data set name (DSN) blocks.

Input Parameters

FILE_NAME

The 8-character name of the file.

TYPE_OF_CONNECTION

Specifies whether the connection is being made to a base or an object.

Values for the parameter are:

```
OBJ
BASE
```

FILE_NAME

The 8-character name of the file.

Output Parameters

REASON

The following value is returned when RESPONSE is EXCEPTION:

```
FILE_NOT_FOUND
```

The following value is returned when RESPONSE is DISASTER:

```
CATALOG_WRITE_FAILED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
DISASTER
PURGED
```

FCDN gate, COMMIT_DSNREFS function

This function commits data set name (DNS) block references.

Input Parameters

TOKEN

A token passed to the COMMIT_DNSREFS function.

Output Parameters

REASON

The following value is returned when RESPONSE is DISASTER:

```
CATALOG_WRITE_FAILED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
DISASTER
INVALID
PURGED
```

FCDN gate, CONNECT_DSNB function

This function connects a file control table entry (FCTE) to a data set name (DSN) block. If the DSN block does not already exist, DFHFCDN creates a new block before connecting it.

Input Parameters

CATALOG_CONNECTION

Values for the parameter are:

```
YES
NO
```

DSNAME

The 44-character name of the data set.

FILE_NAME

The 8-character name of the file.

TYPE_OF_CONNECTION

Specifies whether the connection is being made to a base or an object.

Values for the parameter are:

```
OBJ
BASE
```

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
FILE_NOT_FOUND
DO_NOT_REALLOCATE
FILE_NOT_CLOSED
FILE_NOT_DISABLED
```

The following value is returned when RESPONSE is INVALID:

```
INVALID_TOKEN
```

The following values are returned when RESPONSE is DISASTER:

```
CATALOG_WRITE_FAILED
GETMAIN_FAILED
TM_ADD_FAILED
TM_LOCATE_FAILED
TM_UNLOCK_FAILED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
PURGED

FCDN gate, DELETE_DSNB function

This function checks to ensure that the data set name (DSN) block can be deleted. If the deletion can proceed, the table manager is called to delete the DSN from the DSN index, and the storage domain is called to free the storage.

Input Parameters

DSNAME
The 44-character name of the data set.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DSNB_INUSE
DSNB_NOT_FOUND
DSNB_LOCK_HELD

The following values are returned when RESPONSE is DISASTER:
CATALOG_DELETE_FAILED
FIND_RETAINED_FAILED
FREEMAIN_FAILED
TM_DELETE_FAILED
TM_QUIESCE_FAILED
TM_UNQUIESCE_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

FCDN gate, DISCONNECT_DSNB function

This function breaks the connection between the file control table entry (FCTE) and the data set name (DSN) block. The DSN block remains even if no other FCT entries are connected to it. The request is rejected if uncommitted updates (retained locks) exist for the file.

Input Parameters

DECREMENT_FLAG
Optional Parameter

Flag to indicate that the number of files connected to the DSN block is reduced by one.

FILE_NAME
The 8-character name of the file.

TYPE_OF_CONNECTION
Specifies whether the connection is being made to a base or an object.

Values for the parameter are:
OBJ
BASE

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
FILE_NOT_FOUND
DO_NOT_REALLOCATE
DSNB_NOT_FOUND
FILE_NOT_CLOSED
FILE_NOT_DISABLED

The following value is returned when RESPONSE is INVALID:
INVALID_TOKEN

The following value is returned when RESPONSE is DISASTER:
CATALOG_DELETE_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
PURGED

FCDN gate, END_DSNB_BROWSE function

This function ends the browse of the data set name (DSN) blocks.

Input Parameters

BROWSE_TOKEN
The token returned from the START_DSNB_BROWSE function.

Output Parameters

REASON
The following value is returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

The following value is returned when RESPONSE is DISASTER:
FREEMAIN_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
INVALID
PURGED

FCDN gate, GET_NEXT_DSNB function

This function browses the next data set name (DSN) block and returns the attributes to the caller.

Input Parameters

BROWSE_TOKEN
The token returned from the START_DSNB_BROWSE function.

OBTAIN_VSAM_CATALOG_DATA
Optional Parameter

Values for the parameter are:
YES
NO

Output Parameters

ACCMETH
Specifies the access method.

Values for the parameter are:
VSAM
BDAM
NOT_APPLICABLE

AVAILABILITY
Specifies the availability of the data set.

Values for the parameter are:
AVAILABLE
UNAVAILABLE
NOT_APPLICABLE

BASEDSNAME
The 44-character name of the base data set.

DSNB_TYPE
Specifies the data set name block type.

Values for the parameter are:
PATH
BASE
NOT_APPLICABLE

DSNB_VALID_STATUS
Specifies the status of the DSN block.

Values for the parameter are:
YES
NO

FILECOUNT
This halfword binary field specifies the file count.

FWDRECOVLOG
This halfword binary field specifies the log ID to which the after images for forward recovery are written.

FWDRECOVLSN
This 26-character string specifies the forward recovery log stream name (LSN).

IMAGE
Indicates whether backup images are to be fuzzy or sharp. Values for the parameter are:
FUZZY
SHARP
NOT_APPLICABLE

LOSTLOCKS
Returns the lost locks status of the data set. Values for the parameter are:
REMLOSTLOCKS
RECOVERLOCKS
NOT_APPLICABLE

NOLOSTLOCKS

REASON

The following value is returned when RESPONSE is EXCEPTION:

INVALID_BROWSE_TOKEN

The following values are returned when RESPONSE is EXCEPTION:

DATASET_MIGRATED
DSNB_NOT_FOUND
INQ_DATASET_NOT_KNOWN
VSAM_ERROR
END_OF_LIST

The following values are returned when RESPONSE is DISASTER:

DISASTER_PERCOLATE
TM_LOCATE_FAILED
TM_UNLOCK_FAILED
TM_GETNEXT_FAILED
VSAM_CATALOG_ERROR

RECOV_VALID_STATUS

Values for the parameter are:

YES
NO
NOT_APPLICABLE

RECOVSTATUS

Specifies the recovery status for the data set. Values for the parameter are:

FWD_RECOV
RECOV
NOT_APPLICABLE
NOT_RECOV

REPLICATION LOGGING

Values for the parameter are:

YES
NO
NOT_APPLICABLE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
INVALID
EXCEPTION
DISASTER

FCDN gate, INQUIRE_DSNB function

This function returns the attributes stored in the data set name (DSN) block to the caller.

Input Parameters

DSNAME

The 44-character name of the data set.

OBTAIN_VSAM_CATALOG_DATA

Optional Parameter

Values for the parameter are:

YES
NO

Output Parameters

ACCMETH
Specifies the access method.

Values for the parameter are:
- VSAM
- BDAM
- NOT_APPLICABLE

AVAILABILITY
Specifies the availability of the data set.

Values for the parameter are:
- AVAILABLE
- UNAVAILABLE
- NOT_APPLICABLE

BASEDSNAME
The 44-character name of the base data set.

DSNB_TYPE
Specifies the data set name block type.

Values for the parameter are:
- PATH
- BASE
- NOT_APPLICABLE

DSNB_VALID_STATUS
Specifies the status of the DSN block.

Values for the parameter are:
- YES
- NO

FILECOUNT
This halfword binary field specifies the file count.

FWDRECOVLOG
This halfword binary field specifies the log ID to which the after images for forward recovery are written.

FWDRECOVLSN
This 26-character string specifies the forward recovery log stream name (LSN).

IMAGE
Indicates whether backup images are to be fuzzy or sharp. Values for the parameter are:
- FUZZY
- SHARP
- NOT_APPLICABLE

LOSTLOCKS
Returns the lost locks status of the data set. Values for the parameter are:
- REMLOSTLOCKS
- RECOVERLOCKS
- NOT_APPLICABLE
- NOLOSTLOCKS

REASON
The following values are returned when RESPONSE is EXCEPTION:
- DATASET_MIGRATED
- DSNB_NOT_FOUND
- INQ_DATASET_NOT_KNOWN
- INQ_BASEDSNAME_ERROR
- VSAM_ERROR

The following values are returned when RESPONSE is DISASTER:

DISASTER_PERCOLATE
TM_LOCATE_FAILED
TM_UNLOCK_FAILED
VSAM_CATALOG_ERROR

RECOV_VALID_STATUS
Values for the parameter are:
YES
NO
NOT_APPLICABLE

RECOVSTATUS
Specifies the recovery status for the data set. Values for the parameter are:
FWD_RECOV
RECOV
NOT_APPLICABLE
NOT_RECOV

REPLICATION_LOGGING
Values for the parameter are:
YES
NO
NOT_APPLICABLE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER

FCDN gate, RESET_ALL_QUIESCE_STATUS function

DFHFCRD calls this function. The data set name (DSN) block table is scanned and the quiesce status is reset to normal in each DSN block.

Input Parameters

None.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
TM_GETNEXT_FAILED
TM_UNLOCK_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
INVALID
PURGED

FCDN gate, SET_CATALOG_RECOVERED function

This function causes a named data set to be set to the forward recovered state.

Input Parameters

DSNAME

The 44-character name of the data set.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
DATASET_NOT_KNOWN
DSNB_BDAM_OR_PATH
DSNB_INVREQ
DSNB_NOT_FOUND
FILES_OPEN_AGAINST_DATASET
NO_FUZZY_SUPPORT
```

The following values are returned when RESPONSE is DISASTER:

```
SET_CAT_REC_FAILED
TM_LOCATE_FAILED
TM_UNLOCK_FAILED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
PURGED
```

FCDN gate, SET_DSNB function

This function sets the availability of the named data set.

Input Parameters

AVAILABILITY

Specifies the availability of the data set. Values for the parameter are:

```
AVAILABLE
UNAVAILABLE
```

DSNAME

The 44-character name of the data set.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
DATASET_MIGRATED
DSNB_BDAM_OR_PATH
DSNB_INVREQ
DSNB_NOT_FOUND
VSAM_ERROR
```

The following values are returned when RESPONSE is DISASTER:

```
CATALOG_WRITE_FAILED
DISASTER_PERCOLATE
TM_LOCATE_FAILED
TM_UNLOCK_FAILED
VSAM_CATALOG_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
PURGED

FCDN gate, START_DSNB_BROWSE function

This function starts a browse of the data set name (DSN) block.

Input Parameters

None.

Output Parameters

BROWSE_TOKEN
The token returned from the START_DSNB_BROWSE function.

REASON
The following value is returned when RESPONSE is DISASTER:
GETMAIN_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
PURGED

FCDN gate, UPDATE_RECOVERY_POINTS function

This function updates the recovery point location.

Input Parameters

RECOVERY_POINT
This 8-character field specifies the new location of the recovery point. The recovery point is the place where a forward-recovery utility starts applying log records.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
SET_RECOVERY_POINT_FAILED
TM_GETNEXT_FAILED
TM_UNLOCK_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
INVALID
PURGED

FCDS gate, DISCONNECT_CFDT_POOLS function

This function causes CICS to disconnect from any coupling facility data table pools to which it is connected.

Input Parameters

None.

Output Parameters

REASON

The following value is returned when RESPONSE is EXCEPTION:

```
CFDT_DISCONNECT_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
DISASTER
EXCEPTION
INVALID
KERNERROR
PURGED
```

FCDS gate, EXTRACT_CFDT_STATS function

This function causes statistics relating to coupling facility data table usage to be extracted from the coupling facility data tables server.

Input Parameters

FCTE_POINTER

The address of the FCTE entry of the file for which CFDT statistics are to be extracted.

RESET_STATISTICS

Indicates whether the statistics fields are to be reset to zero or not. Values for the parameter are:

```
YES
NO
```

TRANSACTION_NUMBER

Optional Parameter

4-digit transaction number, which is passed to the CFDT server for inclusion in trace messages.

Output Parameters

CONTENTION_COUNT

Optional Parameter

This fullword parameter returns the number of contentions that have been detected, for a coupling facility data table that uses the contention update model.

CURRENT_RECORDS

Optional Parameter

This fullword parameter returns the current number of records in the coupling facility data table.

CURRENT_USERS
Optional Parameter

This fullword parameter returns the current number of users of the coupling facility data table; that is, the number of opens issued against it.

HIGHEST_RECORDS
Optional Parameter

This fullword parameter returns the highest number of records that have been in this coupling facility data table since it was last created.

MAXIMUM_RECORDS
Optional Parameter

This fullword parameter returns the current value of the MAXNUMRECS limit for the data table.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CFDT_CONNECT_ERROR
CFDT_DISCONNECT_ERROR
CFDT_REOPEN_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_STATS_ERROR
CFDT_SYSIDERR
CFDT_TABLE_GONE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
POOL_ELEMENT_NOT_FOUND
ABEND
DISASTER_PERCOLATION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
DISASTER
EXCEPTION
INVALID
KERNERROR
PURGED
```

FCDU gate, BACKOUT function

This function calls the coupling facility data table (CFDT) server to back out a unit of work (UOW) that has made recoverable updates to one or more CFDTs.

Input Parameters

POOL_ELEM_ADDR
The address of the pool element that identifies the CFDT pool for which the backout is to be issued. One or more of the CFDTs updated by the UOW reside in this pool. The backout call is issued to the CFDT server for this pool.

POOL_NAME
The name of the CFDT pool. The pool name is included for diagnostic purposes.

UOW_ID
The identifier for the unit of work that is going to be backed out.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
SERVER_CONNECTION_FAILED
RECOVERY_NOT_ENABLED
UOW_NOT_FOUND
UOW_MADE_NO_CHANGES
POOL_STATE_ERROR
CF_ACCESS_ERROR
CFDT_SYSIDERR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_CONNECT_ERROR
CFDT_DISCONNECT_ERROR
RESYNC_RETRY_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
KERNERROR
PURGED

FCDU gate, COMMIT function

This function calls the coupling facility data table (CFDT) server to commit a unit of work (UOW) that has made recoverable updates to one or more CFDTs.

Input Parameters

POOL_ELEM_ADDR
The address of the pool element that identifies the CFDT pool for which the backout is to be issued. One or more of the CFDTs updated by the UOW reside in this pool. The backout call is issued to the CFDT server for this pool.

POOL_NAME
The name of the CFDT pool. The pool name is included for diagnostic purposes.

UOW_ID
The identifier for the unit of work that is going to be committed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
SERVER_CONNECTION_FAILED
RECOVERY_NOT_ENABLED

UOW_NOT_FOUND
UOW_MADE_NO_CHANGES
UOW_FAILED
NO_SPACE_IN_POOL
POOL_STATE_ERROR
CF_ACCESS_ERROR
CFDT_SYSIDERR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_CONNECT_ERROR
CFDT_DISCONNECT_ERROR
RESYNC_RETRY_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
KERNERROR
PURGED

FCDU gate, INQUIRE function

This function issues an INQUIRE to the coupling facility data table (CFDT) to obtain information about the status of an active unit of work (UOW).

Input Parameters

BROWSE
Optional Parameter

Specifies whether the inquire is for a single UOW or for the first or next UOW in a browse. Values for the parameter are:
FIRST
NEXT

If the **BROWSE** parameter is omitted, the request is treated as a single UOW inquire. Setting the **BROWSE** parameter to FIRST indicates a search for a UOW ID greater than or equal to the specified UOW ID. Setting the **BROWSE** parameter to NEXT indicates a search for a UOW ID greater than the specified UOW ID.

POOL_ELEM_ADDR
The address of the pool element that identifies the CFDT pool for which the backout is to be issued. One or more of the CFDTs updated by the UOW reside in this pool. The backout call is issued to the CFDT server for this pool.

POOL_NAME
The name of the CFDT pool. The pool name is included for diagnostic purposes.

UOW_ID
This 8-character string identifies the UOW for which status information is being requested or gives the ID for the previous UOW in the browse.

UOW_RESTARTED
Optional Parameter

Indicates whether the inquire will select only UOWs that have been through restart processing. Values for the parameter are:
YES
NO

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
SERVER_CONNECTION_FAILED
RECOVERY_NOT_ENABLED
UOW_NOT_FOUND
CF_ACCESS_ERROR
CFDT_SYSIDERR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_CONNECT_ERROR
CFDT_DISCONNECT_ERROR
RESYNC_RETRY_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
KERNERROR
PURGED

RETURNED_UOW_ID
This 8-character string specifies the UOW for which the browse is returning status information.

UOW_RESTART_STATE
Indicates whether the UOW has been through restart processing. Values for the parameter are:
YES
NO

UOW_RETAINED
Indicates whether the locks for the UOW have been retained. Values for the parameter are:
YES
NO

UOW_STATE
Indicates the state of the UOW. Values for the parameter are:
IN_FLIGHT
IN_DOUBT
IN_COMMIT

IN_BACKOUT

FCDU gate, PREPARE function

This function calls the coupling facility data table (CFDT) server to prepare a unit of work that has made recoverable updates to one or more coupling facility data tables.

Input Parameters

POOL_ELEM_ADDR
The address of the pool element that identifies the CFDT pool for which the prepare is going to be issued. One or more of the CFDTs updated by the unit of work reside in this pool.

POOL_NAME
The name of the CFDT pool. The pool name is included for diagnostic purposes.

UOW_ID
The identifier for the unit of work that is to be prepared.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
SERVER_CONNECTION_FAILED
RECOVERY_NOT_ENABLED
UOW_NOT_FOUND
UOW_MADE_NO_CHANGES
UOW_FAILED
NO_SPACE_IN_POOL
POOL_STATE_ERROR
CF_ACCESS_ERROR
CFDT_SYSIDERR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_CONNECT_ERROR
CFDT_DISCONNECT_ERROR
RESYNC_RETRY_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
KERNERROR
PURGED

FCDU gate, RESTART function

This function establishes recovery status for a coupling facility data table (CFDT) pool when a CICS region has successfully connected to it.

Input Parameters

POOL_ELEM_ADDR
The address of the pool element that identifies the CFDT pool for which recovery status is to be established.

POOL_NAME
The name of the CFDT pool. The pool name is included for diagnostic purposes.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
SUBSYSTEM_ALREADY_ACTIVE
RESTART_ALREADY_ACTIVE
TABLE_OPEN_FAILED
NO_SPACE_IN_POOL
CF_ACCESS_ERROR
CFDT_SYSIDERR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_CONNECT_ERROR
CFDT_DISCONNECT_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
DISASTER
EXCEPTION
INVALID
KERNERROR
PURGED
```

RETURNED_UOW_ID
The unit of work for which the browse is returning status information.

UOW_RESTART_STATE
Indicates whether the unit of work has been through restart processing.

UOW_RETAINED
Indicates whether the locks for the unit of work have been retained.

UOW_STATE
Indicates the state of the unit of work. Values for the parameter are:

```
IN_FLIGHT
IN_DOUBT
IN_COMMIT
IN_BACKOUT
```

FCDU gate, RETAIN function

This function calls the coupling facility data table (CFDT) server to convert locks held by the unit of work against recoverable CFDTs into retained locks.

Input Parameters

POOL_ELEM_ADDR
The address of the pool element that identifies the CFDT pool for which the retain is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The retain call will be issued to the CFDT server for this pool.

POOL_NAME
The name of the CFDT pool. The pool name is included for diagnostic purposes.

UOW_ID
The identifier for the unit of work for which the locks are going to be retained.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
SERVER_CONNECTION_FAILED
RECOVERY_NOT_ENABLED
UOW_NOT_FOUND
UOW_MADE_NO_CHANGES
UOW_FAILED
NO_SPACE_IN_POOL
POOL_STATE_ERROR
CF_ACCESS_ERROR
CFDT_SYSIDERR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_CONNECT_ERROR
CFDT_DISCONNECT_ERROR
RESYNC_RETRY_FAILED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
DISASTER
EXCEPTION
INVALID
KERNERROR
PURGED
```

FCDY gate, RESYNC_CFDT_LINK function

This function causes a link between a unit of work and a coupling facility data table pool to be resynchronized.

Input Parameters

POOL_NAME
The 8-character name of the coupling facility data table pool for which the link is to be resynchronized.

UOW_ID
This 8-character string identifies the link to be resynchronized.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INITIATE_RECOVERY_FAILED
TERMINATE_RECOVERY_FAILED
CFDT_SERVER_CALL_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
KERNERROR
PURGED

FCDY gate, RESYNC_CFDT_POOL function

This function causes a coupling facility data table pool to be resynchronized.

Input Parameters

POOL_NAME
The 8-character name of the coupling facility data table pool that is to be resynchronized.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INITIATE_RECOVERY_FAILED
TERMINATE_RECOVERY_FAILED
CFDT_SERVER_CALL_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION

INVALID
KERNERROR
PURGED

FCDY gate, RETURN_CFDT_ENTRY_POINTS function

This function causes module DFHFCDY to return the entry point addresses of the other modules with which it is link-edited.

Input Parameters

None.

Output Parameters

CFDT_EP_DFHFCDW
The entry point address of module DFHFCDW.

CFDT_EP_DFHFCDU
The entry point address of module DFHFCDU.

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
KERNERROR
PURGED

FCFL gate, END_UOWDSN_BROWSE function

After a browse of all the data set failures in a unit of work, the END_UOWDSN_BROWSE function releases the storage that was used for a snapshot of the failures.

Input Parameters

BROWSE_TOKEN
The token that was used for the browse.

Output Parameters

REASON
The following value is returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
DISASTER
INVALID
PURGED

FCFL gate, FIND_RETAINED function

This function looks for any file lasting access blocks associated with the specified data set that are flagged as retained, indicating that retained locks are associated with the data set.

Input Parameters

DSNAME

The 44-character name of the data set for which associated retained locks are to be found.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
DISASTER_PERCOLATION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

RETLOCKS

Indicates whether retained locks are associated with the data set. Values for the parameter are:

RETAINED
NORETAINED

FCFL gate, FORCE_INDOUBTS function

The CEMT and EXEC CICS SET DSNAME() UOWACTION(COMMIT | BACKOUT | FORCE) commands use this function. Shunted indoubt units of work are forced to complete in the specified direction.

Input Parameters

DSNAME

The 44-character name of the data set for which shunted indoubt units of work are to be forced to complete.

DIRECTION

The direction that the units of work are to complete. Values for the parameter are:

FORWARD
BACKWARD
HEURISTIC

'A value of FORWARD commits the units of work, a value of BACKWARD backs out the units of work, and a value of HEURISTIC uses the action specified on the transaction definition.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCFL gate, GET_NEXT_UOWDSN function

This function returns the failure information for the next data set that has a failure in the unit of work being browsed.

Input Parameters

BROWSE_TOKEN

The token for the browse that was returned by a START_UOWDSN_BROWSE call.

Output Parameters

DSNAME

The 44-character name of the data set for which failure information is returned.

RLSACCESS

Optional Parameter

Indicates whether the data set was last open in RLS or non-RLS access mode. Values for the parameter are:

```
RLS
NOTRLS
```

CAUSE

Optional Parameter

Indicates the cause of the failure. Values for the parameter are:

```
CACHE
RLSSERVER
CONNECTION
DATASET
UNDEFINED
```

RETAIN_REASON

Optional Parameter

Indicates the reason for the failure. Values for the parameter are:

```
RLSGONE
COMMITFAIL
IOERROR
DATASETFULL
INDEXRECFULL
OPENERROR
DELEXITERROR
DEADLOCK
BACKUPNONBWO
```

LOCKSTRUCFULL
FAILEDBKOUT
NOTAPPLIC
RR_COMMITFAIL
RR_INDOUBT

REASON

The following value is returned when RESPONSE is EXCEPTION:

END_OF_LIST

The following value is returned when RESPONSE is INVALID:

INVALID_BROWSE_TOKEN

The following values are returned when RESPONSE is DISASTER:

ABEND
DISASTER_PERCOLATION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID

FCFL gate, RESET_BFAILS function

The CEMT and EXEC CICS SET DSNAME() ACTION(RESETLOCKS) commands use this function. Shunted unit of work log records, which hold backout-failure or commit-failure locks on the specified data set, are purged and locks are released.

Input Parameters

DSNAME

The 44-character name of the data set for which backout and commit failures are to be reset.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
DISASTER_PERCOLATION
REMOVE_FAILURE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCFL gate, RETRY function

The CEMT and EXEC CICS SET DSNAME() UOWACTION(RETRY) commands use this function. The RETRY function retries any failed backouts and commits for the specified data set by informing DFHFCRR that the failed resource is now available.

Input Parameters

DSNAME

The 44-character name of the data set for which backout and commits are to be retried.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
RESOURCE_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCFL gate, START_UOWDSN_BROWSE function

This function starts a browse of the data set failures in a unit of work. A snapshot of the failed data sets for the UOW and the reasons for the failures is collected in an in-storage table to be browsed by the GET_NEXT_UOWDSN function.

Input Parameters

UOW

The 8-byte local unit of work identifier.

Output Parameters

BROWSE_TOKEN

A token used during the browse.

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
UOW_NOT_FOUND
NO_FLABS_FOUND
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
PURGED
```

FCFL gate, TEST_USER function

This function is used to test if the task has updated a record and established itself as a file user, either for any data set or for a specified data set. It can be used either as a domain subroutine call or as an inline macro.

Input Parameters

ENVIRONMENT

Optional Parameter

A fullword environment identifier. If specified, the function tests whether the task is a user of any files in that environment.

DSNAME

Optional Parameter

Specifies that a particular data set is to be tested.

Output Parameters

FLAB_PTR

The address of a file lasting access block (FLAB) that was found by the test. The return of a non-zero value indicates that the user is a task.

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCFR gate, CLEAR_ENVIRONMENT function

Scan the FRTE chain and find all FRTEs for the specified Environment. Clean up the file control state for this environment.

Cleaning up the file control state consists of the following steps:

1. Issue END_BROWSE for any active START_BROWSE.
2. Issue UNLOCK for any active READ_UPDATE or WRITE_MASSINSERT.

Input Parameters

ENVIRONMENT_IDENTIFIER

A token that identifies the caller's environment.

CLEAR_AFTER_ABEND

Optional Parameter

A binary value that indicates whether the request follows a transaction abend, and that the environment must be cleared.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
```

The following values are returned when RESPONSE is EXCEPTION:

CLEAR_ENVIRONMENT_FAILED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCFR gate, DELETE function

Delete a record from a file.

Input Parameters

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:

NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:

NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

GENERIC
A binary value that specifies whether the search key is a generic key.

Values for the parameter are:

NO
YES

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:

DEBKEY
DEBREC
KEY
RBA
RRN

BASE_RECORD_ID_ADDRESS
Optional Parameter

The address of the base record identifier.

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME

Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM

Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS

Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:

```
BDAM_DELETE
BDAM_KEY_CONVERSION
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
DATASET_BEING_COPIED
DEADLOCK_DETECTED
DELETE_AFTER_READ_UPDATE
ESDS_DELETE
FILE_DISABLED
FILE_NOT_OPEN
FULL_KEY_WRONG_LENGTH
GENERIC_DELETE_NOT_KSDS
GENERIC_KEY_TOO_LONG
IO_ERROR
KEY_LENGTH_NEGATIVE
LOADING
LOCK_STRUCTURE_FULL
LOCKED
LOST_LOCKS
NOSUSPEND_NOT_RLS
NOT_IN_SUBSET
PREVIOUS_RLS_FAILURE
RBA_ACCESS_TO_RLS_KSDS
RECLEN_EXCEEDS_LOGGER_BFSZ
RECORD_BUSY
RECORD_NOT_FOUND
RESTART_FAILED
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
```

SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
STORE_FAIL
SYSIDERR
TIMEOUT
TOO_MANY_CFDTS_IN_UOW
UPDATE_NOT_AUTHORISED
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DELETED_RECORD_COUNT
The number of records deleted by the request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, END_BROWSE function

End a browse operation on a file.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

CFDT_LOAD
Optional Parameter

A binary value that indicates whether the request is part of the browse operation used to read records from the source data set during loading of a coupling facility data table.

Values for the parameter are:
NO
YES

CLEAR_AFTER_ABEND
Optional Parameter

A binary value that indicates whether the request follows a transaction abend, and that the environment must be cleared.

Values for the parameter are:
NO
YES

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
TABLE_TOKEN_INVALID

The following values are returned when RESPONSE is EXCEPTION:
CACHE_FAILURE
CFDT_TABLE_GONE
CLEAR_ABENDED
FILENOTFOUND
ISC_NOT_SUPPORTED
ISCINVREQ
NOTAUTH
PREVIOUS_RLS_FAILURE
READ_NOT_AUTHORISED
REMOTE_INVREQ
RLS_DISABLED
RLS_FAILURE
SHIPPED_SECURITY_FAILURE
SYSIDERR

UNKNOWN_REQID_ENDBR
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, FREE_UNUSED_BUFFERS function
Free any file control buffers that are not in use.

Output Parameters
REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCFR gate, PREPARE_FILE_REQUEST function
Prepare to commit file changes made in a unit of work.

Input Parameters
FILE_NAME
The name of the FILE resource.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters
REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is EXCEPTION:

PREPARE_FAILED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:

NO
YES

FCFR gate, PREPARE_TO_BACKOUT function

Prepare to back out file changes made in a unit of work.

Input Parameters

FILE_NAME
The name of the FILE resource.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:

NO
YES

FCFR gate, READ_INTO function

Read a file record into a buffer provided by the caller.

Input Parameters

BUFFER_ADDRESS
The address of the caller's buffer.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:

NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

GENERIC
A binary value that specifies whether the search key is a generic key.

Values for the parameter are:
NO
YES

KEY_COMPARISON
A value that specifies whether the search can be satisfied only by a record having the same key as that specified in the record identification field parameter, or by a record having a greater key.

Values for the parameter are:
EQUAL
GTEQ

READ_INTEGRITY
Specifies the degree of read integrity for the request.

Values for the parameter are:
CR
FCT_VALUE
NRI
RR

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

BASE_RECORD_ID_ADDRESS
Optional Parameter

The address of the base record identifier.

BUFFER_LENGTH
Optional Parameter

The length of the caller's buffer.

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

SUPPRESS_LENGERR
Optional Parameter

A binary value that indicates whether length error indications are to be suppressed.

Values for the parameter are:

NO
YES

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
TABLE_TOKEN_INVALID

The following values are returned when RESPONSE is EXCEPTION:

BDAM_KEY_CONVERSION
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
CR_NOT_RLS
FILE_DISABLED
FILE_NOT_OPEN
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
IO_ERROR
ISC_NOT_SUPPORTED
ISCINVREQ
KEY_LENGTH_NEGATIVE
LOADING
LOCKED
LOST_LOCKS
NO_VARIABLE_LENGTH
NOSUSPEND_NOT_RLS
NOT_IN_SUBSET
NOTAUTH
PREVIOUS_RLS_FAILURE
RBA_ACCESS_TO_RLS_KSDS

READ_NOT_AUTHORISED
RECORD_BUSY
RECORD_NOT_FOUND
REMOTE_INVREQ
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
RR_NOT_RLS
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIP
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

LENGTH_ERROR_CODE
A value that provides details of a length error that occurred when processing the request.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

FCFR gate, READ_NEXT_INTO function

During a file browse, read the next record, and return the record into a buffer provided by the caller.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

BUFFER_ADDRESS
The address of the caller's buffer.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

READ_INTEGRITY
Specifies the degree of read integrity for the request.

Values for the parameter are:
CR
FCT_VALUE
NRI
RR

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

BUFFER_LENGTH
Optional Parameter

The length of the caller's buffer.

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

SUPPRESS_LENGERR
Optional Parameter

A binary value that indicates whether length error indications are to be suppressed.

Values for the parameter are:
NO
YES

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
TABLE_TOKEN_INVALID

The following values are returned when RESPONSE is EXCEPTION:
BDAM_KEY_CONVERSION
BDAM_READ_PREVIOUS
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
CR_NOT_RLS
END_OF_FILE
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
ILLEGAL_KEY_TYPE_CHANGE
IO_ERROR
ISC_NOT_SUPPORTED
ISCINVREQ
KEY_LENGTH_NEGATIVE
LOCKED
NO_VARIABLE_LENGTH
NOSUSPEND_NOT_RLS
NOTAUTH
PREVIOUS_RLS_FAILURE

READ_NOT_AUTHORISED
READPREV_IN_GENERIC_BROWSE
RECORD_BUSY
RECORD_NOT_FOUND
REMOTE_INVREQ
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
RR_NOT_RLS
SELF_DEADLOCK_DETECTED
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
UNKNOWN_REQID_READPREV
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

FULL_RECORD_ID_LENGTH
The length of the record key.

LENGTH_ERROR_CODE
A value that provides details of a length error that occurred when processing the request.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:

NO
YES

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

FCFR gate, READ_NEXT_SET function

During a file browse, read the next record, and return a pointer to a buffer containing the data.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

READ_INTEGRITY
Specifies the degree of read integrity for the request.

Values for the parameter are:
CR
FCT_VALUE
NRI
RR

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

CFDT_LOAD
Optional Parameter

A binary value that indicates whether the request is part of the browse operation used to read records from the source data set during loading of a coupling facility data table.

Values for the parameter are:

NO
YES

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
TABLE_TOKEN_INVALID

The following values are returned when RESPONSE is EXCEPTION:

BDAM_KEY_CONVERSION
BDAM_READ_PREVIOUS
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
CR_NOT_RLS
END_OF_FILE
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
ILLEGAL_KEY_TYPE_CHANGE
IO_ERROR
ISC_NOT_SUPPORTED
ISCINVREQ
KEY_LENGTH_NEGATIVE
LOCKED
NOSUSPEND_NOT_RLS
NOTAUTH
PREVIOUS_RLS_FAILURE
READ_NOT_AUTHORISED
READPREV_IN_GENERIC_BROWSE
RECORD_BUSY
RECORD_NOT_FOUND

REMOTE_INVREQ
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
RR_NOT_RLS
SELF_DEADLOCK_DETECTED
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
UNKNOWN_REQID_READPREV
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

FULL_RECORD_ID_LENGTH
The length of the record key.

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

RECORD_ADDRESS
The address of the target record.

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, READ_NEXT_UPDATE_INTO function

During a file browse, read the previous record for updating, and return the record into a buffer provided by the caller.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

BUFFER_ADDRESS
The address of the caller's buffer.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

BUFFER_LENGTH
Optional Parameter

The length of the caller's buffer.

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

SUPPRESS_LENGERR
Optional Parameter

A binary value that indicates whether length error indications are to be suppressed.

Values for the parameter are:
NO

YES

WORK_ELEMENT_ADDRESS

Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
DISASTER_PERCOLATION
SECURITY_FAILURE

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_UPD_NOT_RLS
CACHE_FAILURE
DATASET_BEING_COPIED
END_OF_FILE
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
ILLEGAL_KEY_TYPE_CHANGE
IO_ERROR
KEY_LENGTH_NEGATIVE
LOCK_STRUCTURE_FULL
LOCKED
LOST_LOCKS
NO_VARIABLE_LENGTH
NOSUSPEND_NOT_RLS
NOTAUTH
PREVIOUS_RLS_FAILURE
READ_NOT_AUTHORISED
READPREV_IN_GENERIC_BROWSE
RECLEN_EXCEEDS_LOGGER_BFSZ
RECORD_BUSY
RECORD_NOT_FOUND
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
UNKNOWN_REQID_READPREV
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE

The return code from the file access method for request.

DUPLICATE_KEY

When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:

NO
YES

FULL_RECORD_ID_LENGTH
The length of the record key.

LENGTH_ERROR_CODE
A value that provides details of a length error that occurred when processing the request.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

FCFR gate, READ_NEXT_UPDATE_SET function

During a file browse, read the next record for updating, and return a pointer to a buffer containing the data.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is

holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION
SECURITY_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_UPD_NOT_RLS
CACHE_FAILURE
DATASET_BEING_COPIED
END_OF_FILE
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
ILLEGAL_KEY_TYPE_CHANGE
IO_ERROR

KEY_LENGTH_NEGATIVE
LOCK_STRUCTURE_FULL
LOCKED
LOST_LOCKS
NOSUSPEND_NOT_RLS
NOTAUTH
PREVIOUS_RLS_FAILURE
READ_NOT_AUTHORISED
READPREV_IN_GENERIC_BROWSE
RECLEN_EXCEEDS_LOGGER_BFSZ
RECORD_BUSY
RECORD_NOT_FOUND
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
UNKNOWN_REQID_READPREV
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

FULL_RECORD_ID_LENGTH
The length of the record key.

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

RECORD_ADDRESS
The address of the target record.

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

FCFR gate, READ_PREVIOUS_INTO function

During a file browse, read the previous record, and return the record into a buffer provided by the caller.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

BUFFER_ADDRESS
The address of the caller's buffer.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

READ_INTEGRITY
Specifies the degree of read integrity for the request.

Values for the parameter are:
CR
FCT_VALUE
NRI
RR

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

BUFFER_LENGTH
Optional Parameter

The length of the caller's buffer.

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

SUPPRESS_LENGERR
Optional Parameter

A binary value that indicates whether length error indications are to be suppressed.

Values for the parameter are:

```
NO
YES
```

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
TABLE_TOKEN_INVALID
```

The following values are returned when RESPONSE is EXCEPTION:

```
BDAM_KEY_CONVERSION
BDAM_READ_PREVIOUS
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
CR_NOT_RLS
END_OF_FILE
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
ILLEGAL_KEY_TYPE_CHANGE
IO_ERROR
ISC_NOT_SUPPORTED
ISCINVREQ
KEY_LENGTH_NEGATIVE
LOCKED
```

NO_VARIABLE_LENGTH
NOSUSPEND_NOT_RLS
NOTAUTH
PREVIOUS_RLS_FAILURE
READ_NOT_AUTHORISED
READPREV_IN_GENERIC_BROWSE
RECORD_BUSY
RECORD_NOT_FOUND
REMOTE_INVREQ
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
RR_NOT_RLS
SELF_DEADLOCK_DETECTED
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
UNKNOWN_REQID_READPREV
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE

The return code from the file access method for request.

DUPLICATE_KEY

When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:

NO
YES

FULL_RECORD_ID_LENGTH

The length of the record key.

LENGTH_ERROR_CODE

A value that provides details of a length error that occurred when processing the request.

Values for the parameter are:

BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

RECORD_LENGTH

Optional Parameter

The length of the record.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST

A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:

NO

YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

FCFR gate, READ_PREVIOUS_SET function

During a file browse, read the previous record, and return a pointer to a buffer containing the data.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

READ_INTEGRITY
Specifies the degree of read integrity for the request.

Values for the parameter are:
CR
FCT_VALUE
NRI
RR

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
TABLE_TOKEN_INVALID
```

The following values are returned when RESPONSE is EXCEPTION:

```
BDAM_KEY_CONVERSION
BDAM_READ_PREVIOUS
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
CR_NOT_RLS
END_OF_FILE
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
ILLEGAL_KEY_TYPE_CHANGE
IO_ERROR
ISC_NOT_SUPPORTED
ISCINVREQ
KEY_LENGTH_NEGATIVE
LOCKED
NOSUSPEND_NOT_RLS
NOTAUTH
PREVIOUS_RLS_FAILURE
READ_NOT_AUTHORISED
READPREV_IN_GENERIC_BROWSE
RECORD_BUSY
RECORD_NOT_FOUND
REMOTE_INVREQ
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
```

RR_NOT_RLS
SELF_DEADLOCK_DETECTED
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
UNKNOWN_REQID_READPREV
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

FULL_RECORD_ID_LENGTH
The length of the record key.

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

RECORD_ADDRESS
The address of the target record.

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, READ_PREVIOUS_UPDATE_INTO function

During a file browse, read the previous record for updating, and return the record into a buffer provided by the caller.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

BUFFER_ADDRESS
The address of the caller's buffer.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

BUFFER_LENGTH
Optional Parameter

The length of the caller's buffer.

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

SUPPRESS_LENGERR
Optional Parameter

A binary value that indicates whether length error indications are to be suppressed.

Values for the parameter are:
NO
YES

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
SECURITY_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_UPD_NOT_RLS
CACHE_FAILURE
DATASET_BEING_COPIED
END_OF_FILE
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
ILLEGAL_KEY_TYPE_CHANGE
IO_ERROR
KEY_LENGTH_NEGATIVE
LOCK_STRUCTURE_FULL
LOCKED
LOST_LOCKS
NO_VARIABLE_LENGTH
NOSUSPEND_NOT_RLS
NOTAUTH
PREVIOUS_RLS_FAILURE
READ_NOT_AUTHORISED
READPREV_IN_GENERIC_BROWSE
RECLEN_EXCEEDS_LOGGER_BFSZ
RECORD_BUSY
RECORD_NOT_FOUND
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
UNKNOWN_REQID_READPREV
VSAM_REQUEST_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:

```
NO
```

YES

FULL_RECORD_ID_LENGTH
The length of the record key.

LENGTH_ERROR_CODE
A value that provides details of a length error that occurred when processing the request.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

FCFR gate, READ_PREVIOUS_UPDATE_SET function

During a file browse, read the previous record for updating, and return a pointer to a buffer containing the data.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is

holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION
SECURITY_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_UPD_NOT_RLS
CACHE_FAILURE
DATASET_BEING_COPIED
END_OF_FILE
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
ILLEGAL_KEY_TYPE_CHANGE
IO_ERROR

KEY_LENGTH_NEGATIVE
LOCK_STRUCTURE_FULL
LOCKED
LOST_LOCKS
NOSUSPEND_NOT_RLS
NOTAUTH
PREVIOUS_RLS_FAILURE
READ_NOT_AUTHORISED
READPREV_IN_GENERIC_BROWSE
RECLEN_EXCEEDS_LOGGER_BFSZ
RECORD_BUSY
RECORD_NOT_FOUND
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
UNKNOWN_REQID_READPREV
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

FULL_RECORD_ID_LENGTH
The length of the record key.

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

RECORD_ADDRESS
The address of the target record.

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

FCFR gate, READ_SET function

Read a record, and return a pointer to a buffer containing the data.

Input Parameters

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

GENERIC
A binary value that specifies whether the search key is a generic key.

Values for the parameter are:
NO
YES

KEY_COMPARISON
A value that specifies whether the search can be satisfied only by a record having the same key as that specified in the record identification field parameter, or by a record having a greater key.

Values for the parameter are:
EQUAL
GTEQ

READ_INTEGRITY
Specifies the degree of read integrity for the request.

Values for the parameter are:
CR
FCT_VALUE
NRI
RR

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:

DEBKEY
DEBREC
KEY
RBA
RRN

BASE_RECORD_ID_ADDRESS
Optional Parameter

The address of the base record identifier.

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
TABLE_TOKEN_INVALID

The following values are returned when RESPONSE is EXCEPTION:

BDAM_KEY_CONVERSION
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
CR_NOT_RLS
FILE_DISABLED
FILE_NOT_OPEN
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
IO_ERROR
ISC_NOT_SUPPORTED
ISCINVREQ
KEY_LENGTH_NEGATIVE
LOADING

LOCKED
LOST_LOCKS
NOSUSPEND_NOT_RLS
NOT_IN_SUBSET
NOTAUTH
PREVIOUS_RLS_FAILURE
RBA_ACCESS_TO_RLS_KSDS
READ_NOT_AUTHORISED
RECORD_BUSY
RECORD_NOT_FOUND
REMOTE_INVREQ
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
RR_NOT_RLS
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIP
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:

NO
YES

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

RECORD_ADDRESS
The address of the target record.

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:

NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:

NO
YES

FCFR gate, READ_UPDATE_INTO function

Read a record for update into a buffer provided by the caller.

Input Parameters

BACKOUT_TYPE

A value that indicates:

- whether the request is for a backout request
- whether the request is to processing a write-add log record or a read-update log record
- for write requests, whether the write is direct or sequential.

Values for the parameter are:

NOT_BACKOUT
READ_UPD
WRITE_DIRECT
WRITE_SEQUENTIAL

BUFFER_ADDRESS

The address of the caller's buffer.

BYPASS_SECURITY_CHECK

A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:

NO
YES

CONDITIONAL

A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:

NO
YES

ENVIRONMENT_IDENTIFIER

A token that identifies the caller's environment.

FILE_NAME

The name of the FILE resource.

GENERIC

A binary value that specifies whether the search key is a generic key.

Values for the parameter are:

NO
YES

KEY_COMPARISON

A value that specifies whether the search can be satisfied only by a record having the same key as that specified in the record identification field parameter, or by a record having a greater key.

Values for the parameter are:

EQUAL
GTEQ

RECORD_ID_ADDRESS

The address of the record identification field.

RECORD_ID_TYPE

The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

TOKEN_REQUEST
A binary value that indicates whether a token is supplied with the request.

Values for the parameter are:
NO
YES

BASE_RECORD_ID_ADDRESS
Optional Parameter

The address of the base record identifier.

BUFFER_LENGTH
Optional Parameter

The length of the caller's buffer.

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

RECORD_LOCK_ONLY
Optional Parameter

A binary value that indicates whether the purpose of the request is solely to lock the record.

Values for the parameter are:
NO
YES

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

SUPPRESS_LENGERR
Optional Parameter

A binary value that indicates whether length error indications are to be suppressed.

Values for the parameter are:
NO
YES

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE

The following values are returned when RESPONSE is EXCEPTION:

BDAM_KEY_CONVERSION
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
DATASET_BEING_COPIED
DEADLOCK_DETECTED
DUPLICATE_READ_UPDATE
FILE_DISABLED
FILE_NOT_OPEN
FILE_NOT_RECOVERABLE
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
IO_ERROR
KEY_LENGTH_NEGATIVE
LOADING
LOCK_STRUCTURE_FULL
LOCKED
LOST_LOCKS
NO_VARIABLE_LENGTH
NOSUSPEND_NOT_RLS
NOT_IN_SUBSET
PREVIOUS_RLS_FAILURE
RBA_ACCESS_TO_RLS_KSDS
READ_NOT_AUTHORISED
RECLEN_EXCEEDS_LOGGER_BFSZ
RECORD_BUSY
RECORD_NOT_FOUND
RESTART_FAILED
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
TOO_MANY_CFDTS_IN_UOW
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

LENGTH_ERROR_CODE
A value that provides details of a length error that occurred when processing the request.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

FCFR gate, READ_UPDATE_SET function

Read a record for updating, and return a pointer to a buffer containing the data.

Input Parameters

BACKOUT_TYPE
A value that indicates:
- whether the request is for a backout request
- whether the request is to processing a write-add log record or a read-update log record
- for write requests, whether the write is direct or sequential.

Values for the parameter are:
NOT_BACKOUT
READ_UPD
WRITE_DIRECT
WRITE_SEQUENTIAL

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

GENERIC
A binary value that specifies whether the search key is a generic key.

Values for the parameter are:
NO
YES

KEY_COMPARISON
A value that specifies whether the search can be satisfied only by a record having the same key as that specified in the record identification field parameter, or by a record having a greater key.

Values for the parameter are:
EQUAL
GTEQ

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

TOKEN_REQUEST
A binary value that indicates whether a token is supplied with the request.

Values for the parameter are:
NO
YES

BASE_RECORD_ID_ADDRESS
Optional Parameter

The address of the base record identifier.

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

RECORD_LOCK_ONLY
Optional Parameter

A binary value that indicates whether the purpose of the request is solely to lock the record.

Values for the parameter are:
NO
YES

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
BDAM_KEY_CONVERSION
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
DATASET_BEING_COPIED
DEADLOCK_DETECTED
DUPLICATE_READ_UPDATE
FILE_DISABLED
FILE_NOT_OPEN
FILE_NOT_RECOVERABLE
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
IO_ERROR
KEY_LENGTH_NEGATIVE
LOADING
LOCK_STRUCTURE_FULL
LOCKED
LOST_LOCKS
NOSUSPEND_NOT_RLS
NOT_IN_SUBSET
PREVIOUS_RLS_FAILURE

RBA_ACCESS_TO_RLS_KSDS
READ_NOT_AUTHORISED
RECLEN_EXCEEDS_LOGGER_BFSZ
RECORD_BUSY
RECORD_NOT_FOUND
RESTART_FAILED
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
TOO_MANY_CFDTS_IN_UOW
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

DUPLICATE_KEY
When the data set is being accessed by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

RECORD_ADDRESS
The address of the target record.

RECORD_LENGTH
Optional Parameter

The length of the record.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

FCFR gate, REPLACE function

Replace a file control record.

Input Parameters

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

RECORD_ADDRESS
The address of the target record.

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

TOKEN_REQUEST
A binary value that indicates whether a token is supplied with the request.

Values for the parameter are:
NO
YES

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

RECORD_LENGTH
Optional Parameter

The length of the record.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:

```
BDAM_LENGTH_CHANGE
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_INVALID_CONTINUATION
CFDT_POOL_FULL
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
CHANGED
DEADLOCK_DETECTED
DUPLICATE_RECORD
INSUFFICIENT_SPACE
IO_ERROR
KEY_STOLEN
LOCK_STRUCTURE_FULL
LOCKED
NO_VARIABLE_LENGTH
NOSUSPEND_NOT_RLS
PREVIOUS_RLS_FAILURE
RECORD_BUSY
RECORD_NOT_FOUND
REPLACE_BEFORE_READ_UPDATE
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
STORE_FAIL
SYSIDERR
TIMEOUT
UPDATE_NOT_AUTHORISED
VSAM_REQUEST_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
INVALID_UPDATE_TOKEN
```

ACCMETH_RETURN_CODE
The return code from the file access method for request.

LENGTH_ERROR_CODE
A value that provides details of a length error that occurred when processing the request.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, REPLACE_DELETE function

Delete and replace a file control record.

Input Parameters

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC

KEY
RBA
RRN

TOKEN_REQUEST
A binary value that indicates whether a token is supplied with the request.

Values for the parameter are:
NO
YES

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
BDAM_LENGTH_CHANGE
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_INVALID_CONTINUATION
CFDT_POOL_FULL
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
CHANGED
DEADLOCK_DETECTED
DELETE_BEFORE_READ_UPDATE
DUPLICATE_RECORD
INSUFFICIENT_SPACE
IO_ERROR
KEY_STOLEN
LOCK_STRUCTURE_FULL
LOCKED
NO_VARIABLE_LENGTH
NOSUSPEND_NOT_RLS
PREVIOUS_RLS_FAILURE
RECORD_BUSY
RECORD_NOT_FOUND

RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
STORE_FAIL
SYSIDERR
TIMEOUT
UPDATE_NOT_AUTHORISED
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_UPDATE_TOKEN

ACCMETH_RETURN_CODE
The return code from the file access method for request.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, RESET_BROWSE function
Reset the position of a browse operation in a file or data table.

Input Parameters
BROWSE_IDENTIFIER
A token that identifies the browse operation.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

GENERIC
A binary value that specifies whether the search key is a generic key.

Values for the parameter are:
NO
YES

KEY_COMPARISON
A value that specifies whether the search can be satisfied only by a record having the same key as that specified in the record identification field parameter, or by a record having a greater key.

Values for the parameter are:
EQUAL
GTEQ

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

BASE_RECORD_ID_ADDRESS
Optional Parameter

The address of the base record identifier.

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
TABLE_TOKEN_INVALID

The following values are returned when RESPONSE is EXCEPTION:
BDAM_KEY_CONVERSION
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR

CFDT_TABLE_GONE
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
ILLEGAL_KEY_TYPE_CHANGE
IO_ERROR
ISC_NOT_SUPPORTED
ISCINVREQ
KEY_LENGTH_NEGATIVE
NOTAUTH
PREVIOUS_RLS_FAILURE
RBA_ACCESS_TO_RLS_KSDS
READ_NOT_AUTHORISED
RECORD_NOT_FOUND
REMOTE_INVREQ
RLS_DISABLED
RLS_FAILURE
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
UNKNOWN_REQID_RESETBR
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE
The return code from the file access method for request.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, RESTART_FILE_CONTROL function

Restart file control's interface with VSAM.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCFR gate, REWRITE function

Rewrite a file record.

Input Parameters

BACKOUT
A binary value that indicates whether the request is issued during transaction backout.

Values for the parameter are:
NO
YES

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

RECORD_ADDRESS
The address of the target record.

TOKEN_REQUEST
A binary value that indicates whether a token is supplied with the request.

Values for the parameter are:
NO
YES

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_LENGTH
Optional Parameter

The length of the record.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:

```
BDAM_LENGTH_CHANGE
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_INVALID_CONTINUATION
CFDT_POOL_FULL
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
CHANGED
DEADLOCK_DETECTED
DUPLICATE_RECORD
INSUFFICIENT_SPACE
IO_ERROR
KEY_STOLEN
LOCK_STRUCTURE_FULL
LOCKED
NO_VARIABLE_LENGTH
NOSUSPEND_NOT_RLS
PREVIOUS_RLS_FAILURE
RECORD_BUSY
RECORD_NOT_FOUND
REWRITE_BEFORE_READ_UPDATE
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
STORE_FAIL
SYSIDERR
TIMEOUT
TOO_MANY_CFDTS_IN_UOW
UPDATE_NOT_AUTHORISED
VSAM_REQUEST_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

INVALID_UPDATE_TOKEN

ACCMETH_RETURN_CODE
The return code from the file access method for request.

LENGTH_ERROR_CODE
A value that provides details of a length error that occurred when processing the request.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, REWRITE_DELETE function

Delete a record and then rewrite it.

Input Parameters

BACKOUT
A binary value that indicates whether the request is issued during transaction backout.

Values for the parameter are:
NO
YES

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

TOKEN_REQUEST
A binary value that indicates whether a token is supplied with the request.

Values for the parameter are:
NO
YES

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
BDAM_DELETE
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_INVALID_CONTINUATION
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
CHANGED
DEADLOCK_DETECTED
DELETE_BEFORE_READ_UPDATE
ESDS_DELETE
IO_ERROR
LOCK_STRUCTURE_FULL
LOCKED
NOSUSPEND_NOT_RLS
PREVIOUS_RLS_FAILURE
RECLEN_EXCEEDS_LOGGER_BFSZ
RECORD_BUSY
RECORD_NOT_FOUND

RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIPPED_SECURITY_FAILURE
STORE_FAIL
SYSIDERR
TIMEOUT
TOO_MANY_CFDTS_IN_UOW
UPDATE_NOT_AUTHORISED
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_UPDATE_TOKEN

ACCMETH_RETURN_CODE
The return code from the file access method for request.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, START_BROWSE function
Start a browse operation.

Input Parameters
BROWSE_IDENTIFIER
A token that identifies the browse operation.

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

GENERIC
A binary value that specifies whether the search key is a generic key.

Values for the parameter are:
NO
YES

KEY_COMPARISON
A value that specifies whether the search can be satisfied only by a record having the same key as that specified in the record identification field parameter, or by a record having a greater key.

Values for the parameter are:
EQUAL
GTEQ

PRIVILEGED_REQUEST
A binary parameter that indicates whether the request is privileged.

Values for the parameter are:
NO
YES

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

BASE_RECORD_ID_ADDRESS
Optional Parameter

The address of the base record identifier.

CFDT_LOAD
Optional Parameter

A binary value that indicates whether the request is part of the browse operation used to read records from the source data set during loading of a coupling facility data table.

Values for the parameter are:
NO
YES

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE
TABLE_TOKEN_INVALID

The following values are returned when RESPONSE is EXCEPTION:

BDAM_KEY_CONVERSION
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
CFDT_TABLE_GONE
DUPLICATE_REQID
FILE_DISABLED
FILE_NOT_OPEN
FILENOTFOUND
FULL_KEY_WRONG_LENGTH
GENERIC_KEY_TOO_LONG
IO_ERROR
ISC_NOT_SUPPORTED
ISCINVREQ
KEY_LENGTH_NEGATIVE
LOADING
NOT_IN_SUBSET
NOTAUTH
PREVIOUS_RLS_FAILURE
RBA_ACCESS_TO_RLS_KSDS
READ_NOT_AUTHORISED
RECORD_NOT_FOUND
REMOTE_INVREQ
RLS_DISABLED
RLS_FAILURE
SERVREQ_VIOLATION
SHIP
SHIPPED_SECURITY_FAILURE
SYSIDERR
TIMEOUT
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE

The return code from the file access method for request.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST

A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:

NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, TEST_FILE_USER function

Determine whether the current task is the user of a file.

Input Parameters

FILE_NAME
The name of the FILE resource.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

FILE_USER
A binary value that indicates whether the current task is the current user of a file.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCFR gate, UNLOCK function

Release the lock on a file record.

Input Parameters

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

TOKEN_REQUEST
A binary value that indicates whether a token is supplied with the request.

Values for the parameter are:
NO
YES

CLEAR_AFTER_ABEND
Optional Parameter

A binary value that indicates whether the request follows a transaction abend, and that the environment must be cleared.

Values for the parameter are:

NO
YES

FCTE_POINTER

Optional Parameter

The address of the file control table entry (FCTE) for the file.

REMOTE_FILE_NAME

Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM

Optional Parameter

The SYSID of the remote system.

UPDATE_TOKEN

Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

WORK_ELEMENT_ADDRESS

Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
DISASTER_PERCOLATION
SECURITY_FAILURE

The following values are returned when RESPONSE is EXCEPTION:

CACHE_FAILURE
CLEAR_ABENDED
IO_ERROR
PREVIOUS_RLS_FAILURE
READ_NOT_AUTHORISED
RLS_DISABLED
RLS_FAILURE
SHIPPED_SECURITY_FAILURE
SYSIDERR
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_UPDATE_TOKEN

ACCMETH_RETURN_CODE

The return code from the file access method for request.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST

A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFR gate, WRITE function

Write to a file.

Input Parameters

BACKOUT
A binary value that indicates whether the request is issued during transaction backout.

Values for the parameter are:
NO
YES

BYPASS_SECURITY_CHECK
A binary value that indicates that security checking can be omitted for the current request.

Values for the parameter are:
NO
YES

CONDITIONAL
A binary value that indicates whether the request should wait if VSAM is holding an active lock against the record, including records locked as the result of a DEADLOCK. **CONDITIONAL(YES)** corresponds to option NOSUSPEND on the CICS API.

Values for the parameter are:
NO
YES

ENVIRONMENT_IDENTIFIER
A token that identifies the caller's environment.

FILE_NAME
The name of the FILE resource.

MASS_INSERT
A binary parameter that specifies whether the WRITE request is part of a mass-insert operation.

Values for the parameter are:
NO
YES

PRIVILEGED_REQUEST
A binary parameter that indicates whether the request is privileged.

Values for the parameter are:
NO
YES

RECORD_ADDRESS
The address of the target record.

RECORD_ID_ADDRESS
The address of the record identification field.

RECORD_ID_TYPE
The type of data contained in the record identification field.

Values for the parameter are:
DEBKEY
DEBREC
KEY
RBA
RRN

CFDT_LOAD
Optional Parameter

A binary value that indicates whether the request is part of the browse operation used to read records from the source data set during loading of a coupling facility data table.

Values for the parameter are:
NO
YES

FCTE_POINTER
Optional Parameter

The address of the file control table entry (FCTE) for the file.

RECORD_ID_LENGTH
Optional Parameter

The length of the record identifier.

RECORD_LENGTH
Optional Parameter

The length of the record.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WORK_ELEMENT_ADDRESS
Optional Parameter

The address of the current file request thread element (FRTE).

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
CFDT_REOPEN_ERROR
DISASTER_PERCOLATION
SECURITY_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
BDAM_KEY_CONVERSION
BDAM_WRITE_MASS_INSERT
CACHE_FAILURE
CFDT_CONNECT_ERROR
CFDT_POOL_FULL
CFDT_POOL_FULL
CFDT_SERVER_NOT_AVAILABLE
CFDT_SERVER_NOT_FOUND

CFDT_SYSIDERR
CFDT_TABLE_GONE
DATASET_BEING_COPIED
DEADLOCK_DETECTED
DUPLICATE_RECORD
FILE_DISABLED
FILE_NOT_OPEN
FULL_KEY_WRONG_LENGTH
INSUFFICIENT_SPACE
IO_ERROR
KEY_LENGTH_NEGATIVE
KEY_STOLEN
LOADING
LOCK_STRUCTURE_FULL
LOCKED
LOST_LOCKS
NO_VARIABLE_LENGTH
NOSUSPEND_NOT_RLS
NOT_IN_SUBSET
PREVIOUS_RLS_FAILURE
RBA_ACCESS_TO_RLS_KSDS
RECORD_BUSY
RECORD_NOT_FOUND
RESTART_FAILED
RIDFLD_KEY_NOT_RECORD_KEY
RLS_DEADLOCK_DETECTED
RLS_DISABLED
RLS_FAILURE
SELF_DEADLOCK_DETECTED
SERVREQ_VIOLATION
SHIP
SHIPPED_SECURITY_FAILURE
STORE_FAIL
SUPPRESSED
SYSIDERR
TABLE_FULL
TIMEOUT
TOO_MANY_CFDTS_IN_UOW
UPDATE_NOT_AUTHORISED
VSAM_REQUEST_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

ACCMETH_RETURN_CODE

The return code from the file access method for request.

LENGTH_ERROR_CODE

A value that provides details of a length error that occurred when processing the request.

Values for the parameter are:

BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REMOTE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCFS gate, CANCEL_CLOSE_FILE function

This function cancels the command to close a file.

Input Parameters

FCTE_POINTER
Optional Parameter

A pointer to the file control table entry (FCTE).

FILE_NAME
Optional Parameter

The 8-character name of the file.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
FILE_NOT_FOUND
FILE_NOT_CLOSING
EXIT_SUPPRESSED_REQUEST

The following values are returned when RESPONSE is DISASTER:
CATALOG_WRITE_FAILED
TM_UNLOCK_FAILED
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
INVALID
EXCEPTION
DISASTER
PURGED

FCFS gate, CLOSE_FILE function

This function closes a named file.

Input Parameters

ACTION
Values for the parameter are:

WAIT
DONT_WAIT
FORCE
FORCE_OTHERS

CFDT_LOAD
Optional Parameter

Values for the parameter are:
YES
NO

CLOSE_QUALIFIER
Optional Parameter

Values for the parameter are:
CLOSE_PENDING
END_LOAD_MODE
END_TABLE_LOAD
CLEAR_IOERROR
SHUTDOWN
IMMEDIATE_CLOSE
IMMEDIATE_CLOSE_PENDING
QUIESCE
END_FAILED_TABLE_LOAD

FCTE_POINTER
Optional Parameter

A pointer to the file control table entry (FCTE).

FILE_NAME
Optional Parameter

The 8-character name of the file.

TABLE_STATS
Optional Parameter

A pointer to the table statistics.

Output Parameters

FCN_RETURN_CODE
The FCN return code.

R15_RETURN_CODE
The R15 return code.

VSAM_RETURN_CODE
The VSAM return code.

REASON
The following value is returned when RESPONSE is INVALID:
INVALID_CLOSE_QUALIFIER

The following values are returned when RESPONSE is EXCEPTION:
FILE_NOT_FOUND
FILE_IN_USE
CLOSE_ERROR
EXIT_SUPPRESSED_REQUEST
DT_DISCONNECT_FAILED
CFDT_CLOSE_ERROR
CFDT_REOPEN_ERROR
CFDT_STATS_ERROR
CFDT_SERVER_ERROR
CFDT_SET_ERROR
CFDT_SYSIDERR

CFDT_TABLE_GONE

The following values are returned when RESPONSE is DISASTER:

CATALOG_WRITE_FAILED
DISPATCHER_WAIT_FAILED
SERIOUS_OPEN_CLOSE_ERROR
DISASTER_PERCOLATION
FCFR_RETURNED_ERROR
TM_UNLOCK_FAILED
ABEND
LOOP
DFHFCQI_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
INVALID
EXCEPTION
DISASTER
PURGED

FCFS gate, DISABLE_FILE function

This function disables a named file and sets its state to unenabled.

Input Parameters

ACTION

Values for the parameter are:

WAIT
DONT_WAIT
FORCE
FORCE_OTHERS

FCTE_POINTER

Optional Parameter

A pointer to the file control table entry (FCTE).

FILE_NAME

Optional Parameter

The 8-character name of the file.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

FILE_NOT_FOUND
FILE_IN_USE
EXIT_SUPPRESSED_REQUEST

The following values are returned when RESPONSE is DISASTER:

CATALOG_WRITE_FAILED
DISPATCHER_WAIT_FAILED
FCFR_RETURNED_ERROR
TM_UNLOCK_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
INVALID
EXCEPTION
DISASTER
PURGED

FCFS gate, ENABLE_FILE function

This function updates files that need to be reset to the enabled state.

Input Parameters

CATALOG_FILE
Specifies whether to catalog the state change. Values for the parameter are:
YES
NO

FCTE_POINTER
Optional Parameter

A pointer to the file control table entry (FCTE).

FILE_NAME
Optional Parameter

The 8-character name of the file.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
FILE_NOT_FOUND
FILE_DISABLING
EXIT_SUPPRESSED_REQUEST

The following values are returned when RESPONSE is DISASTER:
CATALOG_WRITE_FAILED
TM_UNLOCK_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
INVALID
EXCEPTION
DISASTER
PURGED

FCFS gate, OPEN_FILE function

This function opens a named file.

Input Parameters

CURRENT_HIGH_KEY
Optional Parameter

The 16-character string that specifies the current high key.

FCTE_POINTER
Optional Parameter

A pointer to the file control table entry (FCTE).

FILE_NAME
Optional Parameter

The 8-character name of the file.

LOADER_ID
Optional Parameter

The fullword binary field that specifies the ID of the loader.

OPEN_OPTIONS
Optional Parameter

Values for the parameter are:

```
OPEN_BASE
OPEN_FOR_BACKOUT
```

Output Parameters

FCN_RETURN_CODE
The FCN return code.

R15_RETURN_CODE
The R15 return code.

VSAM_RETURN_CODE
The VSAM return code.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
FILE_NOT_FOUND
FILE_DISABLING
OPEN_ERROR
CFDT_OPEN_ERROR
CFDT_NO_DSNAME
CFDT_NOT_KSDS
CFDT_NO_READ_SERVREQS
CFDT_OPEN_MISMATCH
CFDT_SERVER_ERROR
CFDT_SERVER_NOT_AVAILABLE
CFDT_CONNECT_ERROR
CFDT_SERVER_NOT_FOUND
CFDT_SYSIDERR
EXIT_SUPPRESSED_REQUEST
SYSTEM_ID_ERROR
DT_INIT_FAILED
DT_CONNECT_FAILED
DATASET_UNAVAILABLE
DATASET_QUIESCING
DATASET_BEING_COPIED
DATASET_QUIESCED
DATASET_QUIESCED_LOST
RECOVERY_REQUIRED
RLS_NOT_SUPPORTED
COEXISTENCE_ERROR
NO_DSNAME
```

The following values are returned when RESPONSE is DISASTER:

```
CATALOG_WRITE_FAILED
LOADER_ACQUIRE_FAILED
```

LOADER_DEFINE_FAILED
DISPATCHER_WAIT_FAILED
SERIOUS_OPEN_CLOSE_ERROR
FCN_RETURNED_DISASTER
FCM_RETURNED_DISASTER
DISASTER_PERCOLATION
TM_UNLOCK_FAILED
DT_FAILED
DT_INVALID
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
INVALID
EXCEPTION
DISASTER
PURGED

FCIN gate, INITIALISE_FILE_CONTROL function

This function initializes file control and starts the file control restart task.

Input Parameters

None.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
INVALID

FCIN gate, WAIT_FOR_FILE_CONTROL function

Waits for the file control restart task to complete,

Input Parameters

None.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
INVALID

FCLJ gate, BACKOUT_REPLICATION function

This function causes a backout_replication log record to be written to the replication log.

Input Parameters

FRAB_ADDRESS
The address of the FRAB.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LG_RETURNED_ERROR

The following value is returned when RESPONSE is EXCEPTION:
JOURNAL_TOO_SMALL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
INVALID
PURGED
DISASTER
EXCEPTION

FCLJ gate, COMMIT_REPLICATION function

This function causes a commit_replication log record to be written to the replication log.

Input Parameters

FRAB_ADDRESS
The address of the FRAB.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LG_RETURNED_ERROR

The following value is returned when RESPONSE is EXCEPTION:
JOURNAL_TOO_SMALL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
INVALID
PURGED
DISASTER
EXCEPTION

FCLJ gate, DATASET_COPY function

This function is called when DFSMSdss initiates a copy of an RLS data set using the VSAM RLS quiesce mechanism. A tie-up record is written to the log of logs if

the data set is forward recoverable or if autojournalling has been specified in the file definition. If applicable, a record is also written to the forward recovery log.

Input Parameters

FCTE_ADDRESS
The address of the file control table entry for the file associated with a data set being copied.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LG_RETURNED_ERROR
```
The following value is returned when RESPONSE is EXCEPTION:
```
JOURNAL_TOO_SMALL
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
```
OK
DISASTER
EXCEPTION
INVALID
PURGED
```

FCLJ gate, FILE_CLOSE function

This function is called when a file is closed and causes a file_close log record to be written to the forward recovery log, if the file or associated data set, is forward recoverable, or to the autojournal if autojournalling is specified for the file.

Input Parameters

FCTE_ADDRESS
A pointer to the address of the file control table entry for the file being closed.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LG_RETURNED_ERROR
```
The following value is returned when RESPONSE is EXCEPTION:
```
JOURNAL_TOO_SMALL
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
```
OK
DISASTER
EXCEPTION
INVALID
PURGED
```

FCLJ gate, FILE_OPEN function

This function is called when a file is opened and causes a tie-up record to be written to the forward recovery log, if the file or associated data set is forward recoverable, or to the autojournal if autojournalling is specified for the file.

Input Parameters

FCTE_ADDRESS
A pointer to the address of the file control table entry for the file being opened.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LG_RETURNED_ERROR

The following value is returned when RESPONSE is EXCEPTION:
JOURNAL_TOO_SMALL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
PURGED

FCLJ gate, READ_ONLY function

This function causes a read_only log record to be written to an autojournal, if read-only autojournalling is specified on the file definition. The log record is built using the input parameters.

Input Parameters

BASE_ESDS_RBA
The relative byte address (RBA) of the record being read if the file is an extended entry-sequenced data set (ESDS).

FCTE_ADDRESS
The address of the file control table entry for the file being read.

KEY_ADDRESS
The address of the key of the record being read.

KEY_LENGTH
The key length of the record being read.

RECORD_ADDRESS
The address of the record being read.

RECORD_LENGTH
The length of the record being read.

SHUNTED
Indicates whether the unit of work has ever been shunted. Values for the parameter are:
YES
NO

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

LG_RETURNED_ERROR
RM_RETURNED_ERROR

The following value is returned when RESPONSE is EXCEPTION:
JOURNAL_TOO_SMALL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
PURGED

FCLJ gate, READ_UPDATE function

This function causes a read_update log record to be written to the system log if the file is recoverable and if the **DESTINATION** parameter specifies either LOG or BOTH. This function causes a read_update log record to be written to the autojournal if journaling of read updates is specified on the file definition and if the **DESTINATION** parameter specifies either JOURNAL or BOTH. The record is written to the replication log if the file was set to logreplicate.

Input Parameters

BASE_ESDS_RBA
The relative byte address (RBA) of the record being read for update, if the file is an extended entry-sequenced data set (ESDS).

DESTINATION
Specifies whether the log record is to be written to the autojournal, the system log, or both. It is used to suppress writing records that are otherwise requested by the file definition. Values for the parameter are:
JOURNAL
LOG
BOTH

FCTE_ADDRESS
The address of the file control table entry for the file being read for update.

FRAB_ADDRESS
The address of the FRAB.

KEY_ADDRESS
The address of the key of the record being read for update.

KEY_LENGTH
The key length of the record being read for update.

RECORD_ADDRESS
The address of the record being read for update.

RECORD_LENGTH
The length of the record being read for update.

SHUNTED
Indicates whether the unit of work has ever been shunted. Values for the parameter are:
YES
NO

SYNCHRONIZE_LOG
Indicates whether the system log is to be synchronized when the log record is written. Values for the parameter are:
YES
NO

Output Parameters

LOG_TOKEN

Optional Parameter

This parameter is returned if SYNCHRONIZE(NO) was specified. It contains a token to be used when subsequently synchronizing the system log.

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LG_RETURNED_ERROR
RM_RETURNED_ERROR
```

The following value is returned when RESPONSE is EXCEPTION:

```
JOURNAL_TOO_SMALL
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
DISASTER
EXCEPTION
INVALID
PURGED
```

FCLJ gate, SYNCHRONISE_READ_UPDATE function

This function causes any log records previously written to the system log for this file to be synchronized.

Input Parameters

FCTE_ADDRESS

The address of the file control table entry for the file being read for update.

LOG_TOKEN

The token returned on a previous call.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
RM_RETURNED_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
DISASTER
INVALID
PURGED
```

FCLJ gate, TAKE_KEYPOINT function

If BWO copy is supported by this CICS (indicated by a flag in file control static storage), this function performs a scan of the file control table and, unless it has been called within the last half hour, writes a tie-up record for each file open for update in non-RLS mode that is BWO-eligible and forward recoverable to the forward recovery log.

Input Parameters

None.

Output Parameters

KEYPOINT_TAKEN

Indicates whether the set of tie-up records was successfully written. Values for the parameter are:

YES
NO

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LG_RETURNED_ERROR
TM_GETNEXT_FCTE_FAILED

The following value is returned when RESPONSE is EXCEPTION:

JOURNAL_TOO_SMALL

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
DISASTER
EXCEPTION
INVALID
PURGED

FCLJ gate, WRITE_ADD function

This function causes a write_add log record to be written to the system log if the file is recoverable and if the **DESTINATION** parameter specifies BOTH. It causes a write_add log record to be written to the autojournal if journaling of write adds was specified on the file definition. The record is written to the replication log if the file was set to logreplicate.

Input Parameters

BACKOUT

Indicates whether the call is made as part of transaction backout processing. Values for the parameter are:

YES
NO

BASE_ESDS_RBA

The relative byte address (RBA) of the record being added, if the file is an extended entry-sequenced data set (ESDS).

DESTINATION

Specifies whether the log record is to be written to the autojournal only, or to both the autojournal and the system log. It is used to suppress writing records that are otherwise requested by the file definition. Values for the parameter are:

JOURNAL
BOTH

FCTE_ADDRESS

The address of the file control table entry for the file being written to.

FRAB_ADDRESS

The address of the FRAB.

KEY_ADDRESS

The address of the key of the record being added.

KEY_LENGTH
The key length of the record being written to.
MASSINSERT
Indicates whether the record is being added as part of a mass insert. Values for the parameter are:
YES
NO
RECORD_ADDRESS
The address of the record being added.
RECORD_LENGTH
The length of the record being added.
SHUNTED
Indicates whether the unit of work has ever been shunted. Values for the parameter are:
YES
NO

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LG_RETURNED_ERROR
RM_RETURNED_ERROR

The following value is returned when RESPONSE is EXCEPTION:
JOURNAL_TOO_SMALL
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
PURGED

FCLJ gate, WRITE_ADD_COMPLETE function

This function causes a write_add_complete log record to be written to the forward recovery log, if the file or associated data set is forward recoverable, and to the autojournal if write_add_complete journaling is specified on the file definition. The record is written to the replication log if the file was set to logreplicate.

If the file is a recoverable ESDS accessed in non-RLS mode, this function causes a truncated write_add_complete log record to be written to the system log. If the **MASSINSERT** parameter is set to YES and the **MASSINSERT_STAGE** is set to LAST, only the system log record is written and not the forward recovery log or autojournal record.

Input Parameters

BACKOUT
Indicates whether the call is made as part of transaction backout processing. Values for the parameter are:
YES
NO

BASE_ESDS_RBA
The relative byte address (RBA) of the record that was added, if the file is an extended entry-sequenced data set (ESDS).

FCTE_ADDRESS
The address of the file control table entry for the file that was written to.

FRAB_ADDRESS
The address of the FRAB.

KEY_ADDRESS
The address of the key of the record that was added.

KEY_LENGTH
The key length for the file that was written to.

MASSINSERT
Indicates whether the record was added as part of a mass insert. Values for the parameter are:
YES
NO

MASSINSERT_STAGE
Optional Parameter

Indicates whether the record is the first or last record added during a mass insert sequence. Values for the parameter are:
FIRST
LAST

RECORD_ADDRESS
The address of the record that was added.

RECORD_LENGTH
The length of the record that was added.

SHUNTED
Indicates whether the unit of work has ever been shunted. Values for the parameter are:
YES
NO

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LG_RETURNED_ERROR
RM_RETURNED_ERROR

The following value is returned when RESPONSE is EXCEPTION:
JOURNAL_TOO_SMALL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
PURGED

FCLJ gate, WRITE_DELETE function

This function causes a write_delete log record to be written to the forward recovery log, if the file or associated data set is forward recoverable, and to the autojournal if journaling of write_deletes is specified on the file definition. The record is written to the replication log if the file was set to logreplicate.

Input Parameters

BACKOUT
Indicates if the call is made as part of transaction backout processing. Values for the parameter are:
YES
NO

BASE_ESDS_RBA
The relative byte address (RBA) of the record being deleted, if the file is an extended entry-sequenced data set (ESDS).

BASE_KEY_ADDRESS
The address of the base key of the record being deleted, this key is used if the data set is being accessed from a path.

FCTE_ADDRESS
The address of the file control table entry for the file.

FRAB_ADDRESS
The address of the FRAB.

KEY_ADDRESS
The address of the key of the record being deleted.

KEY_LENGTH
The key length for the file.

SHUNTED
Indicates whether the unit of work has ever been shunted. Values for the parameter are:
YES
NO

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LG_RETURNED_ERROR
RM_RETURNED_ERROR

The following value is returned when RESPONSE is EXCEPTION:
JOURNAL_TOO_SMALL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
PURGED

FCLJ gate, WRITE_UPDATE function

This function causes a write_update log record to be written to the forward recovery log, if the file or associated data set is forward recoverable, and to the autojournal if journaling of write updates is specified on the file definition. A write_update log record represents the completion of a file REWRITE request. The record is written to the replication log if the file was set to logreplicate.

Input Parameters

BACKOUT
Indicates whether the call is made as part of transaction backout processing. Values for the parameter are:

YES
NO

BASE_ESDS_RBA
The relative byte address (RBA) of the record being rewritten, if the file is an extended entry-sequenced data set (ESDS).

FCTE_ADDRESS
The address of the file control table entry for the file being rewritten to.

FRAB_ADDRESS
The address of the FRAB.

KEY_ADDRESS
The address of the key of the record being rewritten.

KEY_LENGTH
The key length of the record being rewritten to.

RECORD_ADDRESS
The address of the record being rewritten.

RECORD_LENGTH
The length of the record being rewritten.

SHUNTED
Indicates whether the unit of work has ever been shunted. Values for the parameter are:
YES
NO

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LG_RETURNED_ERROR
RM_RETURNED_ERROR

The following value is returned when RESPONSE is EXCEPTION:
JOURNAL_TOO_SMALL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
EXCEPTION
INVALID
PURGED

FCMT gate, ADD_FILE function

This function builds a new FCT entry for a VSAM file or data table.

Input Parameters

ADD
A binary parameter that indicates whether records can be added to the file.

Values for the parameter are:
NO
YES

BROWSE
A binary parameter that indicates whether records can be retrieved sequentially from the file.

Values for the parameter are:

NO
YES

CATALOG_FCTE
A binary parameter that indicates whether the file definition should be written to the catalog.

Values for the parameter are:
NO
YES

DELETE
A binary parameter that indicates whether records can be deleted from the file.

Values for the parameter are:
NO
YES

DSN_SHARING
Specifies whether VSAM data set name sharing is used for the VSAM file.

Values for the parameter are:
ALL_REQUESTS
MODIFY_REQUESTS

FILE_NAME
The name of the FILE resource.

FILE_TYPE
The location of the file.

Values for the parameter are:
LOCAL
REMOTE

IMAGE
Indicates whether backup images are to be fuzzy or sharp.

Values for the parameter are:
FUZZY
SHARP

JOURNAL_ID
The identifier of the journal used for automatic journaling records.

JOURNAL_READ_ONLY
A binary parameter that indicates whether READ ONLY operations, and not READ UPDATE operations, are to be written to the journal.

Values for the parameter are:
NO
YES

JOURNAL_READ_SYNC
A binary parameter that indicates whether the automatic journaling records that are written for READ operations are to be written synchronously.

Values for the parameter are:
NO
YES

JOURNAL_READ_UPDATE
A binary parameter that indicates whether READ UPDATE operations, and not READ ONLY operations, are to be written to the journal.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_NEW_AFTER
A binary parameter that indicates whether new records are to be written to the journal before they are written to the VSAM file.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_NEW_BEFORE
A binary parameter that indicates whether new records are to be written to the journal after they are written to the VSAM file.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_SYNC
A binary parameter that indicates whether the automatic journaling records that are written for WRITE operations are to be written synchronously.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_UPDATE
A binary parameter that indicates whether the automatic journaling records that are written for REWRITE and DELETE operations are to be written synchronously.

Values for the parameter are:
NO
YES

LSR_POOL_ID
The identity of the local shared resource (LSR) pool.

OPEN_TIME
Specifies whether the file is opened immediately after CICS initialization, or on first reference.

Values for the parameter are:
ASAP
FIRST_REFERENCE

READ
A binary parameter that indicates whether records on this file can be read.

Values for the parameter are:
NO
YES

READ_INTEGRITY
Specifies the level of read integrity required for RLS files.

Values for the parameter are:
CR
NRI
RR

RECORD_FORMAT
The format (fixed- or variable-length) of records on the file.

Values for the parameter are:
FIXED
VARIABLE

RECOVERY
The type of recovery required for the file.

Values for the parameter are:
ALL
BACKOUT_ONLY
NONE

RLS
A binary parameter that indicates whether CICS is to open the file in RLS mode.

Values for the parameter are:
NO
YES

STRING_NUMBER
The number of concurrent requests that can be processed against the file.

UPDATE
A binary parameter that indicates whether records on this file can be updated.

Values for the parameter are:
NO
YES

BASE_NAME
Optional Parameter

The name of the VSAM base cluster.

CF_LOAD
Optional Parameter

A binary parameter that indicates whether a coupling facility data table load is required.

Values for the parameter are:
NO
YES

CF_POOL
Optional Parameter

The name of the coupling facility data table pool containing the table defined by this file definition.

CF_UPDATE_MODEL
Optional Parameter

The type of update model to be used for a coupling facility data table.

Values for the parameter are:
CONTENTION
LOCKING

DATA_BUFFERS
Optional Parameter

The number of buffers to be used for data.

DISPOSITION
Optional Parameter

The disposition of this file.

Values for the parameter are:
OLD
SHARE

DT_NAME
Optional Parameter

The name of the coupling facility data table that is accessed through this file definition.

ENABLE_STATUS
Optional Parameter

Indicates that the initial state of the file is unenabled.

Values for the parameter are:
UNENABLED

FORWARD_RECOVERY_LOG
Optional Parameter

The journal that corresponds to the MVS system logger log stream that is to be used for forward recovery.

INDEX_BUFFERS
Optional Parameter

The number of buffers to be used for the index.

KEY_LENGTH
Optional Parameter

The length in bytes of the logical key of records in remote files, and in coupling facility data tables that are not loaded when they are first loaded.

OBJECT_NAME
Optional Parameter

When the file is associated with a data set that is a VSAM base, the name of the base data set.

RECORD_SIZE
Optional Parameter

The maximum length in bytes of records in a remote file or a coupling facility data table.

REMOTE_NAME
Optional Parameter

The name by which the file is known in the remote region.

REMOTE_SYSTEM
Optional Parameter

The name of the remote system where the file is located.

TABLE_SIZE
Optional Parameter

The maximum number of records (entries) to be accommodated in the data table.

TABLE_TYPE
Optional Parameter

The type of data table.

Values for the parameter are:
CFDT
CICS
NOT_TABLE
USER

VSAM_PASSWORD
Optional Parameter

The VSAM password for the file.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

CATALOG_WRITE_FAILED
CONNECT_DSNB_FAILED
GETMAIN_FAILED
TM_ADD_FAILED
TM_LOCATE_FAILED
TM_UNLOCK_FAILED

The following values are returned when RESPONSE is EXCEPTION:
DUPLICATE_FILE_NAME

The following values are returned when RESPONSE is INVALID:
INVALID_PARAMETERS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCMT gate, COMMIT_FILES function

This function is used during cold start of CICS to catalog all FCT entries in one go using sequential writes to the catalog. This will reduce the number of I/Os incurred writing to the catalog and so improve file control cold start performance.

Input Parameters

TOKEN
A token that identifies the catalog.

TOKEN
A token that identifies the catalog.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
TM_GET_NEXT_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCMT gate, DELETE_FILE function

Delete a file or data table.

Input Parameters

FILE_NAME
The name of the FILE resource.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
CATALOG_DELETE_FAILED
DISCONNECT_DSNB_FAILED
FREEMAIN_FAILED
TM_DELETE_FAILED
TM_QUIESCE_FAILED

The following values are returned when RESPONSE is EXCEPTION:
DO_NOT_REALLOCATE
FCT_ENTRY_IN_USE
FILE_ENABLED
FILE_NAME_NOT_FOUND
FILE_OPEN

The following values are returned when RESPONSE is INVALID:
INVALID_PARAMETERS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCMT gate, END_BROWSE_FILE function

End a browse operation on the set of installed FILE definitions.

Input Parameters

BROWSE_TOKEN
See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
FREEMAIN_FAILED
TM_UNLOCK_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCMT gate, GET_NEXT_FILE function

In a browse operation on the set of installed FILE definitions, return information about the next file.

Input Parameters

BROWSE_TOKEN
See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
TM_GET_NEXT_FAILED

The following values are returned when RESPONSE is EXCEPTION:
END_OF_LIST

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESS_METHOD
Optional Parameter

The access method used for the file.

Values for the parameter are:
BDAM
VSAM

ACTIVE_STRINGS
Optional Parameter

The current number of concurrent requests against the file.

ADD
A binary parameter that indicates whether records can be added to the file.

Values for the parameter are:
NO
YES

BASE_NAME
Optional Parameter

The name of the VSAM base cluster.

BASE_OBJECT_NAME
Optional Parameter

When the file is associated with a data set that is a VSAM base, the name of the base data set.

BLOCK_FORMAT
Optional Parameter

Indicates whether records on the file are blocked or unblocked.

Values for the parameter are:
BLOCKED
UNBLOCKED

BLOCK_KEY_LENGTH
Optional Parameter

The physical block key length for the file.

BLOCK_SIZE
Optional Parameter

The length in bytes of a block. If the blocks are of variable length or are undefined, the value returned is the maximum.

BROWSE
A binary parameter that indicates whether records can be retrieved sequentially from the file.

Values for the parameter are:
NO
YES

CF_LOAD
Optional Parameter

A binary parameter that indicates whether a coupling facility data table load is required.

Values for the parameter are:
NO
YES

CF_POOL
Optional Parameter

The name of the coupling facility data table pool containing the table defined by this file definition.

CF_UPDATE_MODEL
Optional Parameter

The type of update model to be used for a coupling facility data table.

Values for the parameter are:
CONTENTION
LOCKING

DATA_BUFFERS
Optional Parameter

The number of buffers to be used for data.

DELETE
A binary parameter that indicates whether records can be deleted from the file.

Values for the parameter are:
NO
YES

DISPOSITION
Optional Parameter

The disposition of this file.

Values for the parameter are:
OLD
SHARE

DSN_SHARING
Specifies whether VSAM data set name sharing is used for the VSAM file.

Values for the parameter are:
ALL_REQUESTS
MODIFY_REQUESTS

DT_NAME
Optional Parameter

The name of the coupling facility data table that is accessed through this file definition.

EMPTY_STATUS
Optional Parameter

Indicates whether the object associated with this file is to be set to empty when the file is opened.

Values for the parameter are:
EMPTY_REQUESTED
NO_EMPTY_REQUESTED

ENABLE_STATUS
Optional Parameter

Indicates that the initial state of the file is unenabled.

Values for the parameter are:
UNENABLED

EXCLUSIVE_CONTROL
Optional Parameter

A binary value that indicates whether records on this file are to be placed under exclusive control when a read for update is issued.

Values for the parameter are:
NO
YES

FILE_NAME
The name of the FILE resource.

FILE_TYPE
The location of the file.

Values for the parameter are:
LOCAL
REMOTE

FORWARD_RECOVERY_LOG
Optional Parameter

The journal that corresponds to the MVS system logger log stream that is to be used for forward recovery.

INDEX_BUFFERS
Optional Parameter

The number of buffers to be used for the index.

JOURNAL_ID
The identifier of the journal used for automatic journaling records.

JOURNAL_READ_ONLY
A binary parameter that indicates whether READ ONLY operations, and not READ UPDATE operations, are to be written to the journal.

Values for the parameter are:
NO
YES

JOURNAL_READ_SYNC
A binary parameter that indicates whether the automatic journaling records that are written for READ operations are to be written synchronously.

Values for the parameter are:
NO
YES

JOURNAL_READ_UPDATE
A binary parameter that indicates whether READ UPDATE operations, and not READ ONLY operations, are to be written to the journal.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_NEW_AFTER
A binary parameter that indicates whether new records are to be written to the journal before they are written to the VSAM file.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_NEW_BEFORE
A binary parameter that indicates whether new records are to be written to the journal after they are written to the VSAM file.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_SYNC
A binary parameter that indicates whether the automatic journaling records that are written for WRITE operations are to be written synchronously.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_UPDATE
A binary parameter that indicates whether the automatic journaling records that are written for REWRITE and DELETE operations are to be written synchronously.

Values for the parameter are:
NO

YES

KEY_LENGTH
Optional Parameter

The length in bytes of the logical key of records in remote files, and in coupling facility data tables that are not loaded when they are first loaded.

KEY_POSITION
Optional Parameter

The starting position of the key field in each record relative to the beginning of the record.

LSR_POOL_ID
The identity of the local shared resource (LSR) pool.

OBJECT
Optional Parameter

Indicates whether the file is associated with a data set (a VSAM KSDS, ESDS, or RRDS, or an alternate index used directly) or a VSAM path that links an alternate index to its base cluster.

Values for the parameter are:
BASE
PATH

OBJECT_NAME
Optional Parameter

When the file is associated with a data set that is a VSAM base, the name of the base data set.

OPEN_STATUS
Optional Parameter

Indicates whether the file is open, closed, or in a transitional state.

Values for the parameter are:
CLOSED
CLOSING
OPEN
OPENING

READ
A binary parameter that indicates whether records on this file can be read.

Values for the parameter are:
NO
YES

READ_INTEGRITY
Specifies the level of read integrity required for RLS files.

Values for the parameter are:
CR
NRI
RR

RECORD_FORMAT
The format (fixed- or variable-length) of records on the file.

Values for the parameter are:
FIXED
VARIABLE

RECORD_SIZE
Optional Parameter

The maximum length in bytes of records in a remote file or a coupling facility data table.

RECOVERY

The type of recovery required for the file.

Values for the parameter are:

ALL
BACKOUT_ONLY
NONE

RELATIVE_ADDR

Optional Parameter

Indicating whether relative or absolute addressing is used to access the file and the type of relative addressing.

Values for the parameter are:

BLOCK
DECIMAL
HEX
NONE

REUSE

Optional Parameter

This parameter is no longer used.

Values for the parameter are:

NO
YES

RLS

A binary parameter that indicates whether CICS is to open the file in RLS mode.

Values for the parameter are:

NO
YES

STRING_NUMBER

The number of concurrent requests that can be processed against the file.

TABLE_SIZE

Optional Parameter

The maximum number of records (entries) to be accommodated in the data table.

TABLE_TYPE

Optional Parameter

The type of data table.

Values for the parameter are:

CFDT
CICS
NOT_TABLE
USER

TYPE

Optional Parameter

The type of data set that corresponds to the file

Values for the parameter are:

ESDS
KEYED
KSDS

NOT_KEYED
RRDS
VRRDS

UPDATE
A binary parameter that indicates whether records on this file can be updated.

Values for the parameter are:
NO
YES

USING_LSR
Optional Parameter

A binary value that indicates if the file is using a local shared resource (LSR) pool.

Values for the parameter are:
NO
YES

VSAM_PASSWORD
Optional Parameter

The VSAM password for the file.

FCMT gate, INQUIRE_FILE function

Return information about the current state of a FILE resource.

Input Parameters

FILE_NAME
The name of the FILE resource.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
TM_LOCATE_FAILED
TM_UNLOCK_FAILED

The following values are returned when RESPONSE is EXCEPTION:
FILE_NAME_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_PARAMETERS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESS_METHOD
Optional Parameter

The access method used for the file.

Values for the parameter are:
BDAM
VSAM

ACTIVE_STRINGS
Optional Parameter

The current number of concurrent requests against the file.

ADD
A binary parameter that indicates whether records can be added to the file.

Values for the parameter are:
NO

YES

BASE_NAME
Optional Parameter

The name of the VSAM base cluster.

BASE_OBJECT_NAME
Optional Parameter

When the file is associated with a data set that is a VSAM base, the name of the base data set.

BLOCK_FORMAT
Optional Parameter

Indicates whether records on the file are blocked or unblocked.

Values for the parameter are:
BLOCKED
UNBLOCKED

BLOCK_KEY_LENGTH
Optional Parameter

The physical block key length for the file.

BLOCK_SIZE
Optional Parameter

The length in bytes of a block. If the blocks are of variable length or are undefined, the value returned is the maximum.

BROWSE
A binary parameter that indicates whether records can be retrieved sequentially from the file.

Values for the parameter are:
NO
YES

CF_LOAD
Optional Parameter

A binary parameter that indicates whether a coupling facility data table load is required.

Values for the parameter are:
NO
YES

CF_POOL
Optional Parameter

The name of the coupling facility data table pool containing the table defined by this file definition.

CF_UPDATE_MODEL
Optional Parameter

The type of update model to be used for a coupling facility data table.

Values for the parameter are:
CONTENTION
LOCKING

DATA_BUFFERS
Optional Parameter

The number of buffers to be used for data.

DELETE
A binary parameter that indicates whether records can be deleted from the file.

Values for the parameter are:
NO
YES

DISPOSITION
Optional Parameter

The disposition of this file.

Values for the parameter are:
OLD
SHARE

DSN_SHARING
Specifies whether VSAM data set name sharing is used for the VSAM file.

Values for the parameter are:
ALL_REQUESTS
MODIFY_REQUESTS

DT_NAME
Optional Parameter

The name of the coupling facility data table that is accessed through this file definition.

EMPTY_STATUS
Optional Parameter

Indicates whether the object associated with this file is to be set to empty when the file is opened.

Values for the parameter are:
EMPTY_REQUESTED
NO_EMPTY_REQUESTED

ENABLE_STATUS
Optional Parameter

Indicates that the initial state of the file is unenabled.

Values for the parameter are:
UNENABLED

EXCLUSIVE_CONTROL
Optional Parameter

A binary value that indicates whether records on this file are to be placed under exclusive control when a read for update is issued.

Values for the parameter are:
NO
YES

FILE_NAME
The name of the FILE resource.

FILE_TYPE
The location of the file.

Values for the parameter are:
LOCAL
REMOTE

FORWARD_RECOVERY_LOG
Optional Parameter

The journal that corresponds to the MVS system logger log stream that is to be used for forward recovery.

INDEX_BUFFERS
Optional Parameter

The number of buffers to be used for the index.

JOURNAL_ID
The identifier of the journal used for automatic journaling records.

JOURNAL_READ_ONLY
A binary parameter that indicates whether READ ONLY operations, and not READ UPDATE operations, are to be written to the journal.

Values for the parameter are:
NO
YES

JOURNAL_READ_SYNC
A binary parameter that indicates whether the automatic journaling records that are written for READ operations are to be written synchronously.

Values for the parameter are:
NO
YES

JOURNAL_READ_UPDATE
A binary parameter that indicates whether READ UPDATE operations, and not READ ONLY operations, are to be written to the journal.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_NEW_AFTER
A binary parameter that indicates whether new records are to be written to the journal before they are written to the VSAM file.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_NEW_BEFORE
A binary parameter that indicates whether new records are to be written to the journal after they are written to the VSAM file.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_SYNC
A binary parameter that indicates whether the automatic journaling records that are written for WRITE operations are to be written synchronously.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_UPDATE
A binary parameter that indicates whether the automatic journaling records that are written for REWRITE and DELETE operations are to be written synchronously.

Values for the parameter are:
NO
YES

KEY_LENGTH
Optional Parameter

The length in bytes of the logical key of records in remote files, and in coupling facility data tables that are not loaded when they are first loaded.

KEY_POSITION
Optional Parameter

The starting position of the key field in each record relative to the beginning of the record.

LSR_POOL_ID
The identity of the local shared resource (LSR) pool.

OBJECT
Optional Parameter

Indicates whether the file is associated with a data set (a VSAM KSDS, ESDS, or RRDS, or an alternate index used directly) or a VSAM path that links an alternate index to its base cluster.

Values for the parameter are:
- BASE
- PATH

OBJECT_NAME
Optional Parameter

When the file is associated with a data set that is a VSAM base, the name of the base data set.

OPEN_STATUS
Optional Parameter

Indicates whether the file is open, closed, or in a transitional state.

Values for the parameter are:
- CLOSED
- CLOSING
- OPEN
- OPENING

READ
A binary parameter that indicates whether records on this file can be read.

Values for the parameter are:
- NO
- YES

READ_INTEGRITY
Specifies the level of read integrity required for RLS files.

Values for the parameter are:
- CR
- NRI
- RR

RECORD_FORMAT
The format (fixed- or variable-length) of records on the file.

Values for the parameter are:
- FIXED
- VARIABLE

RECORD_SIZE
Optional Parameter

The maximum length in bytes of records in a remote file or a coupling facility data table.

RECOVERY
The type of recovery required for the file.

Values for the parameter are:
- ALL
- BACKOUT_ONLY
- NONE

RELATIVE_ADDR
Optional Parameter

Indicating whether relative or absolute addressing is used to access the file and the type of relative addressing.

Values for the parameter are:
- BLOCK
- DECIMAL
- HEX
- NONE

REMOTE_NAME
Optional Parameter

The name by which the file is known in the remote region.

REMOTE_SYSTEM
Optional Parameter

The name of the remote system where the file is located.

RLS
A binary parameter that indicates whether CICS is to open the file in RLS mode.

Values for the parameter are:
- NO
- YES

STRING_NUMBER
The number of concurrent requests that can be processed against the file.

TABLE_SIZE
Optional Parameter

The maximum number of records (entries) to be accommodated in the data table.

TABLE_TYPE
Optional Parameter

The type of data table.

Values for the parameter are:
- CFDT
- CICS
- NOT_TABLE
- USER

TYPE
Optional Parameter

The type of data set that corresponds to the file

Values for the parameter are:
- ESDS
- KEYED
- KSDS
- NOT_KEYED
- RRDS
- VRRDS

UPDATE
A binary parameter that indicates whether records on this file can be updated.

Values for the parameter are:
- NO
- YES

USING_LSR
Optional Parameter

A binary value that indicates if the file is using a local shared resource (LSR) pool.

Values for the parameter are:
NO
YES

VSAM_PASSWORD
Optional Parameter

The VSAM password for the file.

FCMT gate, START_BROWSE_FILE function

STart a browse operation on installed FILE definitions.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
GETMAIN_FAILED

BROWSE_TOKEN
See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCMT gate, UPDATE_FILE function

Update the attributes of an installed FILE definition.

Input Parameters

FILE_NAME
The name of the FILE resource.

ADD
A binary parameter that indicates whether records can be added to the file.

Values for the parameter are:
NO
YES

BASE_NAME
Optional Parameter

The name of the VSAM base cluster.

BROWSE
A binary parameter that indicates whether records can be retrieved sequentially from the file.

Values for the parameter are:
NO
YES

CF_LOAD
Optional Parameter

A binary parameter that indicates whether a coupling facility data table load is required.

Values for the parameter are:
NO
YES

CF_POOL
Optional Parameter

The name of the coupling facility data table pool containing the table defined by this file definition.

CF_UPDATE_MODEL
Optional Parameter

The type of update model to be used for a coupling facility data table.

Values for the parameter are:
CONTENTION
LOCKING

DATA_BUFFERS
Optional Parameter

The number of buffers to be used for data.

DELETE
A binary parameter that indicates whether records can be deleted from the file.

Values for the parameter are:
NO
YES

DISPOSITION
Optional Parameter

The disposition of this file.

Values for the parameter are:
OLD
SHARE

DSN_SHARING
Specifies whether VSAM data set name sharing is used for the VSAM file.

Values for the parameter are:
ALL_REQUESTS
MODIFY_REQUESTS

DT_NAME
Optional Parameter

The name of the coupling facility data table that is accessed through this file definition.

FORWARD_RECOVERY_LOG
Optional Parameter

The journal that corresponds to the MVS system logger log stream that is to be used for forward recovery.

EMPTY_STATUS
Optional Parameter

Indicates whether the object associated with this file is to be set to empty when the file is opened.

Values for the parameter are:
EMPTY_REQUESTED
NO_EMPTY_REQUESTED

ENABLE_STATUS
Optional Parameter

Indicates that the initial state of the file is unenabled.

Values for the parameter are:
UNENABLED

EXCLUSIVE_CONTROL
Optional Parameter

A binary value that indicates whether records on this file are to be placed under exclusive control when a read for update is issued.

Values for the parameter are:
NO
YES

FILE_TYPE
The location of the file.

Values for the parameter are:
LOCAL
REMOTE

FORWARD_RECOVERY_LOG
Optional Parameter

The journal that corresponds to the MVS system logger log stream that is to be used for forward recovery.

INDEX_BUFFERS
Optional Parameter

The number of buffers to be used for the index.

JOURNAL_ID
The identifier of the journal used for automatic journaling records.

JOURNAL_READ_ONLY
A binary parameter that indicates whether READ ONLY operations, and not READ UPDATE operations, are to be written to the journal.

Values for the parameter are:
NO
YES

JOURNAL_READ_SYNC
A binary parameter that indicates whether the automatic journaling records that are written for READ operations are to be written synchronously.

Values for the parameter are:
NO
YES

JOURNAL_READ_UPDATE
A binary parameter that indicates whether READ UPDATE operations, and not READ ONLY operations, are to be written to the journal.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_NEW_AFTER
A binary parameter that indicates whether new records are to be written to the journal before they are written to the VSAM file.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_NEW_BEFORE
A binary parameter that indicates whether new records are to be written to the journal after they are written to the VSAM file.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_SYNC
A binary parameter that indicates whether the automatic journaling records that are written for WRITE operations are to be written synchronously.

Values for the parameter are:
NO
YES

JOURNAL_WRITE_UPDATE
A binary parameter that indicates whether the automatic journaling records that are written for REWRITE and DELETE operations are to be written synchronously.

Values for the parameter are:
NO
YES

KEY_LENGTH
Optional Parameter

The length in bytes of the logical key of records in remote files, and in coupling facility data tables that are not loaded when they are first loaded.

LSR_POOL_ID
The identity of the local shared resource (LSR) pool.

OBJECT_NAME
Optional Parameter

When the file is associated with a data set that is a VSAM base, the name of the base data set.

OPEN_TIME
Specifies whether the file is opened immediately after CICS initialization, or on first reference.

Values for the parameter are:
ASAP
FIRST_REFERENCE

READ
A binary parameter that indicates whether records on this file can be read.

Values for the parameter are:
NO
YES

READ_INTEGRITY
Specifies the level of read integrity required for RLS files.

Values for the parameter are:
CR
NRI
RR

RECORD_FORMAT
The format (fixed- or variable-length) of records on the file.

Values for the parameter are:
FIXED
VARIABLE

RECORD_SIZE
Optional Parameter

The maximum length in bytes of records in a remote file or a coupling facility data table.

RECOVERY
The type of recovery required for the file.

Values for the parameter are:
ALL
BACKOUT_ONLY
NONE

RLS
A binary parameter that indicates whether CICS is to open the file in RLS mode.

Values for the parameter are:
NO
YES

STRING_NUMBER
The number of concurrent requests that can be processed against the file.

TABLE_SIZE
Optional Parameter

The maximum number of records (entries) to be accommodated in the data table.

TABLE_TYPE
Optional Parameter

The type of data table.

Values for the parameter are:
CFDT
CICS
NOT_TABLE
USER

UPDATE
A binary parameter that indicates whether records on this file can be updated.

Values for the parameter are:
NO
YES

VSAM_PASSWORD
Optional Parameter

The VSAM password for the file.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
CATALOG_WRITE_FAILED
CONNECT_DSNB_FAILED
TM_LOCATE_FAILED
TM_UNLOCK_FAILED

The following values are returned when RESPONSE is EXCEPTION:
DO_NOT_REALLOCATE
FILE_ENABLED
FILE_NAME_NOT_FOUND
FILE_OPEN

The following values are returned when RESPONSE is INVALID:
INVALID_PARAMETERS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FCQI gate, COMPLETE_QUIESCE function

This function issues the IDAQUIES QUICMP macro to SMSVSAM.

When CICS has completed processing a VSAM QUICLOSE (quiesce), QIOCOPY (non-BWO backup), or QUIBWO (BWO backup) request, SMSVSAM must be notified with an IDAQUIES QUICMP .

Input Parameters

DSNAME

The 44-character name of the base data set that has had quiesce processing completed by CICS.

QUIESCE_TOKEN

The token that was supplied by SMSVSAM when it drove the quiesce exit for the original quiesce request. This token must be returned on the IDAQUIES QUICMP.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
IOERR
SERVER_FAILURE
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCQI gate, INITIATE_QUIESCE function

This function takes a quiesce request and creates a file control quiesce send element (FCQSE) to describe the request.

Input Parameters

BUSY

Indicates whether DFHFCQI is to wait for the quiesce to complete, or is to return immediately to the caller. Values for the parameter are:

```
WAIT
NOWAIT
```

DSNAME

The 44-character name of the base data set to be quiesced.

SOURCE

Indicates whether the source of the quiesce request was CICS or a user. Values for the parameter are:

```
CICS
USER
```

QUIESCE_TYPE

The type of quiesce being initiated. Values for the parameter are:

```
QUIESCE
```

IMMQUIESCE
UNQUIESCE
NONBWO_CANCEL
BWO_CANCEL
QUIESCE_CANCEL

Output Parameters

REASON

The following value is returned when RESPONSE is INVALID:

INVALID_QUIESCE_TYPE

The following values are returned when RESPONSE is EXCEPTION:

NOT_SUPPORTED
UNKNOWN_VSAM_DATASET
QUIESCE_NOT_POSSIBLE
UNQUIESCE_NOT_POSSIBLE
CANCELLED
TIMED_OUT
IOERR
SERVER_FAILURE
DATASET_MIGRATED

The following values are returned when RESPONSE is DISASTER:

ABEND
CATALOG_ERROR
DISASTER_PERCOLATION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCQI gate, INQUIRE_QUIESCE function

This function returns the quiesce state of a data set as QUIESCED, UNQUIESCED, or QUIESCING.

Input Parameters

DSNAME

The 44-character name of the base data set on which the quiesce state information is being inquired.

Output Parameters

QUIESCESTATE

Indicates the quiesce state of the data set. Values for the parameter are:

QUIESCED
UNQUIESCED
QUIESCING

REASON

The following values are returned when RESPONSE is EXCEPTION:

NOT_SUPPORTED
UNKNOWN_VSAM_DATASET
IOERR

The following values are returned when RESPONSE is DISASTER:

ABEND
CATALOG_ERROR
DISASTER_PERCOLATION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCQR gate, RECEIVE_QUIESCES function

This function receives quiesce requests and calls the PROCESS_QUIESCE function.

This function consists of a forever loop around a dispatcher wait on an event control block (ECB). It receives work from the CICS RLS quiesce exit DFHFCQX whenever SMSVSAM requires CICS to perform processing for a quiesce request. DFHFCQX queues the request to DFHFCQR by adding an FC Quiesce Receive Element (FCQRE) to a chain anchored in file control static storage and by posting the ECB associated with the chain, also in FC static.

The posting of the ECB initiates the CFQR transaction, which runs the code in DFHFCQR. The FCQREs on the chain are processed, and DFHFCQU is called with function PROCESS_QUIESCE to perform the work. The ECB might also be posted to inform DFHFCQR that CICS is stopping. When DFHFCQU has finished processing, DFHFCQR unchains and frees the FCQRE.

Input Parameters

None.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
PROCESS_QUIESCE_ERROR
DISASTER_PERCOLATION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCQS gate, SEND_QUIESCES function

This function sends a quiesce request to SMSVSAM.

This function consists of a forever loop around a dispatcher wait on a list of event control blocks (ECBs). Work is received from tasks that want to send a quiesce request to SMSVSAM. Such tasks call DFHFCQI with function INITIATE_QUIESCE, which queues the request to DFHFCQS by adding an FC Quiesce Send Element (FCQSE) to the chain anchored in file control static storage and by posting an ECB associated with the chain, also in FC static.

When the ECB is posted, it initiates the CFQS transaction, which runs the code in DFHFCQS. The FCQSEs on the chain are processed, and DFHFCCA is called with function QUIESCE_REQUEST to issue the appropriate type of IDAQUIES macro to SMSVSAM. This operation is asynchronous and SMSVSAM returns the address of an ECB that will be posted when the IDAQUIES completes. The addresses returned by SMSVSAM are saved in the FCQSE.

DFHFCQS then returns to its dispatcher wait for a list of ECBs, the ECB for the chain plus an ECB for each IDAQUIES request. It starts and processes the chain whenever one of these ECBs is posted. The wait also specifies a timeout interval so that IDAQUIES requests that are in an endless loop can be detected. When DFHFCQS starts up, there might be new work on the chain, or a quiesce request has completed, or a quiesce request has timed out, or CICS is stopping. When a quiesce request has completed or timed out, DFHFCQS will resume the initiating task if it is waiting, after issuing appropriate messages and calling global user exit XFCQUIS if active.

Input Parameters

None.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
TIMEOUT_CANCEL_ERROR
DISASTER_PERCOLATION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCQU gate, PROCESS_QUIESCE function

This function is called when a quiesce request is received from VSAM RLS. The quiesce exit, DFHFCQX, queues requests to the CFQR system transaction, DFHFCQR, which calls DFHFCQU to process each one in turn. The PROCESS_QUIESCE function is also called to implement a non-RLS variant of QUIESCE called NON_RLS_CLOSE.

Input Parameters

QUIESCE_TYPE

The type of quiesce being requested. Values for the parameter are:

QUIESCE
Corresponds to an SMSVSAM QUICLOSE. All files open against the data set are closed, the file state of each file is set to unenabled with a flag that says reenable on QUIOPEN, and a QUICMP is issued for the QUICLOSE back to VSAM RLS to indicate that QUICLOSE processing is complete. The **IMMEDIATE** parameter governs how file closes are to be performed. If the **IMMEDIATE** parameter is set to NO, or omitted, files will be closed when all UOWs using the data set have completed normally. If the **IMMEDIATE** parameter is set to YES, all such UOWs will be force purged to speed up file closure.

UNQUIESCE
Corresponds to an SMSVSAM QUIOPEN. All files associated with the data set are checked to see if their file state requires resetting back to enabled because it had been set unenabled by a QUICLOSE.

NONBWO_START
Corresponds to an SMSVSAM QUICOPY. CICS prepares for a non-BWO backup of the data set by preventing new units of work from updating the data set, allowing existing UOWs to finish updating the data set, and issuing a QUICMP for the QUICOPY back to SMSVSAM to indicate that QUICOPY processing is complete. The files involved are not closed.

NONBWO_END
Corresponds to an SMSVSAM QUICEND. All files associated with the data set are checked to see if their file state requires resetting to enabled because it had been set unenabled by an OPEN failure, and a set of 'tie up records' is written for the data set.

BWO_START
Corresponds to an SMSVSAM QUIBWO. CICS prepares for a BWO backup of the data set by writing a set of 'tie up records' allowing existing units of work to finish updating the data set and issuing a QUICMP for the QUIBWO back to SMSVSAM to indicate that QUIBWO processing is complete. The files involved are not closed, and updates are not prevented.

BWO_END
Corresponds to an SMSVSAM QUIBEND. The only processing involved is to stop an existing BWO quiesce if one is in progress.

LOCKS_RECOVERY_COMPLETE
Corresponds to an SMSVSAM QUILLRC. CICS is notified that lost locks recovery has been completed for the data set throughout the sysplex. DFHFCRR is called with the LOST_LOCKS_RECOVERED function to process the availability of the data set.

FORWARD_RECOVERY_COMPLETE
Corresponds to an SMSVSAM QUIFRC. CICS is notified that forward recovery has been completed for the data set. DFHFCRR is called with the RESOURCE_AVAILABLE function to process the availability of the data set.

CACHE_AVAILABLE
Corresponds to an SMSVSAM QUICA. CICS is notified that a previously failed cache structure is now available. DFHFCRR is called with the RESOURCE_AVAILABLE function to process the availability of the cache.

NON_RLS_CLOSE
Processes a non-RLS variant of type CLOSE called NON_RLS_CLOSE. All ACBs open against the specified non-RLS data set are closed. NON_RLS_CLOSE is used internally by CICS and does not run under the CFQR system transaction. Each quiesce request type is processed in a different way by DFHFCQU.

DSNAME|CACHE_NAME
Specifies either the 44-character name of the data set to which the quiesce

request applies, or when the value of the **QUIESCE_TYPE** parameter is CACHE_AVAILABLE, the 16-character name of the cache structure that has become available.

IMMEDIATE
This parameter applies only when the value of the **QUIESCE_TYPE** parameter is QUIESCE or NON_RLS_CLOSE, and indicates whether units of work that have updated the data set will be forced to complete immediately, or whether the request will wait for those units of work to complete naturally. Values for the parameter are:
YES
NO

CONCURRENT
This parameter applies only when the value of the **QUIESCE_TYPE** parameter is NONBWO_START or BWO_START, This parameter indicates whether the concurrent copy technique is being used and has no effect on the processing. Values for the parameter are:
YES
NO

QUIESCE_TOKEN
This token is supplied by SMSVSAM when certain quiesce requests are initiated and must be passed back when the quiesce complete is issued.

Output Parameters

REASON
The following value is returned when RESPONSE is INVALID:
INVALID_QUIESCE_TYPE

The following value is returned when RESPONSE is EXCEPTION:
DSNB_NOT_FOUND

The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION
DFHFCRR_ERROR
DFHFCQI_ERROR
DFHFCFS_ERROR
DFHTM_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCRF gate, BROWSE function

Browse a file in a remote system.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

READ_BUFFER
The buffer that receives the file record data that is returned from the remote system.

RECORD_ID
Optional Parameter

The record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.

BACKEND_DUPLICATE_KEY
When the data set is being accessed in the remote system by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:

```
NO
YES
```

BACKEND_LENGTH_ERR_CODE
A value that provides details of a length error that occurred when processing the request in the remote system.

Values for the parameter are:

```
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE
```

BACKEND_REASON
The reason code from the file control request in the remote system.

BACKEND_RESPONSE
The response code from the file control request in the remote system.

FULL_RECORD_ID_LENGTH
The length of the record key.

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:

```
NO
```

YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

FCRF gate, DELETE function

Delete a record from a file in a remote system.

Input Parameters

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

RECORD_ID
Optional Parameter

The record identifier.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.

BACKEND_DUPLICATE_KEY
When the data set is being accessed in the remote system by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

BACKEND_REASON
The reason code from the file control request in the remote system.

BACKEND_RESPONSE
The response code from the file control request in the remote system.

DELETED_RECORD_COUNT
The number of records deleted by the request.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCRF gate, END_BROWSE function

End a browse operation on a remote file.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.

BACKEND_REASON
The reason code from the file control request in the remote system.

BACKEND_RESPONSE
The response code from the file control request in the remote system.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCRF gate, READ function

Read a remote file.

Input Parameters

READ_BUFFER
The buffer that receives the file record data that is returned from the remote system.

RECORD_ID
Optional Parameter

The record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.

BACKEND_DUPLICATE_KEY
When the data set is being accessed in the remote system by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

BACKEND_LENGTH_ERR_CODE
A value that provides details of a length error that occurred when processing the request in the remote system.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

BACKEND_REASON
The reason code from the file control request in the remote system.

BACKEND_RESPONSE
The response code from the file control request in the remote system.

MAXIMUM_RECORD_LENGTH
The length of the longest record in the data set.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

FCRF gate, REPLACE function

Replace a file record in a remote system.

Input Parameters

RECORD_ID
Optional Parameter

The record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WRITE_RECORD
The record to be written in the remote system.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.

BACKEND_LENGTH_ERR_CODE
A value that provides details of a length error that occurred when processing the request in the remote system.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

BACKEND_REASON
The reason code from the file control request in the remote system.

BACKEND_RESPONSE
The response code from the file control request in the remote system.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCRF gate, REPLACE_DELETE function

Delete and replace a file control record in a remote system.

Input Parameters

RECORD_ID
Optional Parameter

The record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.

BACKEND_DUPLICATE_KEY
When the data set is being accessed in the remote system by way of an alternate index path that allows non-unique alternate keys, a binary value that indicates whether further records exist with the same alternate key.

Values for the parameter are:
NO
YES

BACKEND_REASON
The reason code from the file control request in the remote system.

BACKEND_RESPONSE
The response code from the file control request in the remote system.

DELETED_RECORD_COUNT
The number of records deleted by the request.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCRF gate, RESET_BROWSE function

Reset the start of a browse operation on a remote file.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

RECORD_ID
Optional Parameter

The record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.
BACKEND_REASON
The reason code from the file control request in the remote system.
BACKEND_RESPONSE
The response code from the file control request in the remote system.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES
TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCRF gate, REWRITE function
Rewrite a record in a remote file.

Input Parameters
REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.
REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.
WRITE_RECORD
The record to be written in the remote system.
UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

Output Parameters
REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.
BACKEND_LENGTH_ERR_CODE
A value that provides details of a length error that occurred when processing the request in the remote system.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN

BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

BACKEND_REASON
The reason code from the file control request in the remote system.

BACKEND_RESPONSE
The response code from the file control request in the remote system.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCRF gate, START_BROWSE function

Start a browse operation on a remote file.

Input Parameters

BROWSE_IDENTIFIER
A token that identifies the browse operation.

RECORD_ID
Optional Parameter

The record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.

BACKEND_REASON
The reason code from the file control request in the remote system.

BACKEND_RESPONSE
The response code from the file control request in the remote system.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCRF gate, UNLOCK function

Unlock a file record in a remote system.

Input Parameters

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

UPDATE_TOKEN
Optional Parameter

A token that identifies an update request, and allows subsequent requests to refer to it.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.

BACKEND_REASON
The reason code from the file control request in the remote system.

BACKEND_RESPONSE
The response code from the file control request in the remote system.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO
YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCRF gate, WRITE function

Write a record in a remote file.

Input Parameters

RECORD_ID
Optional Parameter

The record identifier.

REMOTE_FILE_NAME
Optional Parameter

The file name in the remote system.

REMOTE_SYSTEM
Optional Parameter

The SYSID of the remote system.

WRITE_RECORD
The record to be written in the remote system.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DISASTER_PERCOLATION

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

BACKEND_ACCMETH_RETURN_CODE
The return code from the file access method in the remote system.

BACKEND_LENGTH_ERR_CODE
A value that provides details of a length error that occurred when processing the request in the remote system.

Values for the parameter are:
BUFFER_LEN_NOT_FILE_LEN
BUFFER_LEN_TOO_SMALL
LENGTH_OK
RECORD_LEN_NOT_FILE_LEN
RECORD_LEN_TOO_LARGE

BACKEND_REASON
The reason code from the file control request in the remote system.

BACKEND_RESPONSE
The response code from the file control request in the remote system.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINATE_REQUEST
A binary value that indicates whether a remote file request should be terminated.

Values for the parameter are:
NO

YES

TERMINATE_STRING
A binary value that indicates whether the FRTE string should be terminated.

Values for the parameter are:
NO
YES

FCRL gate, COMMIT_POOLS function

This function catalogs all the shared resources control (SHRCTL) blocks in one operation and is used only during cold start initialization.

Input Parameters

TOKEN
An 8-character token.

Output Parameters

REASON
The following value is returned when RESPONSE is DISASTER:
CATALOG_WRITE_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
DISASTER
INVALID

FCRL gate, SET_POOL function

This function updates the attributes in the VSAM local shared resource (LSR) pool.

Input Parameters

POOL_ID
This binary field specifies the ID of the pool to be updated, in the range 1 - 8.

CATALOG_SHRCTL_BLOCK
This parameter indicates whether to catalog the shared resources control (SHRCTL) block. Values for the parameter are:

YES

NO

MAXIMUM_KEY_LENGTH
Optional Parameter

This binary field specifies the maximum length of the key, in the range 0 - 255.

SHARE_LIMIT
Optional Parameter

This binary field specifies the resource share limit, in the range 1 - 100.

STRING_NUMBER
Optional Parameter

This binary field specifies the number of concurrent requests that can be processed, in the range 0 -255.

BUFFERS_ARRAY
Optional Parameter

A pointer to a SHRCTL block containing buffer counts.

Output Parameters

REASON

The following value is returned when RESPONSE is DISASTER:

```
CATALOG_WRITE_FAILED
```

The following value is returned when RESPONSE is INVALID:

```
INVALID_PARAMETERS
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
DISASTER
INVALID
```

FCRP gate, RESTART_FILE_CONTROL function

This function restarts file control.

Input Parameters

None.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
CATALOG_READ_SHRCTL_FAILED
CATALOG_SHRCTL_NOT_FOUND
DCB_NOT_ON_CATALOG
DSNB_NOT_FOUND
DFP_LEVEL_INVALID
FCBP_RETURNED_DISASTER
FCT_LEVEL_INVALID
FCTE_NOT_FOUND
GATE_NOT_ADDED
IGWARLS_LOAD_FAILED
IGWARLS_NOT_FOUND
INQUIRE_SYSID_FAILED
LISTEN_FAILED
PGDD_FAILED
RCEX_LINK_FAILED
RLS_RESTART_FAILED
SET_GATE_FAILED
TM_ADD_FCT_FAILED
TM_ADD_DSN_FAILED
TM_ADD_DSNA_FAILED
TM_CREATE_FCT_I_FAILED
TM_CREATE_DSN_I_FAILED
TM_CREATE_DSNA_I_FAILED
TM_LOCATE_DSNB_FAILED
XMAT_FAILED
XMXD_FAILED
IGGCSI00_LOAD_FAILED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCRR gate, LOST_LOCKS_RECOVERED function

The LOST_LOCKS_RECOVERED function is called when lost locks recovery for a data set has been completed by all the CICS regions that were sharing it. This function causes the flag in the DSNB, which indicates that the data set is in lost locks state, to be cleared.

Input Parameters

RESOURCE_NAME
The 44-character field containing the name of the data set that has completed lost locks recovery.

Output Parameters

REASON
The following value is returned when RESPONSE is INVALID:
INVALID_FUNCTION

The following value is returned when RESPONSE is EXCEPTION:
SPHERE_UNKNOWN

The following values are returned when RESPONSE is DISASTER:
TM_LOCATE_FAILED
TM_UNLOCK_FAILED
ABEND
DISASTER_PERCOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCRR gate, RESOURCE_AVAILABLE function

This function causes the CICS recovery manager to be notified of the availability of the specified resource.

Input Parameters

RESOURCE_TYPE
The type of resource that has become available. Values for the parameter are:
DSET
CACHE
OTHER

When the **RESOURCE_TYPE** parameter is set to DSET, an RMRE AVAIL call is issued for the specified data set. When the **RESOURCE_TYPE** parameter is set to CACHE, an RMRE AVAIL call is issued for every data set that has outstanding work shunted, because of either a cache failure or a general file backout failure. When the **RESOURCE_TYPE** parameter is set to OTHER, an RMRE AVAIL call is issued for the specified resource.

RESOURCE_NAME
The 44-character field containing the name of the resource that has become available.

RESOURCE_NAME_LENGTH
A halfword binary field containing the length of the resource name.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
INVALID_RESOURCE_TYPE
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DISASTER_PERCOLATION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

FCRR gate, RESTART_RLS function

This function performs a restart of the record-level sharing (RLS) component of file control. The exact processing depends on the type of restart being performed: cold, initial, warm, emergency, or dynamic.

COLD and INITIAL

The RLS control ACB is registered and a cold start of RLS is performed; both processes are initiated through calls to DFHFCCA.

WARM and EMERGENCY

The RLS control ACB is registered and recovery information is inquired upon from SMSVSAM; both processes are initiated through calls to DFHFCCA. If the recovery information indicates that some data sets are in lost locks status, the corresponding DSNBs are marked as being in lost locks state and preparation for lost locks recovery is carried out. Any orphan locks are eliminated.

DYNAMIC

This type of restart occurs when a new instance of the SMSVSAM server becomes available following a previous server failure.

Having waited for file control restart to complete, if it was still in progress, and for any in-progress dynamic RLS restarts to complete, RLS access is drained if necessary, the control ACB is registered, and recovery information is inquired upon from SMSVSAM. All three of these processes are initiated through calls to DFHFCCA.

If the recovery information indicates that some data sets are in lost locks status, the corresponding DSNBs are marked as being lost locks, and preparation for lost locks recovery is carried out. Any orphan locks are eliminated.

The CICS recovery manager is called to recover any shunted units of work that are backout-failed because of the SMSVSAM server failure or a general file backout failure and any units of work that are commit-failed because of the SMSVSAM server failure.

Input Parameters

TYPE_OF_RESTART

Indicates the type of RLS restart being performed. Values for the parameter are:

COLD
WARM
EMERGENCY
DYNAMIC

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION
INVALID_RESTART_TYPE

The following values are returned when RESPONSE is EXCEPTION:

REGISTER_CTL_ACB_FAILED
COLD_START_RLS_FAILED
DRAIN_RLS_FAILED
LOST_LOCKS_INFO_LOST
INQUIRE_RECOVERY_FAILED
LOST_LOCKS_COMPLETE_FAILED
ORPHAN_RELEASE_FAILED

The following values are returned when RESPONSE is DISASTER:

DSSR_FAILED
TM_LOCATE_FAILED
TM_UNLOCK_FAILED
ABEND
DISASTER_PERCOLATION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCSD gate, TERMINATE function

This function closes all the files, either through an immediate or a warm shutdown.

Input Parameters

SHUTDOWN

Specifies the type of shutdown that occurs. Values for the parameter are:

IMMEDIATE
WARM

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

CLOSE_ERROR
RECOVERY_ENTERED
TM_GETNEXT_FCTE_FAILED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCST gate, COLLECT_FILE_STATISTICS function

Returns the statistics for the named file.

Input Parameters

FILE_NAME

The 8–character name of the file.

FC_CONNECT_TOKEN

Optional Parameter

This field is an ETOKEN.

STATISTICS_RECORD

Optional Parameter

Specifies the buffer for the output data.

RESET

Values for the parameter are:

YES
NO

Output Parameters

REASON

The following value is returned when RESPONSE is EXCEPTION:

FILE_NAME_NOT_FOUND

The following values are returned when RESPONSE is INVALID:

INVALID_FC_CONNECT_TOKEN
INVALID_RESET
NO_FILE_NAME
NO_RESET
BAD_BUFF_PTR

BAD_BUFF_LEN
BROWSE_TOKEN_NOT_REQD
POOL_ID_NOT_REQD

The following values are returned when RESPONSE is DISASTER:
TM_LOCATE_FAILED
TM_UNLOCK_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCST gate, COLLECT_POOL_STATISTICS function

Returns statistics for the named local shared resources (LSR) pool.

Input Parameters

POOL_ID
The 8-digit binary ID of the LSR pool.

STATISTICS_RECORD
Optional Parameter

Specifies the buffer for the output data.

RESET
Values for the parameter are:
YES
NO

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
POOL_NOT_BUILT

The following values are returned when RESPONSE is INVALID:
INVALID_POOL_ID
INVALID_RESET
NO_POOL_ID
BAD_BUFF_PTR
BAD_BUFF_LEN
BROWSE_TOKEN_NOT_REQD
FILE_NAME_NOT_REQD

The following value is returned when RESPONSE is DISASTER:
SHRCTL_BLOCK_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID

KERNERROR
PURGED

FCST gate, END_FILE_IN_POOL_BROWSE function

Terminates the browse of files for the named local shared resources (LSR) pool.

Input Parameters

BROWSE_TOKEN

Token returned from the previous browse operation.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

INVALID_BROWSE_TOKEN
NO_BROWSE_TOKEN
CONNECT_TOKEN_NOT_REQD
FILE_NAME_NOT_REQD
STATS_RECORD_NOT_REQD
POOL_ID_NOT_REQD
RESET_NOT_REQD

The following value is returned when RESPONSE is DISASTER:

FREEMAIN_FAILED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCST gate, GET_NEXT_FILE_IN_POOL function

Returns statistics for the next file in the named local shared resources (LSR) pool.

Input Parameters

BROWSE_TOKEN

Token returned from the previous browse operation.

RESET

Values for the parameter are:

YES
NO

STATISTICS_RECORD

Optional Parameter

Specifies the buffer for the output data.

Output Parameters

REASON

The following value is returned when RESPONSE is EXCEPTION:

END_OF_LIST

The following values are returned when RESPONSE is INVALID:

INVALID_BROWSE_TOKEN
NO_BROWSE_TOKEN

NO_RESET
BAD_BUFF_PTR
BAD_BUFF_LEN
CONNECT_TOKEN_NOT_REQD
FILE_NAME_NOT_REQD
POOL_ID_NOT_REQD

The following values are returned when RESPONSE is DISASTER:
TM_GETNEXT_FAILED
TM_UNLOCK_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCST gate, START_FILE_IN_POOL_BROWSE function

Initiates the browse of files for the named local shared resources (LSR) pool.

Input Parameters

POOL_ANY
Optional Parameter

Values for the parameter are:
YES
NO

POOL_ID
The 8-digit binary ID of the LSR pool.

Output Parameters

BROWSE_TOKEN
Token returned that describes the state of the browse operation.

REASON
The following value is returned when RESPONSE is EXCEPTION:
FILE_NAME_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_POOL_ID
INVALID_RESET
NO_POOL_ID
NO_RESET
CONNECT_TOKEN_NOT_REQD
BROWSE_TOKEN_NOT_REQD
FILE_NAME_NOT_REQD
STATS_RECORD_NOT_REQD
RESET_NOT_REQD

The following value is returned when RESPONSE is DISASTER:
GETMAIN_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FCVC gate, INQUIRE_CATALOG function

This function issues a call to IGGCSI00 to obtain catalog information.

Input Parameters

DSNAME
The 44-character name of the data set on which the inquiry is being made.

Output Parameters

EXTENDED
Indicates whether the data set supports extended addressing. Values for the parameter are:
YES
NO

HIGH_XRBA
The highest extended relative byte address (XRBA) used, if the data set supports extended content.

REASON
The following value is returned when RESPONSE is EXCEPTION:
CATALOG_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ICUS gate, CREATE_ICE_USER_EXTENSION function

The CREATE_ICE_USER_EXTENSION function of the ICUS gate is used to create an interval control element (ICE) secure storage extension.

Input Parameters

ICE_TOKEN
Internal control element

FLATTENED_ICUS
Control block for flattened ICUS data.

TERMINAL
States whether the ICE is associated with a terminal.

Values for the parameter are:
NO
YES

USERID
User ID.

SYSTEM_USERID
Region user ID.

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
NOT_AUTHORIZED

The following values are returned when RESPONSE is INVALID:
INVALID_FLATTENED_ICUS
INVALID_FORMAT
INVALID_FUNCTION
INVALID_ICE

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ICUS gate, DELETE_ICE_USER_EXTENSION function

The DELETE_ICE_USER_EXTENSION function of the ICUS gate is used to delete an interval control element (ICE) secure storage extension.

Input Parameters

ICE_TOKEN
Internal control element.

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
NOT_AUTHORIZED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ICUS gate, DELETE_USER_TOKEN function

The DELETE_USER_TOKEN function of the ICUS gate is used to delete the user token from data in a secure storage extension of an interval control element (ICE).

Input Parameters

ICE_TOKEN
Internal control element.

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
NOT_AUTHORIZED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ICUS gate, SET_USER_TOKEN function

The SET_USER_TOKEN function of the ICUS gate is used to reset the user token from data in a secure storage extension of an interval control element (ICE).

Input Parameters

ICE_TOKEN
Internal control element.

Output Parameters

USER_TOKEN
User token to be associated with the secure storage extension.

PRIORITY
Interval control require priority.

REASON
The following value is returned when RESPONSE is EXCEPTION:
NOT_AUTHORIZED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ICXM gate, INQUIRE_FACILITY function

The INQUIRE_FACILITY function of the ICXM gate is used to inquire about the interval control facilities that support facility management calls from the transaction management domain.

Input Parameters

FACILITY_TOKEN
Optional Parameter

The token identifying the transaction that has been trigger-level attached.

Output Parameters

FACILITY_NAME
The four-character name of the transaction that has been trigger-level attached.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LEPT gate, CREATE_LE_ENCLAVE function

The CREATE_LE_ENCLAVE function is used to create a Language Environment enclave.

Input Parameters

RUNOPTS

Optional Parameter

A block that contains run time options for Language Environment's preinitialization services (CEEPIPI).

Output Parameters

REASON

The values for the parameter are:

```
ABEND
CEEPIPI_ERROR
ENQUEUE_ERROR
IPT_ATTACH_ERROR
LOOP
```

CEEPIPI_RESPONSE

The return code from Language Environment's preinitialization services (CEEPIPI).

ENCLAVE_TOKEN

A token that identifies the Language Environment enclave.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LEPT gate, CREATE_PTHREAD function

Create a pthread in a Language Environment enclave.

Input Parameters

ENCLAVE_TOKEN

A token that identifies the Language Environment enclave.

KERNEL_INFORMATION

A vector that is used to pass kernel information to a pthread.

Output Parameters

CEEPIPI_RESPONSE

The return code from Language Environment's preinitialization services (CEEPIPI).

LEPT gate, INVOKE_PTHREAD function

Dispatch a nominated Language Environment function routine under the pthread associated with the current kernel mode.

Input Parameters

ACTIVITY

The intended LE function.

FUNCTION_PARAMETERS

Optional Parameter

Parameters used by the Language Environment function

REMARK

Optional Parameter

A text string that identifies the function.

Output Parameters

REASON

The values for the parameter are:

ABEND
LOOP
NO_PTHREAD
PTHREAD_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FUNCTION_RESPONSE

Optional Parameter

The response from the requested function.

LEPT gate, PTHREAD_REPLY function

The PTHREAD_REPLY function is used to invoke initialization under a pthread.

Output Parameters

REASON

The values for the parameter are:

ABEND
LOOP
NO_PTHREAD
PTHREAD_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LEPT gate, TERMINATE_LE_ENCLAVE function

Terminate a Language Environment enclave.

Input Parameters

ENCLAVE_TOKEN

A token that identifies the Language Environment enclave.

Output Parameters

REASON

The values for the parameter are:

ABEND
CEEPIPI_ERROR
LOOP

CEEPIPI_RESPONSE

The return code from Language Environment's preinitialization services (CEEPIPI).

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LEPT gate, TERMINATE_PTHREAD function

Terminate the current pthread in a Language Environment enclave.

Output Parameters

CEEPIPI_RESPONSE

The return code from Language Environment's preinitialization services (CEEPIPI).

SAIQ gate, INQUIRE_SYSTEM function

The INQUIRE_SYSTEM function of the SAIQ gate is used to inquire upon system data values owned by the application domain.

Input Parameters

GMMTEXT

Optional Parameter

A token identifying the text of the "good-morning" message.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INQ_FAILED
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
LENGTH_ERROR
UNKNOWN_DATA
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CICSREL

Optional Parameter

The CICS release and modification number

CICSSTATUS

Optional Parameter

The initialization or termination status of the CICS system.

Values for the parameter are:

```
ACTIVE
FINALQUIESCE
FIRSTQUIESCE
INITIALIZING
```

CICSSYS

Optional Parameter

A character that indicates the system for which this system was built. Only set by CICS for MVS.

CICSTSLEVEL

Optional Parameter

The level of CICS Transaction server.

COLDSTATUS

Optional Parameter

An indication of whether CICS was started with a COLD or INITIAL start.

Values for the parameter are:

```
COLD
INITIAL
NOTCOLD
```

CWA

Optional Parameter

The address of the common work area.

CWALENGTH

Optional Parameter

The length of the common work area.

DATE

Optional Parameter

The date represented as a packed decimal number integer of the form `0cyyddds` where

- `yy` is the year
- `ddd` is the day
- `c` is the century, where 0 indicates 1900-1999, 1 indicates 2000-2099, and 2 indicates 2100-2199.
- `s` is a positive sign

DTRPRGRM

Optional Parameter

The name of the dynamic routing program.

GMMLENGTH

Optional Parameter

the length of the "good-morning" message text.

GMMTRANID

Optional Parameter

The transaction that generates the "good morning" message.

INITSTATUS

Optional Parameter

The status or phase of initialization.

Values for the parameter are:

```
FIRSTINIT
INITCOMPLETE
SECONDINIT
THIRDINIT
```

JOBNAME

Optional Parameter

The eight-character MVS job name for the local CICS region.

OPREL

Optional Parameter

The release number of the operating system currently running. The value is ten times the formal release number. For example, "21" represents Release 2.1.

OPSYS

Optional Parameter

A one-character identifier indicating the type of operating system currently running. A value of "X" represents MVS.

OSLEVEL

Optional Parameter

The version, release, and modification of z/OS® that is running, each in character form, two bytes each.

PLTPI

Optional Parameter

The two-character suffix of the program list table, which contains a list of programs to be run in the final stages of system initialization.

SDTRAN

Optional Parameter

The shutdown transaction.

SECURITYMGR
Optional Parameter

Indicates whether an external security manager (such as RACF®) is active in the CICS region, or whether no security is being used.

Values for the parameter are:
EXTSECURITY
NOSECURITY

SHUTSTATUS
Optional Parameter

The shutdown status of the local CICS region.

Values for the parameter are:
CANCELLED
CONTROLSHUT
NOTSHUTDOWN
SHUTDOWN

STARTUP
Optional Parameter

The type of startup used for the local CICS region.

Values for the parameter are:
AUTOSTART
COLDSTART
EMERGENCY
STANDBY
WARMSTART

STARTUPDATE
Optional Parameter

A four-character packed-decimal value indicating the date on which the local CICS region was started.

TERMURM
Optional Parameter

The eight-character name of the terminal autoinstall program.

TIMEOFDAY
Optional Parameter

A four-character packed-decimal value indicating the time at which the local CICS region was started (hhmmsstc, where hh=hours, mm=minutes, ss=seconds, c is the sign).

XRFSTATUS
Optional Parameter

Indicates whether the local CICS region is a PRIMARY (active) or TAKEOVER (alternate) XRF CICS region, or has no XRF support.

Values for the parameter are:
NOXRF
PRIMARY
TAKEOVER

SAIQ gate, SET_SYSTEM function

The SET_SYSTEM function of the SAIQ gate is used to set system data values owned by the application domain.

Input Parameters

DTRPRGRM

Optional Parameter

The 8-character name of the program controlling the dynamic routing of transactions.

GMMLENGTH

Optional Parameter

The length of the "good-morning" message text.

GMMTEXT

Optional Parameter

Token identifying the text of the "good-morning" message.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
SET_FAILED
```

The following values are returned when RESPONSE is EXCEPTION:

```
LENGTH_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TDOC gate, CLOSE_ALL_EXTRA_TD_QUEUES function

The CLOSE_ALL_EXTRA_TD_QUEUES function of the TDOC gate closes all extrapartition transient data queues which are currently open in the system. The CLOSE_ALL_EXTRA_TD_QUEUES function is usually invoked as part of a warm shutdown.

Output Parameters

REASON

The values for the parameter are:

```
DCT_ERROR
DIRECTORY_MGR_ERROR
LOGIC_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TDOC gate, CLOSE_TRANSIENT_DATA function

The CLOSE_TRANSIENT_DATA function of the TDOC gate is used to close an extrapartition transient data queue.

Input Parameters

QUEUE

The name of the extrapartition transient data queue to be closed.

TD_QUEUE_TOKEN

Can be specified instead of QUEUE. The token uniquely identifies the extrapartition queue to be closed.

Output Parameters

REASON

The values for the parameter are:

DCT_ERROR
DIRECTORY_MGR_ERROR
LOGIC_ERROR
QUEUE_CLOSED
QUEUE_FULL
QUEUE_INTRA
QUEUE_NOT_CLOSED
QUEUE_NOT_FOUND
QUEUE_OMITTED
QUEUE_REMOTE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TDOC gate, OPEN_TRANSIENT_DATA function

The OPEN_TRANSIENT_DATA function of the TDOC gate is used to open an extrapartition transient data queue.

Input Parameters

QUEUE
The name of the extrapartition transient data queue to be opened.

TD_QUEUE_TOKEN
Can be specified instead of QUEUE. The token uniquely identifies the extrapartition queue to be opened.

BLOCK_LENGTH
Optional Parameter

For blocked data sets, the block length.

BLOCKED
Optional Parameter

The block format of the data set. Indicates if the data set is blocked or unblocked.

Values for the parameter are:
NO
YES

BUFFER_NUMBER
Optional Parameter

The number of data buffers.

CONTROL_CHAR
Optional Parameter

The control characters used in the data set.

Values for the parameter are:
A ASA control characters.
M Machine control characters.

DDNAME
Optional Parameter

The DD name by which the data set is referred to in the startup JCL.

RECORD_FORMAT
Optional Parameter

The record format of the data set.

Values for the parameter are:
F Fixed records
U Unblocked records

V Variable records

RECORD_LENGTH
Optional Parameter

The record length in bytes.

TYPE_FILE
Optional Parameter

The type of data set with which the queue is associated.

Values for the parameter are:
```
INPUT
LEAVE
OUTPUT
RDBACK
REREAD
```

Output Parameters

REASON
The values for the parameter are:
```
DCT_ERROR
DDNAME_NOT_FOUND
DIRECTORY_MGR_ERROR
LOGIC_ERROR
QUEUE_INTRA
QUEUE_NOT_FOUND
QUEUE_NOT_OPENED
QUEUE_OMITTED
QUEUE_OPEN
QUEUE_REMOTE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TDTM gate, ADD_REPLACE_TDQDEF function

Install a transient data queue definition.

Input Parameters

CATALOG_TDQ
Indicates whether to catalog the queue when it is installed.

Values for the parameter are:
```
NO
YES
```

QUEUE_NAME
The name of the queue to be installed.

TD_QUEUE_TOKEN
Can be specified instead of QUEUE. The token uniquely identifies a transient data queue definition that has already been created, but needs to be installed.

BLOCK_LENGTH
Optional Parameter

The block length of an extrapartition queue.

BUFFER_NUMBER
Optional Parameter

The number of buffers to be associated with an extrapartition queue.

DDNAME
Optional Parameter

The DDNAME to be associated with an extrapartition queue.

DISPOSITION

Optional Parameter

The disposition of the data set to be associated with an extrapartition queue.

Values for the parameter are:

```
MOD
OLD
SHR
```

DSNAME

Optional Parameter

The DSNAME of the data set to be associated with an extrapartition queue.

ERROR_OPTION

Optional Parameter

The action to be taken in the event of an I/O error. This input parameter applies to extrapartition queues only.

Values for the parameter are:

```
IGNORE
SKIP
```

FACILITY

Optional Parameter

The facility associated with this intrapartition queue when a trigger transaction is attached.

Values for the parameter are:

```
FILE
SYSTEM
TERMINAL
```

FACILITY_ID

Optional Parameter

Specified together with the FACILITY option, FACILITY_ID identifies the facility that the trigger transaction should be associated with.

INDIRECT_DEST

Optional Parameter

The destination queue if this queue is an indirect queue.

OPEN_TIME

Optional Parameter

Specifies whether this extrapartition queue should be opened as part of installation processing.

Values for the parameter are:

```
DEFERRED
INITIAL
```

RECORD_FORMAT

Optional Parameter

The format of records held in an extrapartition queue.

Values for the parameter are:

```
FIXBLK
FIXBLKA
FIXBLKM
FIXUNB
FIXUNBA
```

FIXUNBM
UNSPECIFIED
VARBLK
VARBLKA
VARBLKM
VARUNB
VARUNBA
VARUNBM

RECORD_LENGTH
Optional Parameter

The record length of an extrapartition queue in bytes.

RECOVERY
Optional Parameter

The recovery type of an intrapartition queue.

Values for the parameter are:
LG
NO
PH

REMOTE_NAME
Optional Parameter

The remote name of the queue if this is a remote queue definition.

REMOTE_SYSTEM
Optional Parameter

The remote system identifier (SYSID) if this is a remote queue definition.

REWIND
Optional Parameter

For extrapartition queues only, where the tape is positioned in relation to the end of the data set.

Values for the parameter are:
LEAVE
REREAD

SYSOUTCLASS
Optional Parameter

The SYSOUT class to be used for the associated output extrapartition queue.

TD_TYPE
Optional Parameter

The queue type.

Values for the parameter are:
EXTRA
INDIRECT
INTRA
REMOTE

TERMINAL_ID
Optional Parameter

The terminal associated with a transaction that is invoked when the trigger level is reached.

TRANSACTION_ID
Optional Parameter

The ATI transaction to be invoked when the trigger level is reached.

TRIGGER_LEVEL
Optional Parameter

The trigger level of the intrapartition queue.

TYPE_FILE
Optional Parameter

indicates whether this queue is:
- an input queue
- an output queue
- to be read backwards.

Values for the parameter are:
```
INPUT
OUTPUT
RDBACK
```

USERID
Optional Parameter

The userid to be associated with a trigger-level attached transaction.

WAIT
Optional Parameter

Specifies whether this logically recoverable intrapartition queue can wait for the resolution of an indoubt failure.

Values for the parameter are:
```
NO
YES
```

WAIT_ACTION
Optional Parameter

The action to be taken if this logically recoverable intrapartition queue suffers an indoubt failure.

Values for the parameter are:
```
QUEUE
REJECT
```

Output Parameters

REASON
The values for the parameter are:
```
CATALOG_WRITE_FAILED
COLD_START_IN_PROGRESS
DDNAME_NOT_FOUND
DFHINTRA_NOT_OPENED
DIRECTORY_MGR_ERROR
DISABLE_PENDING
DUPLICATE
INSUFFICIENT_STORAGE
INVALID_FUNCTION
LOGIC_ERROR
NOT_CLOSED
NOT_DISABLED
NOT_SAME_TYPE
QUEUE_NOT_OPENED
SECURITY_FAILURE
USERID_NOTAUTHED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TDTM gate, COMMIT_TDQDEFS function

Catalog all installed transient data queue definitions as part of cold start processing.

Input Parameters

TOKEN
The catalog to which the queue definitions are to be written.

Output Parameters

REASON
The values for the parameter are:
CATALOG_WRITE_FAILED
DIRECTORY_MGR_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TDTM gate, DISCARD_TDQDEF function

The DISCARD_TDQDEF function of the TDTM gate deletes an installed transient data queue definition and removes it from the catalog. A DELETEQ command is issued as part of the discard process.

Input Parameters

QUEUE_NAME
The queue to be discarded.

TD_QUEUE_TOKEN
Can be specified instead of QUEUE_NAME. TD_QUEUE_TOKEN identifies the queue to be discarded.

Output Parameters

REASON
The values for the parameter are:
CATALOG_DELETE_FAILED
DIRECTORY_MGR_ERROR
DISABLE_PENDING
LOGIC_ERROR
NAME_STARTS_WITH_C
NOT_CLOSED
NOT_DISABLED
QUEUE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TDTM gate, END_BROWSE_TDQDEF function

The END_BROWSE_TDQDEF function of the TDTM gate terminates a browse session.

Input Parameters

BROWSE_TOKEN
Identifies the browse session.

Output Parameters

REASON
The values for the parameter are:
DIRECTORY_MGR_ERROR
LOGIC_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TDTM gate, GET_NEXT_TDQDEF function

The GET_NEXT_TDQDEF function of the TDTM gate returns information about a queue as part of a browse operation.

Input Parameters

BROWSE_TOKEN
Identifies the browse session.

Output Parameters

REASON
The values for the parameter are:
DIRECTORY_MGR_ERROR
LOGIC_ERROR
NO_MORE_DATA_AVAILABLE

QUEUE_NAME
The name of the queue.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ATI_FACILITY
Optional Parameter

The facility associated with this intrapartition queue when a trigger transaction is attached.

Values for the parameter are:
NOTERM
TERM

ATI_TERMID
Optional Parameter

Specified together with the FACILITY option, FACILITY_ID identifies the facility that the trigger transaction should be associated with.

ATI_TRANID
Optional Parameter

The ATI transaction to be invoked when the trigger level is reached.

ATI_USERID
Optional Parameter

The USERID associated with the ATI transaction that is invoked when the trigger level is reached.

BLOCK_LENGTH
Optional Parameter

The block length of an extrapartition queue.

BUFFER_NUMBER
Optional Parameter

The number of buffers to be associated with an extrapartition queue.

DDNAME
Optional Parameter

The DDNAME associated with an extrapartition queue.

DISPOSITION
Optional Parameter

The disposition of the data set associated with an extrapartition queue.

Values for the parameter are:
- MOD
- OLD
- SHR

DSNAME
Optional Parameter

The DSNAME of the data set associated with the extrapartition queue.

EMPTY_STATUS
Optional Parameter

Indicates whether the queue contains any records, and whether the queue is full. This option applies to extrapartition queues only.

Values for the parameter are:
- EMPTY
- FULL
- NOTEMPTY

ENABLE_STATUS
Optional Parameter

The status of the queue.

Values for the parameter are:
- DISABLED
- DISABLING
- ENABLED

ERROR_OPTION
Optional Parameter

The action is to be taken in the event of an I/O error. This option applies to extrapartition queues only.

Values for the parameter are:
- IGNORE
- SKIP

INDIRECT_DEST
Optional Parameter

The destination queue if this queue is an indirect queue.

MEMBER
Optional Parameter

The member name when a PDS member is used for an extrapartition queue.

NUM_ITEMS
Optional Parameter

The number of committed items in the queue.

OPEN_STATUS
Optional Parameter

Indicates whether the queue is open.

Values for the parameter are:

CLOSED
OPEN

RECORD_FORMAT
Optional Parameter

The format of the records held on the extrapartition queue.

Values for the parameter are:
FIXBLK
FIXBLKA
FIXBLKM
FIXUNB
FIXUNBA
FIXUNBM
UNDEFINED
VARBLK
VARBLKA
VARBLKM
VARUNB
VARUNBA
VARUNBM

RECORD_LENGTH
Optional Parameter

The record length of the extrapartition queue.

RECOVERY
Optional Parameter

The recovery type of an intrapartition queue.

Values for the parameter are:
LG
NO
PH

REMOTE_NAME
Optional Parameter

The remote name of the queue if this is a remote queue definition.

REMOTE_SYSTEM
Optional Parameter

The remote system identifier (SYSID) for a remote queue definition.

REWIND
Optional Parameter

Where the tape is positioned in relation to the end of the data set. This parameter applies to extrapartition queues only.

Values for the parameter are:
LEAVE
REREAD

SYSOUTCLASS
Optional Parameter

The SYSOUT class to be used for the associated output extrapartition queue.

TD_TYPE
Optional Parameter

The queue type.

Values for the parameter are:
EXTRA

INDIRECT
INTRA
REMOTE

TERMINAL_ID
Optional Parameter

The terminal associated with a transaction that is invoked when the trigger level is reached.

TRANSACTION_ID
Optional Parameter

The ATI transaction to be invoked when the trigger level is reached.

TRIGGER_LEVEL
Optional Parameter

The trigger level of the intrapartition queue.

TYPE_FILE
Optional Parameter

specifies whether this queue is:
- an input queue
- an output queue
- a queue that is to be read backwards.

Values for the parameter are:
INPUT
OUTPUT
RDBACK

USERID_TOKEN
Optional Parameter

A token for the USERID that was specified for this intrapartition queue.

WAIT
Optional Parameter

Specifies whether this logically recoverable intrapartition queue can wait for the resolution of an indoubt failure.

Values for the parameter are:
NO
YES

WAIT_ACTION
Optional Parameter

The action to be taken if this logically recoverable intrapartition queue suffers an indoubt failure.

Values for the parameter are:
QUEUE
REJECT

TDTM gate, INQUIRE_TDQDEF function

The INQUIRE_TDQUEUE function of the TDTM gate is used to inquire on a specified queue.

Input Parameters

QUEUE_NAME
The name of the queue.

Output Parameters

REASON

The values for the parameter are:

DIRECTORY_MGR_ERROR
LOGIC_ERROR
QUEUE_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ATI_FACILITY

Optional Parameter

The facility associated with this intrapartition queue when a trigger transaction is attached.

Values for the parameter are:

NOTERM
TERM

ATI_TERMID

Optional Parameter

Specified together with the FACILITY option, FACILITY_ID identifies the facility that the trigger transaction should be associated with.

ATI_TRANID

Optional Parameter

The ATI transaction to be invoked when the trigger level is reached.

BLOCK_LENGTH

Optional Parameter

The block length of an extrapartition queue.

BUFFER_NUMBER

Optional Parameter

The number of buffers to be associated with an extrapartition queue.

DDNAME

Optional Parameter

The DDNAME associated with an extrapartition queue.

DISPOSITION

Optional Parameter

The disposition of the data set associated with an extrapartition queue.

Values for the parameter are:

MOD
OLD
SHR

DSNAME

Optional Parameter

The DSNAME of the data set associated with the extrapartition queue.

EMPTY_STATUS

Optional Parameter

Indicates whether the queue contains any records, and whether the queue is full. This option applies to extrapartition queues only.

Values for the parameter are:

EMPTY
FULL
NOTEMPTY

ENABLE_STATUS
Optional Parameter

The status of the queue.

Values for the parameter are:
DISABLED
DISABLING
ENABLED

ERROR_OPTION
Optional Parameter

The action is to be taken in the event of an I/O error. This option applies to extrapartition queues only.

Values for the parameter are:
IGNORE
SKIP

INDIRECT_DEST
Optional Parameter

The destination queue if this queue is an indirect queue.

MEMBER
Optional Parameter

The member name when a PDS member is used for an extrapartition queue.

NUM_ITEMS
Optional Parameter

The number of committed items in the queue.

OPEN_STATUS
Optional Parameter

Indicates whether the queue is open.

Values for the parameter are:
CLOSED
OPEN

RECORD_FORMAT
Optional Parameter

The format of the records held on the extrapartition queue.

Values for the parameter are:
FIXBLK
FIXBLKA
FIXBLKM
FIXUNB
FIXUNBA
FIXUNBM
UNDEFINED
VARBLK
VARBLKA
VARBLKM
VARUNB
VARUNBA
VARUNBM

RECORD_LENGTH
Optional Parameter

The record length of the extrapartition queue.

RECOVERY
Optional Parameter

The recovery type of an intrapartition queue.

Values for the parameter are:
LG
NO
PH

REMOTE_NAME
Optional Parameter

The remote name of the queue if this is a remote queue definition.

REMOTE_SYSTEM
Optional Parameter

The remote system identifier (SYSID) for a remote queue definition.

REWIND
Optional Parameter

Where the tape is positioned in relation to the end of the data set. This parameter applies to extrapartition queues only.

Values for the parameter are:
LEAVE
REREAD

SYSOUTCLASS
Optional Parameter

The SYSOUT class to be used for the associated output extrapartition queue.

TD_QUEUE_TOKEN
Optional Parameter

The token associated with the queue.

TD_TYPE
Optional Parameter

The queue type.

Values for the parameter are:
EXTRA
INDIRECT
INTRA
REMOTE

TERMINAL_ID
Optional Parameter

The terminal associated with a transaction that is invoked when the trigger level is reached.

TRANSACTION_ID
Optional Parameter

The ATI transaction to be invoked when the trigger level is reached.

TRIGGER_LEVEL
Optional Parameter

The trigger level of the intrapartition queue.

TYPE_FILE
Optional Parameter

specifies whether this queue is:
- an input queue
- an output queue

- a queue that is to be read backwards.

Values for the parameter are:
INPUT
OUTPUT
RDBACK

USERID_TOKEN
Optional Parameter

A token for the USERID that was specified for this intrapartition queue.

WAIT
Optional Parameter

Specifies whether this logically recoverable intrapartition queue can wait for the resolution of an indoubt failure.

Values for the parameter are:
NO
YES

WAIT_ACTION
Optional Parameter

The action to be taken if this logically recoverable intrapartition queue suffers an indoubt failure.

Values for the parameter are:
QUEUE
REJECT

TDTM gate, SET_TDQDEF function

The SET_TDQUEUE function of the TDTM gate updates attributes of an installed transient data queue.

Input Parameters

QUEUE_NAME
The name of the queue.

ATI_FACILITY
Optional Parameter

The facility associated with this intrapartition queue when a trigger transaction is attached.

Values for the parameter are:
NOTERM
TERM

ATI_TERMID
Optional Parameter

Specified together with the FACILITY option, FACILITY_ID identifies the facility that the trigger transaction should be associated with.

ATI_TRANID
Optional Parameter

The ATI transaction to be invoked when the trigger level is reached.

ATI_USERID
Optional Parameter

The USERID associated with the ATI transaction that is invoked when the trigger level is reached.

ENABLE_STATUS
Optional Parameter

The status of the queue.

Values for the parameter are:

DISABLED
DISABLING
ENABLED

TRIGGER_LEVEL

Optional Parameter

The trigger level of the intrapartition queue.

USERID_TOKEN

Optional Parameter

A token for the USERID that was specified for this intrapartition queue.

Output Parameters

REASON

The values for the parameter are:

CATALOG_WRITE_FAILED
DIRECTORY_MGR_ERROR
DISABLE_PENDING
IS_CXRF
LOGIC_ERROR
NOT_CLOSED
NOT_DISABLED
QUEUE_IS_INDOUBT
QUEUE_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OLD_USER_TOKEN

Optional Parameter

The token associated with a previous USERID.

TDTM gate, START_BROWSE_TDQDEF function

The START_BROWSE_TDQDEF function of the TDTM gate initiates a browse of transient data queues. You can optionally specify the queue where the browse starts.

Input Parameters

START_AT

Optional Parameter

The queue from which the browse should start.

Output Parameters

REASON

The values for the parameter are:

DIRECTORY_MGR_ERROR
LOGIC_ERROR

BROWSE_TOKEN

A token that uniquely identifies the browse session.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TDXM gate, BIND_SECONDARY_FACILITY function

The BIND_FACILITY function of the TDXM gate is used to associate a transaction with the definition for the transient data queue that caused the transaction to be trigger-level attached, where the principal facility is the queue itself (that is there is no terminal associated with the queue).

Output Parameters

REASON

The values for the parameter are:

ABEND

FACILITY_NAME

The name of the transient data queue that is associated with the transaction as its principal facility.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TDXM gate, INQUIRE_TRAN_DATA_FACILITY function

Return attributes of a transient data queue.

Input Parameters

TRANSIENT_DATA_TOKEN

Optional Parameter

A token that represents the transient data queue.

Output Parameters

FACILITY_NAME

The name of the transient data queue that is associated with the transaction as its principal facility.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, ALLOCATE function

The ALLOCATE function of the TFAL gate is used to allocate a terminal for a transaction.

Input Parameters

REQUEST_ID

The four-character transaction identifier initiating the attach.

SYSTEM_TOKEN

The token identifying the CICS region to which the terminal is to be attached.

MODE_NAME

Optional Parameter

The eight-character mode-name of the terminal to be attached.

NON_PURGEABLE

Optional Parameter

Indicates whether or not the terminal is to be purgeable.

Values for the parameter are:

NO

YES

PRIVILEGED

Optional Parameter

Indicates whether or not the terminal is to be attached as a privileged terminal.

Values for the parameter are:
NO
YES

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMINAL_TOKEN
A token identifying the terminal that has been attached.

TFAL gate, CANCEL_AID function

The CANCEL_AID function of the TFAL gate is used to cancel a terminal-transaction AID.

Input Parameters

TERM_OWNER_NETNAME
The APPLID of the CICS region that owns the terminal.

TERMID
The four-character terminal identifier.

TRANID
The four-character transaction identifier.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, CANCEL_AIDS_FOR_CONNECTION function

The CANCEL_AIDS_FOR_CONNECTION function of the TFAL gate is used to cancel AIDs for the given CICS region.

Input Parameters

CALLER
The method used to call this function.

Values for the parameter are:
API
BUILDER

FACILITY
The facility type associated with the AIDs.

Values for the parameter are:
CONNECTION
TERMINAL

FORCE
Indicates whether or not system AIDs are to be canceled.

Values for the parameter are:
NO
YES

SYSTEM_TOKEN
The token identifying the CICS region.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

AIDS_CANCELLED

Optional Parameter

Indicates whether or not AIDs were canceled as a result of this request.

Values for the parameter are:

NO
YES

TFAL gate, CANCEL_AIDS_FOR_TERMINAL function

The CANCEL_AIDS_FOR_TERMINAL function of the TFAL gate is used to cancel all AIDs for the given terminal.

Input Parameters

CALLER

The method used to call this function.

Values for the parameter are:

API
BUILDER
BUILDER_REMDEL

FACILITY

The facility type associated with the AIDs.

Values for the parameter are:

CONNECTION
TERMINAL

TERMID

The four-character terminal identifier.

TERMINAL_TOKEN

The token identifying the terminal.

BMSONLY

Optional Parameter

Indicates whether to cancel BMS AIDs only.

Values for the parameter are:

NO
YES

FORCE

Optional Parameter

Indicates whether or not system AIDs are to be canceled.

Values for the parameter are:

NO
YES

TERM_OWNER_NETNAME

Optional Parameter

The netname of the terminal owner.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

AIDS_CANCELLED
Optional Parameter

Indicates whether or not AIDs were canceled as a result of this request.

Values for the parameter are:
NO
YES

TFAL gate, CANCEL_SPECIFIC_AID function

Cancel a single, specified AID.

Input Parameters

AID_TOKEN
A token for the AID that is to be canceled.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, CHECK_TRANID_IN_USE function

The CHECK_TRANID_IN_USE function of the TFAL gate is used to check whether any of the AID chains contain references to the given TRANID.

Input Parameters

TRANID
The four-character transaction identifier.

Output Parameters

IN_USE
Indicates whether or not the transaction identifier specified by the TRANID parameter is in use.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, DISCARD_AIDS function

The DISCARD_AIDS function of the TFAL gate is used to attach a task which will release start data and free the AIDs in the chain addressed by the AID_TOKEN.

Input Parameters

AID_TOKEN
The token identifying the chain of AIDs.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, FIND_TRANSACTION_OWNER function

The FIND_TRANSACTION_OWNER function of the TFAL gate is used to determine the CICS region that owns the given transaction (that is, at which the transaction instance originated).

Input Parameters

TERMINAL_TOKEN
The token identifying the terminal.

TRANID
The four-character transaction identifier.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRAN_OWNER_SYSID
The four-character system identifier for the CICS region that owns the transaction instance.

TFAL gate, GET_MESSAGE function

The GET_MESSAGE function of the TFAL gate is used to get a message from a terminal.

Input Parameters

PREVIOUS_AID_TOKEN
The AID token identifying the previous transaction that ran at this terminal.

TERMINAL_TOKEN
The token identifying the terminal.

Output Parameters

AID_TOKEN
The AID token identifying the current transaction for which the message was got.

BMS_TITLE_PRESENT
Indicates whether or not a BMS title is present on the terminal.

Values for the parameter are:

```
NO
YES
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSQUEUE_NAME
the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

TFAL gate, INITIALIZE_AID_POINTERS function

The INITIALIZE_AID_POINTERS function of the TFAL gate is used to initialize the AID pointers for the given CICS region.

Input Parameters

SYSTEM_TOKEN
The token identifying the CICS region.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, INQUIRE_ALLOCATE_AID function

The INQUIRE_ALLOCATE_AID function of the TFAL gate is used to inquire about the AIDs allocated for the given CICS region.

Input Parameters

SYSTEM_TOKEN
The token identifying the CICS region.

PRIVILEGED
Optional Parameter

indicates whether or not to inquire only about privileged ISC type AIDs.

Values for the parameter are:
NO
YES

Output Parameters

EXISTS
Indicates whether or not the AID exists.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, LOCATE_AID function

The LOCATE_AID function of the TFAL gate is used for automatic transaction initiation to determine the AID for the specified terminal, and if found, to use the transaction identifier from the AID to attach the task.

Input Parameters

TERMID
The four-character terminal-identifier.

TYPE
Optional Parameter

The type of AID to be located.

Values for the parameter are:
BMS
INT
ISC
PUT
REMDEL
TDP

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANID
Optional Parameter

Te four-character transaction identifier associated with the specified terminal.

TFAL gate, LOCATE_REMDEL_AID function

The LOCATE_REMDEL_AID function of the TFAL gate is used to determine the AID (for a delete remote TERMINAL definition request) for the specified system (SYSTEM_TOKEN specified) or after the given transaction (PREVIOUS_AID_TOKEN specified).

Input Parameters

PREVIOUS_AID_TOKEN
The AID token identifying the previous transaction that ran at this terminal.

SYSTEM_TOKEN
The token identifying the CICS region.

Output Parameters

AID_TOKEN
The AID token identifying the transaction to be deleted.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TARGET_SYSID
The four-character system identifier for the target CICS system.

TERM_OWNER_NETNAME
The eight-character netname from the REMDEL AID.

TERMID
The four-character terminal identifier from the REMDEL AID.

TFAL gate, LOCATE_SHIPPABLE_AID function

The LOCATE_SHIPPABLE_AID function of the TFAL gate is used to determine an AID (for a delete remote TERMINAL definition request or for a remote terminal request) to be shipped to the specified system.

Input Parameters

SYSTEM_TOKEN
The token identifying the CICS region.

Output Parameters

AID_TOKEN
the AID token identifying the transaction to be deleted.

LAST
Indicates that:

- there is a single qualifying AID or all qualifying AIDs have the same AIDTRMID (YES)
- *or* in addition to the AID returned there are other qualifying AIDs (NO).

Values for the parameter are:

```
NO
YES
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, MATCH_TASK_TO_AID function

The MATCH_TASK_TO_AID function of the TFAL gate is used to inquire about AIDs for the given terminal and transaction.

Input Parameters

TERMINAL_TOKEN
The token identifying the terminal

TRANID
The four-character transaction identifier.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, PURGE_ALLOCATE_AIDS function

The PURGE_ALLOCATE_AIDS function of the TFAL gate is used to delete purgeable allocate AIDs for a given connection after user exit XZIQUE in DFHZISP has issued return code 8 (delete all) or return code 12 (delete all for given modegroup).

Input Parameters

SYSTEM_TOKEN
The token identifying the CICS region.

MODE_NAME
Optional Parameter

The name of the modegroup. If this parameter is omitted, the default is all modegroups.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ALLOCATES_PURGED
Optional Parameter

The number of ALLOCATE AIDs purged.

TFAL gate, RECOVER_START_DATA function

The RECOVER_START_DATA function of the TFAL gate is used to retrieve a PUT-type AID stored in a DWE and rechain it onto the TCTSE in front of the first AID for the terminal.

Input Parameters

AID_TOKEN
The AID token identifying the transaction to be recovered.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, REMOTE_DELETE function

The REMOTE_DELETE function of the TFAL gate is used to chain a REMOTE DELETE (REMDEL) AID onto the system entry of the specified target CICS region. The REMDEL AID tells the target region to delete its shipped definition of the specified terminal.

Input Parameters

TARGET_SYSID
the four-character system identifier for the target CICS region.

TERM_OWNER_NETNAME
Is the z/OS Communications Server APPLID of the CICS region that "owns" the terminal.

Note: The terminal identifier can either be specified as TERMID and TERM_OWNER_NETNAME (where TERMID is the name known in the terminal owning system), or it can be specified by TERMINAL_TOKEN if the TCTTE address is known.

TERMID
The four-character terminal identifier for the terminal associated with the transaction.

TERMINAL_TOKEN
The token identifying the terminal.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, REMOVE_EXPIRED_AID function

The REMOVE_EXPIRED_AID function of the TFAL gate is used to search all AID chains for a BMS AID that has yet to be initiated and which matches the eligibility parameters. Unchain the first such AID found, copy details from the AID into the caller's parameter list, and freemain the AID.

Input Parameters

ADJUSTED_EXPIRY_TIME
Optional Parameter

The adjusted threshold time

LDC
Optional Parameter

The logical device code.

Note: If MSGID and LDC are specified, the expiry time is not checked.

MSGID
Optional Parameter

The BMS message identifier

NORMAL_EXPIRY_TIME
Optional Parameter

The normal threshold time

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TERMID
The four-character terminal identifier for the terminal associated with the transaction.

TRANID
The four-character transaction identifier associated with the specified terminal.

TSQUEUE_NAME
The eight-character name of the temporary storage queue name of the message whose BMS AID was found.

TFAL gate, REMOVE_EXPIRED_REMOTE_AID function

Search for an uninitiated remote AID which is older than the expiry time specified by the caller. Unchain the AID and clean up any associated resources.

Input Parameters

ADJUSTED_EXPIRY_TIME
The adjusted threshold time

NORMAL_EXPIRY_TIME
The normal threshold time

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHIPPED
Identifies whether the AID has been shipped.

TERM_OWNER_SYSID
The system identifier of the CICS region that "owns" the terminal.

TERMID
The four-character terminal identifier for the terminal associated with the transaction.

TRANID
The four-character transaction identifier associated with the terminal.

TFAL gate, REMOVE_MESSAGE function

The REMOVE_MESSAGE function of the TFAL gate is used to find an uninitiated BMS AID for the specified terminal; unchain and freemain the AID, provided that the AID security fields match those of the currently signed-on operator; and return the TS queue name from the AID.

Input Parameters

TERMINAL_TOKEN
The token identifying the terminal.

MSGID
Optional Parameter

The BMS message identifier

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSQUEUE_NAME
The eight-character name of the temporary storage queue name for the message whose BMS AID was found.

TFAL gate, REMOVE_REMOTE_DELETES function

The REMOVE_REMOTE_DELETES function of the TFAL gate is used to unchain and freemain all REMDEL AIDs from the AID chain of the specified system entry. Optional parameters TERMID and TERM_OWNER_NETNAME may be specified; in which case only those REMDEL AIDs which match the specified values are removed.

Input Parameters

SYSTEM_TOKEN
is the token identifying the CICS region.

Note: Specify either the TARGET_SYSID parameter or the SYSTEM_TOKEN parameter, not both.

TARGET_SYSID
The four-character system identifier for the target CICS region.

TERM_OWNER_NETNAME
Optional Parameter

The netname of the region that "owns" the terminal.

TERMID
Optional Parameter

The four-character terminal identifier for the terminal associated with the transaction.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, REROUTE_SHIPPABLE_AIDS function

The REROUTE_SHIPPABLE_AIDS function of the TFAL gate is used to redirect AIDs for remote terminals from one remote system to another.

Input Parameters

ORIGINAL_SYSTEM_TOKEN
The token identifying the remote system which was the AIDs' original target.

PREV_TERM_OWNER_NETNAME
The APPLID of the CICS region that previously owned the terminal.

TARGET_SYSTEM_TOKEN
The token identifying the remote system which is the AIDs' new target.

TERM_OWNER_NETNAME
The APPLID of the CICS region that owns the terminal.

TERMINAL_NETNAME
The eight-character NETNAME which identifies the terminal whose AIDs are to be rerouted.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, RESCHEDULE_BMS function

The RESCHEDULE_BMS function of the TFAL gate is used to build a BMS AID and chain it to the front of the AID queue.

Input Parameters

BMS_TIMESTAMP
The time stamp for a BMS AID that is used to test if AID is older than specified EXPIRY_TIME.

TERMINAL_TOKEN
The token identifying the terminal.

TRANID
The four-character transaction identifier associated with the specified terminal.

TSQUEUE_NAME
The eight-character name of the temporary storage queue name of the message whose BMS AID was found.

BMS_TITLE_PRESENT
Optional Parameter

Indicates if there is a title in the message control record

Values for the parameter are:

NO
YES

OPCLASS
Optional Parameter

Identifies the operator class.

Note: You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

OPIDENT
Optional Parameter

Identifies the operator.

Note: You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, RESET_AID_QUEUE function

The RESET_AID_QUEUE function of the TFAL gate is used to give DFHALP an opportunity to reset the AID queue when a transaction ends, and to bid for the use of the terminal if ATI tasks are waiting.

Input Parameters

TERMINAL_TOKEN
The token identifying the terminal.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, RESTORE_FROM_KEYPOINT function

The RESTORE_FROM_KEYPOINT function of the TFAL gate is used to reschedule a chain of AIDs that we restored from the catalog during CICS system initialization.

Input Parameters

AID_TOKEN
A token denoting the chain of AIDs which are to be rescheduled.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, RETRIEVE_START_DATA function

The RETRIEVE_START_DATA function of the TFAL gate is used to return the AID address and temporary storage queue name associated with the start data for the specified transaction and terminal.

Input Parameters

TERMINAL_TOKEN
The token identifying the terminal.

TRANID
The four-character transaction identifier associated with the specified terminal.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSQUEUE_NAME
The eight-character name of the temporary storage queue name of the message whose BMS AID was found.

TFAL gate, SCHEDULE_BMS function

The SCHEDULE_BMS function of the TFAL gate is used to schedule a BMS AID.

Input Parameters

BMS_TIMESTAMP
The timestamp for the BMS AID. This is used to test if the AID is older than its EXPIRY_TIME.

TERMID
The four-character terminal identifier for the terminal associated with the transaction.

TRANID
The four-character transaction identifier associated with the specified terminal.

TSQUEUE_NAME
The eight-character name of the temporary storage queue name of the message whose BMS AID was found.

BMS_TITLE_PRESENT
Optional Parameter

Indicates if the title is in the message control record.

Values for the parameter are:

```
NO
YES
```

OPCLASS
Optional Parameter

Identifies the operator class.

Note: You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

OPIDENT
Optional Parameter

Identifies the operator.

Note: You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

TERMINAL_NETNAME
Optional Parameter

The eight-character NETNAME which identifies the terminal.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, SCHEDULE_START function

The SCHEDULE_START function of the TFAL gate is used to schedule a PUT or INT type AID

Input Parameters

TERMID
The four-character terminal identifier for the terminal associated with the transaction.

TRANID
the four-character transaction identifier associated with the specified terminal.

CHANNEL_TOKEN
Optional Parameter

A token for the channel associated with the START request.

DYNAMIC_TRAN
Optional Parameter

Indicates if the transaction is dynamically routed.

Values for the parameter are:
NO
YES

FEPI
Optional Parameter

Indicates whether this is a FEPI START request.

Values for the parameter are:
NO
YES

IN_DOUBT
Optional Parameter

Indicates whether the Unit of Work making the request is in doubt, and, if so, that the request should not be scheduled until the Unit of Work is committed.

Values for the parameter are:
NO
YES

MODE_NAME
Optional Parameter

The mode name to be used.

RECOVERABLE_DATA
Optional Parameter

Indicates whether the request is associated with recoverable data.

Values for the parameter are:
NO
YES

ROUTABLE_START
Optional Parameter

Indicates if the START request can be routed.

Values for the parameter are:

NO
YES

ROUTED_FROM_TERMID
Optional Parameter

The four-character terminal identifier for the terminal from which a task was transaction-routed to issue this START request.

SHIPPED_VIA_SESSID
Optional Parameter

The identifier of the session via which this START request was function shipped.

SHIPPED_VIA_SYSID
Optional Parameter

Identifies the connection via which this request was function shipped or transaction routed.

START_DATA_LEN
Optional Parameter

The length of the data associated with the START request.

TERM_OWNER_NETNAME
Optional Parameter

The system identifier of the CICS region to which the request should be shipped.

Note: You can specify either the TERM_OWNER_SYSID parameter or TERM_OWNER_NETNAME parameter, not both.

TERM_OWNER_SYSID
Optional Parameter

The system identifier of the CICS region to which the request should be shipped.

Note: You can specify either the TERM_OWNER_SYSID parameter or TERM_OWNER_NETNAME parameter, not both.

TERMINAL_NETNAME
Optional Parameter

The eight-character NETNAME of the terminal associated with the transaction.

TERMINAL_TOKEN
Optional Parameter

The token identifying the terminal.

TOR_NETNAME
Optional Parameter

The netname of the CICS region that owns the terminal.

TRAN_OWNER_SYSID
Optional Parameter

The system identifier of the CICS region that “owns” the transaction.

TSQUEUE_NAME
Optional Parameter

The name of the temporary storage queue which contains the data associated with the START request.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, SCHEDULE_TDP function

The SCHEDULE_TDP function of the TFAL gate is used to schedule a TDP type AID.

Input Parameters

TDQUEUE_NAME
The destination identifier for the TD queue.

TERMID
The four-character terminal identifier for the terminal associated with the transaction.

TRANID
The four-character transaction identifier associated with the specified terminal.

TERMINAL_NETNAME
Optional Parameter

The eight-character NETNAME of the terminal associated with the transaction.

Output Parameters

AID_TOKEN
The AID token identifying the transaction to be scheduled.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, SLOWDOWN_PURGE function

The SLOWDOWN_PURGE function of the TFAL gate is used to search the specified system entry's AID chain for the first allocate-type AID associated with a stall-purgeable task, and cancel the identified transaction.

Input Parameters

SYSTEM_TOKEN
The four-character terminal identifier for the terminal associated with the transaction.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, TAKE_KEYPOINT function

The TAKE_KEYPOINT function of the TFAL gate is used to return a chain of AIDs which are to be written to the global catalog.

Output Parameters

AID_TOKEN
The token identifying the chain of AIDs.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, TERM_AVAILABLE_FOR_QUEUE function

The TERM_AVAILABLE_FOR_QUEUE function of the TFAL gate is used, when a terminal becomes available for allocation, to give DFHALP the chance to attach or resume a task which requires this terminal.

Input Parameters

TERMINAL_TOKEN
The token identifying the terminal.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, TERMINAL_NOW_UNAVAILABLE function

The TERMINAL_NOW_UNAVAILABLE function of the TFAL gate is used to perform required actions when a terminal or connection becomes unavailable.

Input Parameters

TERMINAL_TOKEN
The token identifying the terminal.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, UNCHAIN_AID function

The UNCHAIN_AID function of the TFAL gate is used to unchain and optionally freemain the specified AID.

Input Parameters

AID_TOKEN
The AID token identifying the transaction to be deleted.

FREEMAIN
Indicates whether freemain is wanted.

Values for the parameter are:
NO
YES

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFAL gate, UPDATE_TRANNUM_FOR_RESTART function

The UPDATE_TRANNUM_FOR_RESTART function of the TFAL gate is used to update the AID's TRANNUM to that of the restarted task.

Input Parameters

NEW_TRANNUM
The new TRANNUM to be set in the AID.

ORIGINAL_TRANNUM
The TRANNUM set in the AID when original task was attached.

TERMINAL_TOKEN
The token identifying the terminal.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFBF gate, BIND_FACILITY function

The BIND_FACILITY function of the TFBF gate is used to associate a transaction with the terminal.

Input Parameters

PARTITIONSET

Indicates if a partition set is to be used for the terminal facility.

The values for the parameter are:

NONE
NAME
OWN
KEEP

PARTITIONSET_NAME

Optional Parameter

The eight-character name of a partition set. This parameter is used only if the value of PARTITIONSET is NAME.

Output Parameters

REASON

The values for the parameter are:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
NO_TERMINAL
REMOTE_SCHEDULE_FAILURE
SECURITY_FAILURE
TABLE_MANAGER_FAILURE
TRANSACTION_ABEND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFIQ gate, INQUIRE_MONITOR_DATA function

Return monitoring data for a terminal facility.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

NO_TERMINAL

The following values are returned when RESPONSE is INVALID:

INVALID_TERMINAL_TYPE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESS_METHOD

Optional Parameter

A value that indicates the access method for the terminal.

Values for the parameter are:

ACC_NOTAPPLIC

BGAM
BSAM
BTAM
CONSOLE
TCAM
TCAMSNA
VTAM

Note: VTAM® is the z/OS Communications Server.

CONNECTION_NAME
Optional Parameter

The name of the connection that is associated with the terminal facility. If the facility is a surrogate, the value of the parameter is the name of the connection associated with the relay session entry. If the facility is a session, the value of the parameter is the name of the connection associated with the session.

DEVICE
Optional Parameter

The type of device represented by the terminal facility.

FACILITY_NAME
Optional Parameter

The four-character name of the terminal facility.

FACILITY_TYPE
Optional Parameter

The terminal facility type.

Values for the parameter are:
IRC
IRC_XCF
IRC_XM
LU61
LU62
OTHER

INPUT_MESSAGE_LENGTH
Optional Parameter

The length of the current input message for the terminal facility.

NATURE
Optional Parameter

Indicates the nature of the terminal facility.

Values for the parameter are:
MODEL
SESSION
SURROGATE
TERMINAL

NETID
Optional Parameter

The network identifier of the terminal facility.

NETNAME
Optional Parameter

The network name of the terminal facility.

REAL_NETNAME
Optional Parameter

The real network name if a network qualified name has been received from z/OS Communications Server.

SESSION_TYPE
Optional Parameter

The type of session represented by the terminal facility.

Values for the parameter are:
```
APPCPARALLEL
APPCSINGLE
LU61
TYPE_NOTAPPLIC
```

TNADDR_PORT
Optional Parameter

The port number for a Telnet resource

TNADDR_TPADDR
Optional Parameter

The IP address for a Telnet resource.

TFIQ gate, INQUIRE_TERMINAL_FACILITY function

The INQUIRE_TERMINAL_FACILITY function of the TFIQ gate is used to inquire about attributes of a named terminal facility.

Input Parameters

TERMINAL_TOKEN
Optional Parameter

A token identifying a terminal.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
NO_TERMINAL
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_TERMINAL_TYPE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CHANNEL_TOKEN
Optional Parameter

A token that identifies a channel that is to be associated with the terminal.

DEVICE
Optional Parameter

The type of device represented by the terminal facility.

FACILITY_NAME
Optional Parameter

The four-character name of the terminal facility.

FREE_REQUIRED
Optional Parameter

A binary value that indicates that the terminal facility is ready to be freed.

Values for the parameter are:
```
NO
YES
```

INSPECT_DATA
Optional Parameter

A token indicating the Language Environment runtime options for the terminal facility.

NATIONAL_LANGUAGE_IN_USE
Optional Parameter

The three-character code indicating the national language in use for the terminal facility.

NATURE
Optional Parameter

Indicates the nature of the terminal facility.

Values for the parameter are:
```
MODEL
SESSION
SURROGATE
TERMINAL
```

NETNAME
Optional Parameter

The eight-character netname of the terminal facility.

OPERATOR_ID
Optional Parameter

The operator identifier associated with the terminal facility.

PSEUDO_CONV_COMMAREA
Optional Parameter

A block into which the communications area for a pseudo-conversational transaction is copied.

STORAGE_FREEZE
Optional Parameter

Indicates whether or not storage normally freed during the processing of a transaction for the terminal facility is to be frozen. The frozen storage is not freed until the end of the transaction.

Values for the parameter are:
```
NO
YES
```

TERMINAL_TRAFFIC_READ
Optional Parameter

Indicates whether or not reading is supported.

Values for the parameter are:
```
NO
YES
```

TERMINAL_TRAFFIC_WRITE
Optional Parameter

Indicates whether or not writing is supported.

Values for the parameter are:
```
NO
YES
```

TERMINAL_USER_AREA
Optional Parameter

A block into which the terminal user area is copied.

TFIQ gate, SET_TERMINAL_FACILITY function

The SET_TERMINAL_FACILITY function of the TFIQ gate is used to set attributes of a named terminal facility.

Input Parameters

CHANNEL_TOKEN
Optional Parameter

A token that identifies a channel that is to be associated with the terminal.

COUNT_STORAGE_VIOLATION
Optional Parameter

Indicates whether or not storage violations are to be counted for this terminal facility.

Values for the parameter are:
NO
YES

INPUTMSG
Optional Parameter

A block into which the input message for a pseudo-conversational transaction is copied.

INSPECT_DATA
Optional Parameter

Data used by the Inspect tool.

NATIONAL_LANGUAGE_IN_USE
Optional Parameter

The three-character code indicating the national language in use for the terminal facility.

PSEUDO_CONV_COMMAREA
Optional Parameter

A block into which the communications area for a pseudo-conversational transaction is copied.

PSEUDO_CONV_IMMEDIATE
Optional Parameter

A binary value that indicates whether the terminal is to be set into a pseudo conversation.

Values for the parameter are:
NO
YES

PSEUDO_CONV_NEXT_TRANSID
Optional Parameter

The four-character identifier of the transaction to which control is passed on a normal return from a pseudo-conversational transaction (to which the pseudo_conversational data is passed).

STORAGE_FREEZE
Optional Parameter

Indicates whether or not storage normally freed during the processing of a transaction for the terminal facility is to be frozen. (The frozen storage is not freed until the end of the transaction.)

Values for the parameter are:
NO
YES

TERMINAL_TOKEN
Optional Parameter

A token identifying a terminal.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NO_TERMINAL
PERMANENT_TRANSID
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_TERMINAL_TYPE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFRF gate, RELEASE_FACILITY function

Release a transaction's principal facility.

Input Parameters

RESTART
Specifies whether to restart the transaction.

Values for the parameter are:

```
NO
YES
```

TERMINATION_TYPE
Specifies whether transaction termination is normal or abnormal.

Values for the parameter are:

```
ABNORMAL
NORMAL
```

TF_TOKEN
A token representing the terminal facility.

Output Parameters

REASON
The values for the parameter are:

```
INVALID_FORMAT
INVALID_FUNCTION
RESTART_FAILURE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFUS gate, CREATE_USER_EXTENSION function

The CREATE_USER_EXTENSION function of the TFUS gate is used to create a secure storage extension for a TCTTE.

Input Parameters

TCTTE_TOKEN
TCTTE

USERID
User ID

Output Parameters

REASON

The following value is returned when RESPONSE is EXCEPTION:

```
NOT_AUTHORIZED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_TCTTE
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TFUS gate, DELETE_USER_EXTENSION function

The DELETE_USER_EXTENSION function of the TFUS gate is used to delete a secure storage extension for a TCTTE.

Input Parameters

TCTTE_TOKEN

TCTTE

Output Parameters

REASON

The following value is returned when RESPONSE is EXCEPTION:

```
NOT_AUTHORIZED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSWM gate, XRF_GET function

This function reads security rebuild records from the CAVM data set. A response of `OK` indicates that a security rebuild is required in the Alternate. A response of `EXCEPTION` and a reason of `END_OF_DATA` indicates that tracking has finished and that takeover is beginning.

Output Parameters

REASON

The values for the parameter are:

```
END_OF_DATA
INVALID_DATA
SHUTDOWN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSWM gate, XRF_PUT function

This function writes security rebuild records to the CAVM data set. This informs the Alternate that a security rebuild is required.

Output Parameters

REASON

The values for the parameter are:

NOT_THERE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Application manager domain's call-back gates

Table 4 summarizes the domain's call-back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 4. Application domain's call-back gates

Gate	Trace	Function	Format
APRD	AP D610 AP D611	START_DELIVERY DELIVER_RECOVERY END_DELIVERY DELIVER_FORGET	RMDE
APRD	AP D610 AP D611	TAKE_KEYPOINT	RMKP
APRD	AP D610 AP D611	PERFORM_COMMIT PERFORM_PREPARE START_BACKOUT DELIVER_BACKOUT_DATA END_BACKOUT PERFORM_SHUNT PERFORM_UNSHUNT	RMRO
BRXM	AP 2860 AP 2861	INIT_XM_CLIENT BIND_XM_CLIENT TRANSACTION_HANG ABEND_TERMINATE RELEASE_XM_CLIENT	XMAC
ECRL	EC 3570 EC 3571	CREATE DISCARD INQUIRE SET	RLCB
ICRC		DELIVER_IC_RECOVERY_DATA SOLICIT_INQUIRES	TSIC

Application manager domain's generic gates

Table 5 on page 334 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 5. Application manager domain's generic gates

Gate	Trace	Function	Format
APDM	AP 0900 AP 0901	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
APDS	AP 0500 AP 0501	TASK_REPLY PURGE_INHIBIT_QUERY	DSAT
APST	AP D400 AP D401	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
APSM	AP F110 AP F111	STORAGE_NOTIFY	SMNT
APTI	AP F300 AP F301	NOTIFY	TISR

Application manager domain's generic formats

Table 6 describes the generic formats owned by the application domain and shows the functions performed on the calls.

Table 6. Application manager domain's generic formats

Format	Calling module	Function
APUE	DFHUEM	SET_EXIT_STATUS

Note: In the descriptions of the formats that follow, the input parameters are input not to the application domain, but to the domain being called by the application domain. Similarly, the output parameters are output by the domain that was called by the application domain, in response to the call.

APUE gate, SET_EXIT_STATUS function

Enable or disable a user exit point.

Input Parameters

EXIT_POINT

Identifies the user exit to be enabled or disabled

EXIT_STATUS

The intended status of the exit.

Values for the parameter are:

```
ACTIVE
INACTIVE
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_EXIT_POINT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Business application manager domain (BA)

The business application manager domain (also sometimes known as *business application manager*) is responsible for managing CICS business transaction services (BTS) processes, process types and activities. It deals with the hardening of the associated data to BTS repository files. Along with scheduler services domain and event manager domain it forms the CICS BTS function.

Business application manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the BA domain.

BAAC gate, ACQUIRE_ACTIVITY function

The ACQUIRE_ACTIVITY function of the BAAC gate is used to acquire the specified activity.

Input Parameters

ACTIVITYID
the buffer containing the activity identifier.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_ALREADY_ACQUIRED
ACTIVITY_NOT_FOUND
READ_FAILURE
RECORD_BUSY
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, ADD_ACTIVITY function

The ADD_ACTIVITY function of the BAAC gate is used to define a new activity in response to an EXEC CICS DEFINE ACTIVITY call.

Input Parameters

ACTIVITY_NAME
the 16-character activity name.

COMPLETION_EVENT
the 16-character completion event.

TRANID
the 4-character transaction id.

ACTIVITYID
Optional Parameter

the buffer containing the activity identifier.

PROGRAM
Optional Parameter

the 8-character program name associated with the root activity.

USERID
Optional Parameter

the 8-character userid.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
DUPLICATE_ACTIVITY_NAME
INVALID_NAME
NO_CURRENT_ACTIVITY
UNKNOWN_TRANSACTION_ID
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, ADD_REATTACH_ACQUIRED function

The ADD_REATTACH_ACQUIRED function of the BAAC gate is used to reattach an activity.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
NO_ACQUIRED_ACTIVITY
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, ADD_TIMER_REQUEST function

The ADD_TIMER_REQUEST function of the BAAC gate is used to add a delayed request to BAM domain in response to an EXEC CICS DEfINE TIMER call.

Input Parameters

DATETIME

the time at which the timer expires.

EVENT_VERSION

the version of the event.

REQUEST_TOKEN

the token representing the request.

TIMER_EVENT

the timer event name.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
NO_CURRENT_ACTIVITY
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, CANCEL_ACTIVITY function

The CANCEL_ACTIVITY function of the BAAC gate is used to synchronously cancel the named child activity or the acquired activity.

Input Parameters

ACTIVITY_NAME

Optional Parameter

the 16-character activity name.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
FILE_NOT_AUTH
INVALID_ACTIVITYID
INVALID_MODE
NO_CURRENT_ACTIVITY
RECORD_BUSY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BUFFER_LENGTH
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, CHECK_ACTIVITY function

The CHECK_ACTIVITY function of the BAAC gate is used to establish how the named child activity or acquired activity completed.

Input Parameters

ACTIVITY_NAME

Optional Parameter

the 16-character activity name.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
NO_CURRENT_ACTIVITY
READ_FAILURE
RECORD_BUSY
```

ABEND_CODE

the 4-character abend code.

ABEND_PROG

the 8-character name of the program which abended.

ACTMODE

the active mode of the process.

Values for the parameter are:

```
ACTIVE
CANCELLING
COMPLETE
DORMANT
INITIAL
```

COMPLETION_STATUS

is the completion status of the process.

Values for the parameter are:

```
ABENDED
FORCEDCOMPLETE
INCOMPLETE
NORMAL
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SUSPENDED
indicates whether the process is suspended.

Values for the parameter are:
NO
YES

BAAC gate, DELETE_ACTIVITY function

The DELETE_ACTIVITY function of the BAAC gate is used to delete the named child activity from the repository.

Input Parameters

ACTIVITY_NAME
the 16-character activity name.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ACTIVITY_NOT_FOUND
INVALID_MODE
NO_CURRENT_ACTIVITY
READ_FAILURE
RECORD_BUSY

ACTMODE
the active mode of the process.

Values for the parameter are:
ACTIVE
CANCELLING
COMPLETE
DORMANT
INITIAL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, LINK_ACTIVITY function

The LINK_PROCESS function of the BAAC gate is used to invoke the named child activity or acquired activity synchronously, without a context switch.

Input Parameters

INPUT_EVENT
the 16-character name of the input event.

ACTIVITY_NAME
Optional Parameter

the 16-character activity name.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ACTIVITY_NOT_FOUND
AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
AUTOSTART_DISABLED
INVALID_EVENT

INVALID_MODE
JVM_PROFILE_NOT_FOUND
JVM_PROFILE_NOT_VALID
JVMPOOL_DISABLED
NO_COMPLETION_EVENT
NO_CURRENT_ACTIVITY
NO_EVENTS_PROCESSED
PENDING_ACTIVITY_EVENTS
PROGRAM_NOT_AUTHORISED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
READ_FAILURE
RECORD_BUSY
REMOTE_PROGRAM
SECOND_H8_PROGRAM
SECOND_JVM_PROGRAM
SYSTEM_PROPERTIES_NOT_FND
USER_CLASS_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, RESET_ACTIVITY function

The RESET_ACTIVITY function of the BAAC gate is used to reset the state of the named child activity to initial, so it may be run again.

Input Parameters

ACTIVITY_NAME
the 16-character activity name.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ACTIVITY_NOT_FOUND
FILE_NOT_AUTH
INVALID_MODE
NO_CURRENT_ACTIVITY
READ_FAILURE
RECORD_BUSY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, RESUME_ACTIVITY function

The RESUME_ACTIVITY function of the BAAC gate is used to resume a previously suspended activity.

Input Parameters

ACTIVITY_NAME
Optional Parameter

the 16-character activity name.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

ACTIVITY_NOT_FOUND
INVALID_MODE
NO_ACQUIRED_ACTIVITY
READ_FAILURE
RECORD_BUSY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, RETURN_END_ACTIVITY function

The RETURN_END_ACTIVITY function of the BAAC gate is used to indicate the completion of the current activity and so raise the completion event.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NO_CURRENT_ACTIVITY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, RUN_ACTIVITY function

The RUN_ACTIVITY function of the BAAC gate is used to execute the named child activity or the acquired activity either asynchronously or synchronously i.e. with a context switch.

Input Parameters

INPUT_EVENT
the 16-character name of the input event.

MODE
Indicates if the activity should run asynchronously or synchronously.

Values for the parameter are:
ASYNC
SYNC

ACTIVITY_NAME
Optional Parameter

the 16-character activity name.

FACILITY_TOKEN
Optional Parameter

the 8-character facility token.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ACTIVITY_NOT_FOUND
ACTIVITY_SUSPENDED
INVALID_EVENT
INVALID_EVENT
INVALID_MODE
NO_COMPLETION_EVENT
NO_CURRENT_ACTIVITY
READ_FAILURE
RECORD_BUSY
REMOTE_PROGRAM
REMOTE_TRAN

RUN_SYNC_ABENDED
TRAN_NOT_AUTH

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAAC gate, SUSPEND_ACTIVITY function

The SUSPEND_ACTIVITY function of the BAAC gate is used to suspend the named child activity or the acquired activity.

Input Parameters

ACTIVITY_NAME
Optional Parameter

the 16-character activity name.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ACTIVITY_NOT_FOUND
INVALID_MODE
NO_ACQUIRED_ACTIVITY
READ_FAILURE
RECORD_BUSY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BABR gate, COMMIT_BROWSE function

The COMMIT_BROWSE function of the BABR gate is used to release any CICS BTS browses associated with this UOW.

Input Parameters

CHAIN_HEAD
pointer to the head of the browse chain.

BABR gate, ENDBR_ACTIVITY function

The ENDBR_ACTIVITY function of the BABR gate is used to end the specified activity browse.

Input Parameters

BROWSE_TOKEN
is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_BROWSE_TOKEN
INVALID_BROWSE_TYPE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BABR gate, ENDBR_CONTAINER function

The ENDBR_CONTAINER function of the BABR gate is used to end the specified container browse.

Input Parameters

BROWSE_TOKEN
is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_BROWSE_TOKEN
INVALID_BROWSE_TYPE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BABR gate, ENDBR_PROCESS function

The ENDBR_PROCESS function of the BABR gate is used to end the specified process browse.

Input Parameters

BROWSE_TOKEN
is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_BROWSE_TOKEN
INVALID_BROWSE_TYPE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BABR gate, GETNEXT_ACTIVITY function

The GETNEXT_ACTIVITY function of the BABR gate is used to return the next activity in the specified browse.

Input Parameters

BROWSE_TOKEN
is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

RETURNED_ACTIVITYID
Optional Parameter

is a buffer containing the activity identifier.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
INVALID_BROWSE_TOKEN
INVALID_BROWSE_TYPE
RECORD_BUSY

The following values are returned when RESPONSE is INVALID:
INVALID_BUFFER_LENGTH

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACTIVITY_NAME
Optional Parameter

is the 16-character activity name.

LEVEL

Optional Parameter

is the level into the activity tree.

BABR gate, GETNEXT_CONTAINER function

The GETNEXT_CONTAINER function of the BABR gate is used to return the next container in the specified browse.

Input Parameters

BROWSE_TOKEN

is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
INVALID_BROWSE_TOKEN
INVALID_BROWSE_TYPE
RECORD_BUSY
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONTAINER_NAME

Optional Parameter

is the 16-character container name.

BABR gate, GETNEXT_PROCESS function

The GETNEXT_PROCESS function of the BABR gate is used to return the next process in the specified browse.

Input Parameters

BROWSE_TOKEN

is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

RETURNED_ACTIVITYID

Optional Parameter

is a buffer containing the activity identifier.

RETURNED_PROCESS_NAME

Optional Parameter

is a buffer containing the returned process name.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
INVALID_BROWSE_TOKEN
INVALID_BROWSE_TYPE
RECORD_BUSY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BUFFER_LENGTH
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BABR gate, INQUIRE_ACTIVATION function

The INQUIRE_ACTIVATION function of the BABR gate is used to obtain information about the activation associated with a running transaction, if there is one.

Input Parameters

RETURNED_ACTIVITYID
is a buffer containing the activity identifier.

RETURNED_PROCESS_NAME
is a buffer containing the returned process name.

TRANSACTION_TOKEN
is a token representing an instance of a transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BUFFER_LENGTH
```

ACTIVITY_NAME
is the 16-character activity name.

PROCESS_TYPE
is the 8-character process type.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BABR gate, INQUIRE_ACTIVITY function

The INQUIRE_ACTIVITY function of the BABR gate is used to obtain information about the specified activity.

Input Parameters

ACTIVITYID
Optional Parameter

the buffer containing the activity identifier.

RETURNED_ACTIVITYID
Optional Parameter

is a buffer containing the activity identifier.

RETURNED_PROCESS_NAME
Optional Parameter

is a buffer containing the returned process name.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
FILE_NOT_AUTH
NO_CURRENT_ACTIVITY
RECORD_BUSY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_ACTIVITYID_LEN
INVALID_BUFFER_LENGTH
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ABEND_CODE
Optional Parameter

the 4-character abend code.

ABEND_PROGRAM
Optional Parameter

the 8-character name of the program which abended.

ACTIVITY_NAME
Optional Parameter

is the 16-character activity name.

COMPLETION_STATUS
Optional Parameter

is the completion status of the process.

Values for the parameter are:
```
ABENDED
FORCED
INCOMPLETE
NORMAL
```

EVENT_NAME
Optional Parameter

is the 16-character event name.

INIT_TRANSID
Optional Parameter

is the 4-character transaction identifier of the transaction under which the activity was initiated.

MODE
Optional Parameter

is the mode of the activity.

Values for the parameter are:
```
ACTIVE
CANCELLING
COMPLETE
DORMANT
INITIAL
```

PROCESS_TYPE
Optional Parameter

is the 8-character process type.

PROGRAM
Optional Parameter

is the 8-character program name.

SUSPENDED
Optional Parameter

indicates whether the process is suspended.

Values for the parameter are:
```
NO
YES
```

TRANSID
Optional Parameter

is the 4-character transaction identifier.

USERID
Optional Parameter

is the 8-character userid.

BABR gate, INQUIRE_CONTAINER function

The INQUIRE_CONTAINER function of the BABR gate is used to obtain information about the specified container.

Input Parameters

CONTAINER_NAME
the 16-character container name.

ACTIVITYID
Optional Parameter

the buffer containing the activity identifier.

PROCESS_NAME
Optional Parameter

the 36-character process name.

PROCESS_TYPE
Optional Parameter

is the 8-character process type.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
CONTAINER_NOT_FOUND
FILE_NOT_AUTH
NO_CURRENT_ACTIVITY
PROCESS_NOT_FOUND
PROCESSTYPE_NOT_FOUND
RECORD_BUSY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_ACTIVITYID_LEN
INVALID_PROCESSNAME_LEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DATA_ADDRESS
Optional Parameter

is the address of the container data.

DATA_LENGTH
Optional Parameter

is the length of the container data.

BABR gate, INQUIRE_PROCESS function

The INQUIRE_PROCESS function of the BABR gate is used to obtain information about the specified process.

Input Parameters

PROCESS_NAME
the 36-character process name.

PROCESS_TYPE
is the 8-character process type.

RETURNED_ACTIVITYID
Optional Parameter

is a buffer containing the activity identifier.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
FILE_NOT_AUTH
PROCESS_NOT_FOUND
PROCESSTYPE_NOT_FOUND
RECORD_BUSY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BUFFER_LENGTH
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BABR gate, STARTBR_ACTIVITY function

The STARTBR_ACTIVITY function of the BABR gate is used to initiate a browse of activities from the specified activity identifier or from the root activity of the specified process.

Input Parameters

ACTIVITYID
Optional Parameter

the buffer containing the activity identifier.

PROCESS_NAME
Optional Parameter

the 36-character process name.

PROCESS_TYPE
Optional Parameter

is the 8-character process type.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
FILE_NOT_AUTH
NO_CURRENT_ACTIVITY
PROCESS_NOT_FOUND
PROCESSTYPE_NOT_FOUND
RECORD_BUSY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_ACTIVITYID_LEN
INVALID_PROCESSNAME_LEN
```

BROWSE_TOKEN
is the token used to identify this browse.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BABR gate, STARTBR_CONTAINER function

The STARTBR_CONTAINER function of the BABR gate is used to initiate a browse of containers associated with a specified activity or process.

Input Parameters

ACTIVITYID
Optional Parameter

the buffer containing the activity identifier.

PROCESS_NAME
Optional Parameter

the 36-character process name.

PROCESS_TYPE
Optional Parameter

is the 8-character process type.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
FILE_NOT_AUTH
NO_CURRENT_ACTIVITY
PROCESS_NOT_FOUND
PROCESSTYPE_NOT_FOUND
RECORD_BUSY
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_ACTIVITYID_LEN
INVALID_PROCESSNAME_LEN
```

BROWSE_TOKEN
is the token used to identify this browse.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BABR gate, STARTBR_PROCESS function

The STARTBR_PROCESS function of the BABR gate is used to initiate a browse of the processes of a certain type.

Input Parameters

PROCESS_TYPE
is the 8-character process type.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
FILE_NOT_AUTH
FILE_UNAVAILABLE
PROCESSTYPE_NOT_FOUND
RECORD_BUSY
```

BROWSE_TOKEN
is the token used to identify this browse.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BACR gate, COPY_CONTAINER function

Copy a container from one activity to another.

Input Parameters

CONTAINER_NAME
is the 16-character source container name.

ACTIVITY_NAME
Optional Parameter

is the 16-character name of the activity with which the source container is associated.

AS_CONTAINER
Optional Parameter

is the 16-character destination container name.

CONTAINER_SCOPE
Optional Parameter

identifies the scope of the source container.

Values for the parameter are:

```
ACQUIRED_ACTIVITY
ACQUIRED_PROCESS
ACTIVITY
CHILD_ACTIVITY
PROCESS
```

TO_ACTIVITY
Optional Parameter

s the 16-character activity name of the activity with which the destination container is associated.

TO_PROCESS
Optional Parameter

is a value indicating if the destination container is to be a process container rather than an activity container.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_READONLY
INVALID_CONTAINER_NAME
NO_ACQUIRED_ACTIVITY
NO_ACQUIRED_PROCESS
NO_CURRENT_ACTIVITY
NO_CURRENT_PROCESS
RECORD_BUSY
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BACR gate, DELETE_CONTAINER function

The DELETE_CONTAINER function of the BACR gate is used to delete a named container and its associated data.

Input Parameters

CONTAINER_NAME
the 16-character container name.

ACTIVITY_NAME
Optional Parameter

the 16-character activity name.

CONTAINER_SCOPE
Optional Parameter

identifies the scope of this container.

Values for the parameter are:

```
ACQUIRED_ACTIVITY
ACQUIRED_PROCESS
ACTIVITY
CHILD_ACTIVITY
PROCESS
```

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_READONLY
NO_ACQUIRED_ACTIVITY
NO_ACQUIRED_PROCESS
NO_CURRENT_ACTIVITY
NO_CURRENT_PROCESS
RECORD_BUSY
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BACR gate, GET_CONTAINER_INTO function

The GET_CONTAINER_INTO function of the BACR gate is used to place the data in a named container into an area provided by the caller.

Input Parameters

CONTAINER_NAME
the 16-character container name.

ITEM_BUFFER
is the buffer into which the container data is placed.

ACTIVITY_NAME
Optional Parameter

the 16-character activity name.

CONTAINER_SCOPE
Optional Parameter

identifies the scope of this container.

Values for the parameter are:

```
ACQUIRED_ACTIVITY
ACQUIRED_PROCESS
```

ACTIVITY
CHILD_ACTIVITY
PROCESS

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ACTIVITY_NOT_FOUND
CONTAINER_NOT_FOUND
LENGTH_ERROR
NO_ACQUIRED_ACTIVITY
NO_ACQUIRED_PROCESS
NO_CURRENT_ACTIVITY
NO_CURRENT_PROCESS
RECORD_BUSY

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BACR gate, GET_CONTAINER_LENGTH function

The GET_CONTAINER_LENGTH function of the BACR gate is used to query the length of application data in a named container.

Input Parameters

CONTAINER_NAME

the 16-character container name.

ACTIVITY_NAME

Optional Parameter

the 16-character activity name.

CONTAINER_SCOPE

Optional Parameter

identifies the scope of this container.

Values for the parameter are:

ACQUIRED_ACTIVITY
ACQUIRED_PROCESS
ACTIVITY
CHILD_ACTIVITY
PROCESS

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ACTIVITY_NOT_FOUND
CONTAINER_NOT_FOUND
INVALID_CONTAINER_NAME
NO_ACQUIRED_ACTIVITY
NO_ACQUIRED_PROCESS
NO_CURRENT_ACTIVITY
NO_CURRENT_PROCESS
RECORD_BUSY

CONTAINER_LENGTH

is the fullword length of the application data.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BACR gate, GET_CONTAINER_SET function

The GET_CONTAINER_SET function of the BACR gate is used to place the data in a named container into an area provided by BAM domain and return this area to the caller.

Input Parameters

CONTAINER_NAME
the 16-character container name.

ACTIVITY_NAME
Optional Parameter

the 16-character activity name.

CONTAINER_SCOPE
Optional Parameter

identifies the scope of this container.

Values for the parameter are:

```
ACQUIRED_ACTIVITY
ACQUIRED_PROCESS
ACTIVITY
CHILD_ACTIVITY
PROCESS
```

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
CONTAINER_NOT_FOUND
NO_ACQUIRED_ACTIVITY
NO_ACQUIRED_PROCESS
NO_CURRENT_ACTIVITY
NO_CURRENT_PROCESS
RECORD_BUSY
```

ITEM_DATA
a block holding the named container's data.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BACR gate, MOVE_CONTAINER function

The MOVE_CONTAINER function of the BACM gate is used to move a container between activities. If a container of the same name as the destination container name already exists in the destination activity then it is overwritten.

Input Parameters

CONTAINER_NAME
is the 16-character source container name.

ACTIVITY_NAME
Optional Parameter

is the 16-character name of the activity with which the source container is associated.

AS_CONTAINER
Optional Parameter

is the 16-character destination container name.

CONTAINER_SCOPE
Optional Parameter

identifies the scope of the source container.

Values for the parameter are:
ACQUIRED_ACTIVITY
ACQUIRED_PROCESS
ACTIVITY
CHILD_ACTIVITY
PROCESS

TO_ACTIVITY
Optional Parameter

is the 16-character activity name of the activity with which the destination container is associated.

TO_PROCESS
Optional Parameter

is a value indicating if the destination container is to be a process container rather than an activity container.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ACTIVITY_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_READONLY
INVALID_CONTAINER_NAME
NO_ACQUIRED_ACTIVITY
NO_ACQUIRED_PROCESS
NO_CURRENT_ACTIVITY
NO_CURRENT_PROCESS
RECORD_BUSY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BACR gate, PUT_CONTAINER function

The PUT_CONTAINER function of the BACR gate is used to place data into a named container.

Input Parameters

CONTAINER_NAME
the 16-character container name.

ITEM_DATA
a block holding the data to be placed in the named container.

ACTIVITY_NAME
Optional Parameter

the 16-character activity name.

CONTAINER_SCOPE
Optional Parameter

identifies the scope of this container.

Values for the parameter are:
ACQUIRED_ACTIVITY
ACQUIRED_PROCESS
ACTIVITY
CHILD_ACTIVITY
PROCESS

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
ACTIVITY_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_READONLY
INVALID_CONTAINER_NAME
LENGTH_ERROR
NO_ACQUIRED_ACTIVITY
NO_ACQUIRED_PROCESS
NO_CURRENT_ACTIVITY
NO_CURRENT_PROCESS
RECORD_BUSY

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAPR gate, ACQUIRE_PROCESS function

The ACQUIRE_PROCESS function of the BAPR gate is used to acquire the named process.

Input Parameters

PROCESS_NAME

the 36-character process name.

PROCESSTYPE

the 8-character process type.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
FILE_NOT_AUTH
OTHER_PROCESS_CURRENT
PROCESS_NOT_FOUND
PROCESSTYPE_NOT_FOUND
RECORD_BUSY

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAPR gate, ADD_PROCESS function

The ADD_PROCESS function of the BAPR gate is used to define a new process in reponse to an EXEC CICS(R) DEFINE PROCESS call.

Input Parameters

PROCESS_NAME

the 36-character process name.

PROCESSTYPE

the 8-character process type.

TRANID
the 4-character transaction id.

CHECK_UNIQUE
Optional Parameter

a Boolean value indicating whether a check should be made to ensure that the process name is unique within the scope of the process-type.

Values for the parameter are:
NO
YES

PROGRAM
Optional Parameter

the 8-character program name associated with the root activity.

USERID
Optional Parameter

the 8-character userid.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DUPLICATE_PROCESS_NAME
FILE_NOT_AUTH
PROCESS_ALREADY_ACQUIRED
PROCESSTYPE_NOT_ENABLED
PROCESSTYPE_NOT_FOUND
WRITE_FAILED

PROCESS_TOKEN
a token representing this process internally.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAPR gate, CANCEL_PROCESS function

The CANCEL_PROCESS function of the BAPR gate is used to synchronously cancel the acquired process.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
FILE_NOT_AUTH
PROCESS_NOT_FOUND
PROCESSTYPE_NOT_FOUND
RECORD_BUSY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAPR gate, CHECK_PROCESS function

The CHECK_PROCESS function of the BAPR gate is used to establish how the acquired process completed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
PROCESS_NOT_FOUND

RECORD_BUSY

ABEND_CODE
the 4-character abend code.

ABEND_PROG
the 8-character name of the program which abended.

ACTMODE
the active mode of the process.

Values for the parameter are:
ACTIVE
CANCELLING
COMPLETE
DORMANT
INITIAL

COMPLETION_STATUS
is the completion status of the process.

Values for the parameter are:
ABENDED
FORCEDCOMPLETE
INCOMPLETE
NORMAL

RESPONSE
is the domain's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

SUSPENDED
indicates whether the process is suspended.

Values for the parameter are:
NO
YES

BAPR gate, LINK_PROCESS function

The LINK_PROCESS function of the BAPR gate is used to invoke the acquired process synchronously, without a context switch.

Input Parameters

INPUT_EVENT
Optional Parameter

the 16-character name of the input event.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
AUTOSTART_DISABLED
INVALID_EVENT
INVALID_MODE

JVM_PROFILE_NOT_FOUND
JVM_PROFILE_NOT_VALID
JVMPOOL_DISABLED
NO_EVENTS_PROCESSED
OTHER_PROCESS_CURRENT
PENDING_ACTIVITY_EVENTS
PROCESS_NOT_FOUND
PROCESS_SUSPENDED
PROCESSTYPE_NOT_FOUND
PROGRAM_NOT_AUTHORISED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
REMOTE_PROGRAM
SECOND_H8_PROGRAM
SECOND_JVM_PROGRAM
SYSTEM_PROPERTIES_NOT_FND
USER_CLASS_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAPR gate, RESET_PROCESS function

The RESET_PROCESS function of the BAPR gate is used to reset the state of the acquired root activity to initial, so it may be run again.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
FILE_NOT_AUTH
INVALID_MODE
PROCESS_NOT_FOUND
PROCESSTYPE_NOT_FOUND
RECORD_BUSY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAPR gate, RESUME_PROCESS function

The RESUME_PROCESS function of the BAPR gate is used to resume a previously suspended process.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
PROCESS_NOT_FOUND
RECORD_BUSY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAPR gate, RUN_PROCESS function

The RUN_PROCESS function of the BAPR gate is used to execute the acquired process (invoke the root activity), either asynchronously or synchronously i.e. with a context switch.

Input Parameters

MODE

Indicates if the process should run asynchronously or synchronously.

Values for the parameter are:

ASYNC
SYNC

FACILITY_TOKEN

Optional Parameter

the 8-character facility token.

INPUT_EVENT

Optional Parameter

the 16-character name of the input event.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
AUTOSTART_DISABLED
INVALID_EVENT
INVALID_MODE
JVM_PROFILE_NOT_FOUND
JVM_PROFILE_NOT_VALID
JVMPOOL_DISABLED
OTHER_PROCESS_CURRENT
PROCESS_NOT_FOUND
PROCESS_SUSPENDED
PROCESSTYPE_NOT_FOUND
PROGRAM_NOT_AUTHORISED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
RECORD_BUSY
REMOTE_PROGRAM
REMOTE_TRAN
RUN_SYNC_ABENDED
SECOND_H8_PROGRAM
SECOND_JVM_PROGRAM
SYSTEM_PROPERTIES_NOT_FND
TRAN_NOT_AUTH
USER_CLASS_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAPR gate, SUSPEND_PROCESS function

The SUSPEND_PROCESS function of the BAPR gate is used to suspend the acquired process.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

PROCESS_NOT_FOUND

RECORD_BUSY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BATT gate, ADD_REPLACE_PROCESSTYPE function

The ADD_REPLACE_PROCESSTYPE function of the BATT gate is used to add a new process type definition or replace an existing process type definition. Process types are defined using RDO.

Input Parameters

AUDITLEVEL
determines the level of auditing to be undertaken for this process type.

Values for the parameter are:
ACTIVITY
FULL
OFF
PROCESS

AUDITLOG_NAME
is an 8-character name of the audit log to be associated with this process type. The log is defined using RDO.

CATALOG_PTDEF
indicates whether the definition should be written to the global catalog.

Values for the parameter are:
NO
YES

FILE_NAME
is an 8-character name of the repository file to be associated with this process type. The file is defined using RDO.

PROCESSTYPE_NAME
is an 8-character name.

STATUS
indicates whether the process type definition should be installed in a disabled or enabled state.

Values for the parameter are:
DISABLED
ENABLED

USERRECORDS
indicates whether user audit records are to be written to the log.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
NOT_DISABLED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BATT gate, COMMIT_PROCESSTYPE_TABLE function

The COMMIT_PROCESSTYPE_TABLE function of the BATT gate is used to commit the process type definitions to the global catalog.

Input Parameters

TOKEN
is the token identifying the table of process type definitions.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BATT gate, DISCARD_PROCESSTYPE function

The DISCARD_PROCESSTYPE function of the BATT gate is used to discard the named processtype definition.

Input Parameters

PROCESSTYPE_NAME
is an 8-character name.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
ENTRY_NOT_FOUND
NOT_DISABLED
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BATT gate, END_BROWSE_PROCESSTYPE function

The END_BROWSE_PROCESSTYPE function of the BATT gate is used to end the browse identified by the browse token.

Input Parameters

BROWSE_TOKEN
is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BATT gate, GET_NEXT_PROCESSTYPE function

The GET_NEXT_PROCESSTYPE function of the BATT gate is used to return the name of the next process type in the browse, identified by the browse token.

Input Parameters

BROWSE_TOKEN
is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
NO_MORE_DATA_AVAILABLE
```

PROCESSTYPE_NAME
the 8-character process type name.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BATT gate, INQUIRE_PROCESSTYPE function

The INQUIRE_PROCESSTYPE function of the BATT gate is used to return information on the named process type.

Input Parameters

PROCESSTYPE_NAME
is an 8-character name.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ENTRY_NOT_FOUND
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
AUDITLEVEL
Optional Parameter

identifies the level of auditing for this process type.

Values for the parameter are:
ACTIVITY
FULL
OFF
PROCESS
AUDITLOG_NAME
Optional Parameter

is an 8-character name of the audit log associated with this process type.
FILE_NAME
Optional Parameter

is the 8-character name of the repository file associated with this process type.
STATUS
Optional Parameter

indicates the status of the process type.

Values for the parameter are:
DISABLED
ENABLED
USERRECORDS
Optional Parameter

indicates whether user audit records are to being written to the log.

Values for the parameter are:
NO
YES

BATT gate, SET_PROCESSTYPE function

The SET_PROCESSTYPE function of the BATT gate is used to alter the named processtype definition.

Input Parameters

PROCESSTYPE_NAME
is an 8-character name.

AUDITLEVEL
Optional Parameter

determines the level of auditing to be undertaken for this process type.

Values for the parameter are:
```
ACTIVITY
FULL
OFF
PROCESS
```

STATUS
Optional Parameter

indicates whether the process type definition should be installed in a disabled or enabled state.

Values for the parameter are:
```
DISABLED
ENABLED
```

USERRECORDS
Optional Parameter

indicates whether user audit records are to be written to the log.

Values for the parameter are:
```
NO
YES
```

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
ENTRY_NOT_FOUND
NOT_DISABLED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BATT gate, START_BROWSE_PROCESSTYPE function

The START_BROWSE_PROCESSTYPE function of the BATT gate is used to initiate a browse of the process types known to this region.

Output Parameters

BROWSE_TOKEN
is the token used to identify this browse.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BAXM gate, BIND_ACTIVITY_REQUEST function

The BIND_ACTIVITY_REQUEST function of the BAXM gate is used to make the current UOW an activation of the activity specified in the activity request. This activation could be used to mark the activity complete abended because the previous activation failed, hence the abend information.

Input Parameters

REQUEST_BLOCK
a block used to hold the request data.

ABEND_CODE
Optional Parameter

the 4-character abend code.

ABEND_MSG
Optional Parameter

the 6-character abend message number.

ABEND_PROG
Optional Parameter

the 8-character abend program name.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ACTIVITY_NOT_FOUND
READ_FAILURE
TIMEOUT
```

PROGRAM
is the 8-character program name.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RUN_PROGRAM
is used to indicate if a program is to be invoked on the program manager INITIAL_LINK.

Values for the parameter are:

```
NO
YES
```

BAXM gate, INIT_ACTIVITY_REQUEST function

The INIT_ACTIVITY_REQUEST function of the BAXM gate is used when the transaction requires a 3270 bridge facility, in which case the named bridge exit program is invoked.

Input Parameters

BRIDGE_EXIT
the 8-character name of the bridge exit program.

REQUEST_BLOCK
a block used to hold the request data.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Business application manager domain's generic gates

Table 7 on page 364 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 7. Business application manager domain's generic gates

Gate	Trace	Function	Format
APUE	BA 0180 BA 0181	SET_EXIT_STATUS	APUE
DMDM	BA 0101 BA 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

Business application manager domain's call-back gates

Table 8 summarizes the domain's call-back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 8. Business application manager domain's call-back gates

Gate	Trace	Function	Format
RMDE	BA 0140 BA 0141	START_DELIVERY DELIVER_RECOVERY END_DELIVERY	RMDE
RMKP	BA 0140 BA 0141	TAKE_KEYPOINT	RMKP
RMRO	BA 0140 BA 0141	PERFORM_PREPARE PERFORM_COMMIT START_BACKOUT DELIVER_BACKOUT_DATA END_BACKOUT PERFORM_SHUNT PERFORM_UNSHUNT	RMRO

Business application manager domain's generic formats

Table 9 describes the generic formats owned by the domain and shows the functions performed on the calls.

Table 9. Business application manager domain's generic formats

Format	Calling module	Function
BAGD		INQUIRE_DATA_LENGTH GET_DATA DESTROY_TOKEN

Note: In the descriptions of the formats that follow, the input parameters are input not to the business application manager domain, but to the domain being called by the business application manager domain. Similarly, the output parameters are output by the domain that was called by the business application manager domain, in response to the call.

Modules

Module	Function
DFHBAAC	DFHBAAC is the gate module for the following requests: ADD_ACTIVITY RUN_ACTIVITY CHECK_ACTIVITY RETURN_END_ACTIVITY DELETE_ACTIVITY SUSPEND_ACTIVITY RESUME_ACTIVITY CANCEL_ACTIVITY sliNK_ACTIVITY ACQUIRE_ACTIVITY RESET_ACTIVITY ADD_TIMER_REQUEST ADD_REATTACH_ACQUIRED
DFHBAAC0	Implements general activity class methods.
DFHBAAC1	Initializes the activity class.
DFHBAAC2	Implements the prepare method of the activity class.
DFHBAAC3	Implements the commit method of the activity class.
DFHBAAC4	Implements the delete method of the activity class.
DFHBAAC5	Implements the set_complete method of the activity class.
DFHBAAC6	Implements the invoke_exit method of the activity class.
DFHBAAR1	Intialises the audit class.
DFHBAAR2	Implements the write method of the audit class.
DFHBAA10	Implements the read_activity method of the activity class.
DFHBAA11	Implements the get_activity_instance method of the activity class.
DFHBAA12	Implements the run_sync method of the activity class.
DFHBABR	DFHBABR is the gate module for the following requests: STARTBR_ACTIVITY GETNEXT_ACTIVITY ENDBR_ACTIVITY INQUIRE_ACTIVITY STARTBR_CONTAINER GETNEXT_CONTAINER ENDBR_CONTAINER INQUIRE_CONTAINER STARTBR_PROCESS GETNEXT_PROCESS ENDBR_PROCESS INQUIRE_PROCESS INQUIRE_ACTIVATION COMMIT_BROWSE
DFHBABU1	Initializes the buffer class.
DFHBACO1	Initialization of the BAAC class: obtains and initializes the class data and sets its address into the BADM object.
DFHBACR	DFHBACR is the gate module for the following requests: DELETE_CONTAINER GET_CONTAINER_INTO GET_CONTAINER_SET PUT_CONTAINER

Module	Function
DFHBADM	DFHBADM is the gate module for the following requests: PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHBADUF	Formats the BAM domain control blocks
DFHBADU1	Formats the BAM domain control blocks
DFHBALR2	Implements the create_key method of the logical record class.
DFHBALR3	Implements the write_buffer method of the logical record class.
DFHBALR4	Implements the read_key method of the logical record class.
DFHBALR5	Implements the read_record method of the logical record class.
DFHBALR6	Implements the delete_record method of the logical record class.
DFHBALR7	Implements the get_browse_token method of the logical record class.
DFHBALR8	Implements the read_next_record method of the logical record class.
DFHBALR9	Implements the release_browse_token method of the logical record class.
DFHBAOFI	Initialises the object factory class.
DFHBAPR	DFHBAPR is the gate module for the following requests: ADD_PROCESS RUN_PROCESS CHECK_PROCESS SUSPEND_PROCESS RESUME_PROCESS CANCEL_PROCESS LINK_PROCESS ACQUIRE_PROCESS RESET_PROCESS
DFHBAPR0	Implements general process class methods.
DFHBAPT1	Initialises the processtype class.
DFHBAPT2	Implements the rebuild_table method of the processtype class.
DFHBAPT3	Implements the purge_catalog method of the processtype class.
DFHBARUC	The BTS repository utility program.
DFHBARUD	The BTS repository utility program.
DFHBARUP	The BTS repository utility program.
DFHBASP	DFHBASP is the gate module for the following requests: PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT START_BACKOUT DELIVER_BACKOUT_DATA END_BACKOUT START_RECOVERY DELIVER_RECOVERY END_RECOVERY TAKE_KEYPOINT
DFHBATRI	Interprets BAM domain trace entries

Module	Function
DFHBATT	DFHBATT is the gate module for the following requests: ADD_REPLACE_PROCESSTYPE INQUIRE_PROCESSTYPE START_BROWSE_PROCESSTYPE GET_NEXT_PROCESSTYPE END_BROWSE_PROCESSTYPE DISCARD_PROCESSTYPE COMMIT_PROCESSTYPE_TABLE
DFHBAUE	DFHBAUE is the gate module for the following requests: SET_EXIT_STATUS
DFHBAVP1	Initialises the variable length subpool class.
DFHBAXM	DFHBAXM is the gate module for the following requests: INIT_ACTIVITY_REQUEST BIND_ACTIVITY_REQUEST

Exits

There are two user exit points in BAM domain, XRSINDI and XBADEACT.

CICS catalog domain (CC)

The catalog domain manages the global and local catalog.

CICS catalog domain's specific gates

The specific gates provide access for other domains to functions that are provided by the CC domain.

CCCC gate, ADD function

The ADD function of the CCCC gate is used to add a record.

Input Parameters

DATA_IN
is the data to be added to the record.

NAME
is used to construct a record key, together with the domain and the type.

TYPE
identifies a block of data.

Output Parameters

REASON
The values for the parameter are:

```
CATALOG_FULL
DUPLICATE
INVALID_DATA_LENGTH
IO_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, DELETE function

The DELETE function of the CCCC gate is used to delete a record.

Input Parameters

NAME
is used to construct a record key, together with the domain and the type.

TYPE
identifies a block of data.

WRITE_TOKEN
Optional Parameter

is an optional token corresponding to a START_WRITE. This avoids the need for additional connects or disconnects.

Output Parameters

REASON
The values for the parameter are:

```
BAD_TOKEN
IO_ERROR
RECORD_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, END_BROWSE function

The END_BROWSE function of the CCCC gate is used to end a browse session.

Input Parameters

BROWSE_TOKEN
is the token identifying this browse session.

Output Parameters

REASON
The values for the parameter are:

```
BAD_TOKEN
IO_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, END_WRITE function

The END_WRITE function of the CCCC gate is used to end a write session.

Input Parameters

WRITE_TOKEN
is an optional token corresponding to a START_WRITE. This avoids the need for additional connects or disconnects.

Output Parameters

REASON
The values for the parameter are:

```
BAD_TOKEN
IO_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, GET function

The GET function of the CCCC gate is used to get a record.

Input Parameters

DATA_OUT
If the response is OK, this contains a copy of the specified record.

NAME
is used to construct a record key, together with the domain and the type.

TYPE
identifies a block of data.

Output Parameters

REASON
The values for the parameter are:

```
INVALID_DATA_LENGTH
IO_ERROR
RECORD_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, GET_NEXT function

The GET_NEXT function of the CCCC gate is used to get the next record.

Input Parameters

BROWSE_TOKEN
is the token identifying this browse session.

DATA_OUT
If the response is OK, this contains a copy of the specified record.

Output Parameters

REASON
The values for the parameter are:

```
BAD_TOKEN
BROWSE_END
INVALID_DATA_LENGTH
IO_ERROR
```

NAME_OUT
The name that was supplied when the record was created.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, GET_UPDATE function

The GET_UPDATE function of the CCCC gate is used to get a record and to establish a thread. This thread, identified by a token, is used in a corresponding PUT_REPLACE.

Input Parameters

DATA_OUT
If the response is OK, this contains a copy of the specified record.

NAME
is used to construct a record key, together with the domain and the type.

TYPE
identifies a block of data.

Output Parameters

REASON
The values for the parameter are:

INVALID_DATA_LENGTH
IO_ERROR
RECORD_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

UPDATE_TOKEN
Token to be used by the corresponding PUT_REPLACE.

CCCC gate, PUT_REPLACE function

The PUT_REPLACE function of the CCCC gate is used to replace a record.

Input Parameters

DATA_IN
is the data to be added to the record.

UPDATE_TOKEN
is the token obtained from a previous GET_UPDATE, used to identify an existing record in the catalog.

Output Parameters

REASON
The values for the parameter are:
BAD_TOKEN
CATALOG_FULL
INVALID_DATA_LENGTH
IO_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, START_BROWSE function

The START_BROWSE function of the CCCC gate is used to start a browse session.

Input Parameters

TYPE
identifies a block of data.

Output Parameters

REASON
The values for the parameter are:
IO_ERROR

BROWSE_TOKEN
is the token identifying this browse session.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, START_WRITE function

The START_WRITE function of the CCCC gate is used to start a write session.

Output Parameters

REASON
The values for the parameter are:
IO_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WRITE_TOKEN
is the token identifying a unique file string (thread).

CCCC gate, STARTUP_CLOSE function

Close the thread that is used for catalog domain requests during startup.

Output Parameters

REASON
The values for the parameter are:
NO_STARTUP_OPEN
NOT_FOR_LCD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, STARTUP_OPEN function

Open a thread that is used for catalog domain requests during startup.

Output Parameters

REASON
The values for the parameter are:
NOT_FOR_LCD
THREAD_IN_USE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, TYPE_PURGE function

The TYPE_PURGE function of the CCCC gate is used to purge records. This deletes all records within the specified TYPE block for that domain.

Input Parameters

TYPE
identifies a block of data.

Output Parameters

REASON
The values for the parameter are:
IO_ERROR
TYPE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, WRITE function

The WRITE function of the CCCC gate is used to write a record.

Input Parameters

DATA_IN
is the data to be added to the record.

NAME
is used to construct a record key, together with the domain and the type.

TYPE
identifies a block of data.

Output Parameters

REASON
The values for the parameter are:
BAD_TOKEN
CATALOG_FULL
INVALID_DATA_LENGTH
IO_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCCC gate, WRITE_NEXT function

The WRITE_NEXT function of the CCCC gate is used to write the next record.

Input Parameters

DATA_IN
is the data to be added to the record.

NAME
is used to construct a record key, together with the domain and the type.

TYPE
identifies a block of data.

WRITE_TOKEN
is an optional token corresponding to a START_WRITE. This avoids the need for additional connects or disconnects.

Output Parameters

REASON
The values for the parameter are:
BAD_TOKEN
CATALOG_FULL
INVALID_DATA_LENGTH
IO_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CICS catalog domain's generic gates

Table 10 on page 373 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 10. CICS catalog domain's generic gates

Gate	Trace	Function	Format
DMDM	Global catalog domain: GC 1010 GC 1040 Local catalog domain LC 1010 LC 1040	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

In preinitialization processing, the local catalog domain opens the CICS local catalog, DFHLCD. There is no preinitialization processing for the global catalog domain.

In initialization processing, the global catalog domain opens the CICS global catalog, DFHGCD.

In quiesce processing, the local and global catalog domains close their respective catalog data sets.

In termination processing, the CICS catalog domains perform no termination processing. They do not close either the local catalog or the global catalog; the operating system closes these data sets.

Modules

Module	Function
DFHCCCC	Handles the following functions: ADD DELETE GET WRITE GET_UPDATE PUT_REPLACE START_BROWSE GET_NEXT END_BROWSE TYPE_PURGE START_WRITE WRITE_NEXT END_WRITE
DFHCCDM	Handles the initialization and termination of the CICS catalog domains.
DFHCCDUF	Catalog dump formatting routine.
DFHCCTRI	Trace interpreter routine for the catalog domains.
DFHCCUTL	Offline utility to initialize the local catalog.

Directory manager domain (DD)

The directory manager domain manages directories of named tokens.

Directory manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the DD domain.

DDAP gate, BIND_LDAP function

The BIND_LDAP function of the DDAP gate establishes a session with an LDAP server.

Input Parameters

CACHE_SIZE
Optional parameter

a fullword that specifies the number of bytes available for caching LDAP search results. A value of zero indicates an unlimited cache size. If CACHE_SIZE is specified, CACHE_TIME_LIMIT must also be specified. If neither parameter is specified, results will not be cached.

CACHE_TIME_LIMIT
Optional parameter

a fullword that specifies the amount of time (in seconds) that LDAP search results are cached. A value of zero indicates an unlimited cache time limit.

DISTINGUISHED_NAME
specifies the location of the LDAP distinguished name, of the user permitted to bind to the chosen server. The block-descriptor is two fullwords of data, in which the first word contains the address of the data, and the second word contains the length in bytes of the data.

LDAP_BIND_PROFILE
specifies the location of the name of a RACF profile in the LDAPBIND class that contains the URL and credentials for the LDAP server being accessed. The block-descriptor is two fullwords of data, in which the first word contains the address of the data, and the second word contains the length in bytes of the data.

LDAP_SERVER_URL
specifies the location of the LDAP URL (in the format ldap://server:port) of the LDAP server being accessed. If the colon and port number are omitted, the port defaults to 389. The block-descriptor is two fullwords of data, in which the first word contains the address of the data, and the second word contains the length in bytes of the data.

PASSWORD
specifies the location of the password for the user identified in the DISTINGUISHED_NAME input. The block-descriptor is two fullwords of data, in which the first word contains the address of the data, and the second word contains the length in bytes of the data.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NOTAUTH
NOTFOUND
LDAP_INACTIVE
INVALID_LDAP_URL
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDAP_RESPONSE
Optional parameter

specifies the return code that is sent by the LDAP API, in response to receiving URL and user credentials.

LDAP_SESSION_TOKEN
the name of the fullword token that specifies the LDAP connection.

DDAP gate, END_BROWSE_RESULTS function

The END_BROWSE_RESULTS function of the DDAP gate allows you to end the browse session that was started by the START_BROWSE_RESULTS call.

Input Parameters

SEARCH_TOKEN
the name of the fullword token that is returned by the SEARCH_LDAP function.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
INVALID_TOKEN
NOTFOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDAP_RESPONSE
Optional parameter

specifies the return code that is sent by the LDAP API.

DDAP gate, FLUSH_LDAP_CACHE function

The FLUSH_LDAP_CACHE function of the DDAP gate removes the contents of all cached search responses for the specified LDAP connection.

Input Parameters

LDAP_SESSION_TOKEN
the name of the fullword token that was returned by the BIND_LDAP function.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
INVALID_TOKEN
LDAP_INACTIVE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDAP_RESPONSE
Optional parameter

specifies the return code that is sent by the LDAP API.

DDAP gate, FREE_SEARCH_RESULTS function

The FREE_SEARCH_RESULTS function of the DDAP gate releases all storage held by the SEARCH_LDAP function.

Input Parameters

SEARCH_TOKEN
the name of the fullword token that is returned by the SEARCH_LDAP function.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDAP_RESPONSE

Optional parameter

specifies the return code that is sent by the LDAP API.

DDAP gate, GET_ATTRIBUTE_VALUE function

The GET_ATTRIBUTE_VALUE function of the DDAP gate allows you to retrieve the value associated with an attribute returned by the SEARCH_LDAP call.

Input Parameters

ATTRIBUTE_TYPE

Optional parameter

Specifies the keyword CHARACTER or BINARY, indicating the format of the attribute. If this parameter is not specified, a value of CHARACTER is assumed.

LDAP_ATTRIBUTE_NAME

Specifies the location of the LDAP attribute name. The block-descriptor is two fullwords of data, in which the first word contains the address of the attribute name, and the second word contains the length in bytes of the attribute name. For more information on block-descriptors, see XPI syntax.

LDAP_ATTRIBUTE_VALUE

Indicates the buffer where you want the attribute value returned. A group of three fullwords are specified for the buffer-descriptor:

- The address where the result is returned.
- The maximum size in bytes, of the data returned.
- The actual length in bytes of the result. This can be specified as *, and the length is then returned in DDAP_LDAP_ATTRIBUTE_VALUE_N.

For more information on buffer-descriptors, see XPI syntax.

SEARCH_TOKEN

the name of the fullword token that is returned by the SEARCH_LDAP function.

VALUE_ARRAY_POSITION

Optional parameter

Specifies the position of the requested value, in the value array for the current attribute. This parameter is only required if multiple values are expected. Array indexing starts at position 1.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_TOKEN
NOTFOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDAP_RESPONSE

Optional parameter

specifies the return code that is sent by the LDAP API.

DDAP gate, GET_NEXT_ATTRIBUTE function

The GET_NEXT_ATTRIBUTE function of the DDAP gate allows you to get the next attribute in a series, from an entry returned by the SEARCH_LDAP call.

Input Parameters

LDAP_ATTRIBUTE_NAME

indicates the buffer where you want the attribute name returned. A group of three fullwords are specified for the buffer-descriptor:

- The address where the data is returned.
- The maximum size in bytes, of the data returned.
- The actual length in bytes of the data. This can be specified as *, and the length is then returned in DDAP_LDAP_ATTRIBUTE_NAME_N.

For more information on buffer-descriptors, see XPI syntax.

SEARCH_TOKEN

the name of the fullword token that is returned by the SEARCH_LDAP function.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
INVALID_TOKEN
NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDAP_RESPONSE

Optional parameter

specifies the return code that is sent by the LDAP API.

VALUE_COUNT

Optional parameter

a fullword containing the number of values returned for this attribute. There is usually one value returned.

DDAP gate, GET_NEXT_ENTRY function

The GET_NEXT_ENTRY function of the DDAP gate allows you to get the next entry, from a series of entries returned by the SEARCH_LDAP call.

Input Parameters

DISTINGUISHED_NAME

Optional parameter

indicates the buffer where you want the distinguished name of the next entry in the search returned. A group of three fullwords are specified for the buffer-descriptor:

- The address where the data is returned.
- The maximum size in bytes, of the data is returned.
- The actual length in bytes of the data. This can be specified as *, and the length is then returned in DDAP_DISTINGUISHED_NAME_N.

For more information on buffer-descriptors, see XPI syntax.

SEARCH_TOKEN

the name of the fullword token that is returned by the SEARCH_LDAP function.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_TOKEN
BROWSE_END
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ATTRIBUTE_COUNT

Optional parameter

specifies the number of attributes in the retrieved entry.

LDAP_RESPONSE

Optional parameter

specifies the return code that is sent by the LDAP API.

DDAP gate, SEARCH_LDAP function

The SEARCH_LDAP function of the DDAP gate sends a search request to a specified LDAP server.

Input Parameters

DISTINGUISHED_NAME

specifies the location of the LDAP distinguished name. The block-descriptor is two fullwords of data, in which the first word contains the address of the data, and the second word contains the length in bytes of the data. For more information on block-descriptors, see XPI syntax.

FILTER

Optional parameter

specifies the location of an LDAP filter string that limits the search. If this parameter is not specified or is zero, the search filter is set to `(objectClass=*)`. The block-descriptor is two fullwords of data, in which the first word contains the address of the data, and the second word contains the length in bytes of the data. For more information on block-descriptors, see XPI syntax.

LDAP_SESSION_TOKEN

the name of the fullword token that was returned by the BIND_LDAP function.

SEARCH_TIME_LIMIT

Optional parameter

specifies the time limit for the search (in seconds). If the search is not successful within this time limit, the search is abandoned. If this parameter is not specified or is zero, the search time is unlimited.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_TOKEN
NOTFOUND
TIMED_OUT
LDAP_INACTIVE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ENTRY_COUNT

Optional parameter

the number of LDAP entries returned by the search.

LDAP_RESPONSE
Optional parameter

specifies the return code that is sent by the LDAP API.

SEARCH_TOKEN
the name of the fullword token that identifies and holds the current position in the search.

DDAP gate, START_BROWSE_RESULTS function

The START_BROWSE_RESULTS function of the DDAP gate allows you to browse the results (attributes or entries) returned by the SEARCH_LDAP call.

Input Parameters

DISTINGUISHED_NAME
Optional parameter

indicates the buffer where you want the distinguished name of the first, or only located result returned. A group of three fullwords are specified for the buffer-descriptor:

- The address where the data is returned.
- The length of the buffer in bytes, where the data is returned.
- The maximum length in bytes of the data. This can be specified as *, and the length is then returned in DDAP_DISTINGUISHED_NAME_N.

For more information on buffer-descriptors, see XPI syntax.

SEARCH_TOKEN
the name of the fullword token that is returned by the SEARCH_LDAP function.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ENTRY_COUNT
Optional parameter

a fullword indicating the number of attributes that can be browsed in the current entry.

LDAP_RESPONSE
Optional parameter

specifies the return code that is sent by the LDAP API.

DDAP gate, UNBIND_LDAP function

The UNBIND_LDAP function of the DDAP gate terminates a session with an LDAP server.

Input Parameters

LDAP_SESSION_TOKEN
the name of the fullword token that was returned by the BIND_LDAP function.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_TOKEN
```

LDAP_INACTIVE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDAP_RESPONSE
Optional parameter

specifies the return code that is sent by the LDAP API.

DDBR gate, END_BROWSE function

The END_BROWSE function of the DDBR gate is used to end a browse on a directory.

Input Parameters

BROWSE_TOKEN
is the token for the browse.

DIRECTORY_TOKEN
is the token for the directory.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE
INVALID_DIRECTORY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DDBR gate, GET_NEXT_ENTRY function

The GET_NEXT_ENTRY function of the DDBR gate is used to get the next entry name in alphabetical order in a directory.

Input Parameters

BROWSE_TOKEN
is the token for the browse.

DIRECTORY_TOKEN
is the token for the directory.

ENTRY_NAME
is the address of the entry name. The length is fixed for the directory.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE
INVALID_DIRECTORY
INVALID_NAME

DATA_TOKEN
is the data associated with the entry name when it was deleted.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DDBR gate, START_BROWSE function

The START_BROWSE function of the DDBR gate is used to start an alphabetical browse through all of the entries in a directory.

Input Parameters

DIRECTORY_TOKEN
is the token for the directory.

AT_NAME
Optional Parameter

is the address of an entry name at which the browse is to start. The first name found will be the first which is greater than or equal to this in alphabetical order.

TASK_RELATED
Optional Parameter

is an optional parameter which indicates whether the browse will end at task end.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_DIRECTORY

BROWSE_TOKEN
is the token for this browse.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DDDI gate, ADD_ENTRY function

The ADD_ENTRY function of the DDDI gate is used to add an entry to a directory.

Input Parameters

DATA_TOKEN
is the data to be associated with the entry name in the directory.

DIRECTORY_TOKEN
is the token for the directory.

ENTRY_NAME
is the address of the entry name. The length is fixed for the directory.

SUSPEND
indicates whether Storage Manager GETMAIN requests should be conditional or unconditional.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DUPLICATE
INSUFFICIENT_STORAGE

The following values are returned when RESPONSE is INVALID:

INVALID_DIRECTORY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUPLICATE_DATA_TOKEN
Optional Parameter

is the data currently associated with the entry name if it already exists in the directory.

DDDI gate, CREATE_DIRECTORY function

The CREATE_DIRECTORY function of the DDDI gate is used to create a new directory with entry names of a given length.

Input Parameters

DIRECTORY_NAME
is the four_character name of the directory to be created.

NAME_LENGTH
is the length of entry names in the directory. This value must be a multiple of four, and less than 256.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:

DUPLICATE_DIRECTORY
INVALID_NAME_LEN

DIRECTORY_TOKEN
is the token for the directory.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DDDI gate, DELETE_DIRECTORY function

The DELETE_DIRECTORY function of the DDDI gate is used to delete an empty directory and all associated resources.

Input Parameters

DIRECTORY_TOKEN
is the token for the directory.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:

INVALID_DIRECTORY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DDDI gate, DELETE_ENTRY function

The DELETE_ENTRY function of the DDDI gate is used to delete an entry from a directory.

Input Parameters

DIRECTORY_TOKEN
is the token for the directory.

ENTRY_NAME
is the address of the entry name. The length is fixed for the directory.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_DIRECTORY
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DATA_TOKEN

Optional Parameter

is the data associated with the entry name when it was deleted.

DDDI gate, REPLACE_DATA function

The REPLACE_DATA function of the DDDI gate is used to replace the data associated with an existing entry name in a directory.

Input Parameters

DIRECTORY_TOKEN

is the token for the directory.

ENTRY_NAME

is the address of the entry name. The length is fixed for the directory.

NEW_DATA_TOKEN

is the new data to be associated with the entry name.

PRIOR_DATA_TOKEN

Optional Parameter

is an optional parameter that indicates the data expected to be associated with the entry name just before it being replaced.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
DATA_CHANGED
NOT_FOUND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_DIRECTORY
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DDLO gate, LOCATE function

The LOCATE function of the DDLO gate is used to locate the data associated with an existing entry name in a directory.

Input Parameters

DIRECTORY_TOKEN

is the token for the directory.

ENTRY_NAME

is the address of the entry name. The length is fixed for the directory.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

The following values are returned when RESPONSE is INVALID:
INVALID_DIRECTORY

DATA_TOKEN
is the data associated with the entry name when it was deleted.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Directory manager domain's generic gates

Table 11 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 11. Directory manager domain's generic gates

Gate	Trace	Functions	Format
DDDM	DD 0101 DD 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

Document handler domain (DH)

The document handler domain manages CICS Documents.

Document handler domain's specific gates

The specific gates provide access for other domains to functions that are provided by the DH domain.

DHDH gate, CREATE_DOCUMENT function

The CREATE_DOCUMENT function of the DHDH gate is used to create a new CICS document.

Input Parameters

BINARY
Optional Parameter

is a buffer containing a block of binary data to be added to the document.

HOST_CODEPAGE
Optional Parameter

is the character encoding for the block of data being added to the document. This parameter is taken into account for the TEXT and TEMPLATE_BUFFER options and ignored for all other options.

PRIVATE_DATA
Optional Parameter

indicates that the block of data is private, and should not be exposed in trace records.

Values for the parameter are:
NO
YES

RETRIEVED_DOCUMENT
Optional Parameter

is a buffer containing a document in a retrieved format which is to be added to the document.

SOURCE_DOCUMENT
Optional Parameter

is the document token of an existing document created by the same CICS task which is to be added to the document.

SYMBOL_DELIMITER
Optional Parameter

is the character used to delimit symbol name-value pairs.

SYMBOL_LIST
Optional Parameter

is a buffer containing a list of symbols to be added to the symbol table of the document.

TEMPLATE_BUFFER
Optional Parameter

is a buffer containing a template to be added to the document.

TEMPLATE_IN_ERROR
Optional Parameter

is a buffer which is used by the Document Handler domain to return the name of a DOCTEMPLATE in which an error has been detected. This parameter is only meaningful when specified with the TEMPLATE_NAME option or the TEMPLATE_BUFFER option where the template in the TEMPLATE_BUFFER option contains an embedded template.

TEMPLATE_NAME
Optional Parameter

is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

TEXT
Optional Parameter

is a buffer containing a block of text to be added to the document.

UNESCAPED_DATA
Optional Parameter

indicates if CICS should unescape symbol values in the data.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
CODEPAGE_NOT_SPECIFIED
EMBED_DEPTH_EXCEEDED
INVALID_HOST_CODEPAGE
INVALID_RETRIEVE_FORMAT
INVALID_SYMBOL_LIST_LENGTH
INVALID_TEMPLATE_LENGTH
INVALID_TEMPLATE_SYNTAX
IO_ERROR
SOURCE_DOC_NOT_FOUND
SYMBOL_NAME_INVALID
SYMBOL_VALUE_INVALID
TEMPLATE_NOT_FOUND
TEMPLATE_NOT_USABLE

DOCUMENT_TOKEN
is the token identifying the newly created document.
ERROR_OFFSET
is the offset into a template where a syntax error has been detected.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
DOCUMENT_SIZE
Optional Parameter

is the size of the data in a document.
RETRIEVE_SIZE
Optional Parameter

is the maximum size in bytes that a retrieved copy of the document can be.

DHDH gate, DELETE_BOOKMARK function

The DELETE_BOOKMARK function of the DHDH gate is used to delete a bookmark in an existing document.

Input Parameters

BOOKMARK_NAME
is the 16 byte name of a bookmark to be added to the document.
DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

Output Parameters

REASON
The values for the parameter are:
BOOKMARK_NOT_FOUND
DOCUMENT_NOT_FOUND
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

DHDH gate, DELETE_DATA function

The DELETE_DATA function of the DHDH gate is used to delete the data between 2 bookmarks in an existing document.

Input Parameters

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.
FROM_BOOKMARK
is the name of a bookmark which identifies the start of the data which is to be replaced.
FROM_POSITION
identifies the beginning or end of the document as the start of the data which is to be replaced in the document.
TO_BOOKMARK
is the name of a bookmark which identifies the end of the data which is to be replaced.
TO_POSITION
identifies the beginning or end of the document as the end of the data which is to be replaced in the document.

Output Parameters

REASON

The values for the parameter are:

```
DOCUMENT_NOT_FOUND
FROM_BOOKMARK_NOT_FOUND
INVALID_BOOKMARK_SEQUENCE
TO_BOOKMARK_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RETRIEVE_SIZE

is the maximum size in bytes that a retrieved copy of the document can be.

DHDH gate, DELETE_DOCUMENT function

The DELETE_DOCUMENT function of the DHDH gate is used to delete a document.

Input Parameters

DOCUMENT_TOKEN

is the token which identifies the document into which the data will be inserted.

Output Parameters

REASON

The values for the parameter are:

```
DOCUMENT_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHDH gate, INQUIRE_DOCUMENT function

The INQUIRE_DOCUMENT function of the DHDH gate is used to obtain information about the document.

Input Parameters

DOCUMENT_TOKEN

is the token which identifies the document into which the data will be inserted.

Output Parameters

REASON

The values for the parameter are:

```
DOCUMENT_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DOCUMENT_SIZE

Optional Parameter

is the size of the data in a document.

RETRIEVE_SIZE

Optional Parameter

is the maximum size in bytes that a retrieved copy of the document can be.

DHDH gate, INSERT_BOOKMARK function

The INSERT_BOOKMARK function of the DHDH gate is used to insert a bookmark into an existing document.

Input Parameters

BOOKMARK_NAME
is the 16 byte name of a bookmark to be added to the document.

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

INSERT_AT
is the name of a bookmark which identifies the position at which the data should be inserted.

INSERT_POINT
identifies the beginning or end as the position at which data should be inserted into a document.

Output Parameters

REASON
The values for the parameter are:

```
DOCUMENT_NOT_FOUND
DUPLICATE_BOOKMARK
INSERTPOINT_NOT_FOUND
INVALID_BOOKMARK_NAME
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

DHDH gate, INSERT_DATA function

The INSERT_DATA function of the DHDH gate is used to insert a block of data into an existing document.

Input Parameters

BINARY
is a buffer containing a block of binary data to be added to the document.

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

INSERT_AT
is the name of a bookmark which identifies the position at which the data should be inserted.

INSERT_POINT
identifies the beginning or end as the position at which data should be inserted into a document.

RETRIEVED_DOCUMENT
is a buffer containing a document in a retrieved format which is to be added to the document.

SOURCE_DOCUMENT
is the document token of an existing document created by the same CICS task which is to be added to the document.

SYMBOL
is the name of a symbol defined in the symbol table. The value associated with the symbol will be added to the document.

TEMPLATE_BUFFER
is a buffer containing a template to be added to the document.

TEMPLATE_NAME
is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

TEXT
is a buffer containing a block of text to be added to the document.

HOST_CODEPAGE
Optional Parameter

is the character encoding for the block of data being added to the document. This parameter is taken into account for the TEXT and TEMPLATE_BUFFER options and ignored for all other options.

PRIVATE_DATA
Optional Parameter

indicates that the block of data is private, and should not be exposed in trace records.

Values for the parameter are:
NO
YES

TEMPLATE_IN_ERROR
Optional Parameter

is a buffer which is used by the Document Handler domain to return the name of a DOCTEMPLATE in which an error has been detected. This parameter is only meaningful when specified with the TEMPLATE_NAME option or the TEMPLATE_BUFFER option where the template in the TEMPLATE_BUFFER option contains an embedded template.

Output Parameters

REASON
The values for the parameter are:
CODEPAGE_NOT_SPECIFIED
DOCUMENT_NOT_FOUND
EMBED_DEPTH_EXCEEDED
INSERTPOINT_NOT_FOUND
INVALID_HOST_CODEPAGE
INVALID_RETRIEVE_FORMAT
INVALID_TEMPLATE_LENGTH
INVALID_TEMPLATE_SYNTAX
IO_ERROR
SOURCE_DOC_NOT_FOUND
SYMBOL_NOT_FOUND
TEMPLATE_NOT_FOUND
TEMPLATE_NOT_USABLE

ERROR_OFFSET
is the offset into a template where a syntax error has been detected.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

DHDH gate, REPLACE_DATA function

The REPLACE_DATA function of the DHDH gate is used to replace the data between 2 bookmarks in an existing document.

Input Parameters

BINARY
is a buffer containing a block of binary data to be added to the document.

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

FROM_BOOKMARK
is the name of a bookmark which identifies the start of the data which is to be replaced.

FROM_POSITION
identifies the beginning or end of the document as the start of the data which is to be replaced in the document.

RETRIEVED_DOCUMENT
is a buffer containing a document in a retrieved format which is to be added to the document.

SOURCE_DOCUMENT
is the document token of an existing document created by the same CICS task which is to be added to the document.

SYMBOL
is the name of a symbol defined in the symbol table. The value associated with the symbol will be added to the document.

TEMPLATE_BUFFER
is a buffer containing a template to be added to the document.

TEMPLATE_NAME
is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

TEXT
is a buffer containing a block of text to be added to the document.

TO_BOOKMARK
is the name of a bookmark which identifies the end of the data which is to be replaced.

TO_POSITION
identifies the beginning or end of the document as the end of the data which is to be replaced in the document.

HOST_CODEPAGE
Optional Parameter

is the character encoding for the block of data being added to the document. This parameter is taken into account for the TEXT and TEMPLATE_BUFFER options and ignored for all other options.

PRIVATE_DATA
Optional Parameter

Indicates that the block of data is private, and should not be exposed in trace records.

Values for the parameter are:
NO
YES

TEMPLATE_IN_ERROR
Optional Parameter

is a buffer which is used by the Document Handler domain to return the name of a DOCTEMPLATE in which an error has been detected. This parameter is only meaningful when specified with the TEMPLATE_NAME option or the TEMPLATE_BUFFER option where the template in the TEMPLATE_BUFFER option contains an embedded template.

Output Parameters

REASON
The values for the parameter are:
CODEPAGE_NOT_SPECIFIED
DOCUMENT_NOT_FOUND
EMBED_DEPTH_EXCEEDED

FROM_BOOKMARK_NOT_FOUND
INVALID_HOST_CODEPAGE
INVALID_RETRIEVE_FORMAT
INVALID_TEMPLATE_LENGTH
INVALID_TEMPLATE_SYNTAX
IO_ERROR
SOURCE_DOC_NOT_FOUND
SYMBOL_NOT_FOUND
SYMBOL_NOT_FOUND
TEMPLATE_NOT_FOUND
TEMPLATE_NOT_USABLE
TO_BOOKMARK_NOT_FOUND

ERROR_OFFSET
is the offset into a template where a syntax error has been detected.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

DHDH gate, RETRIEVE_WITH_CTLINFO function

The RETRIEVE_WITH_CTLINFO function of the DHDH gate is used to retrieve a copy of an existing document. The retrieved copy will contain embedded control information.

Input Parameters

DOCUMENT_BUFFER
is a buffer into which the Document Handler domain will place the copy of the document.

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

Output Parameters

REASON
The values for the parameter are:
DOCUMENT_NOT_FOUND
OUTPUT_BUFFER_OVERFLOW

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHDH gate, RETRIEVE_WITHOUT_CTLINFO function

The RETRIEVE_WITHOUT_CTLINFO function of the DHDH gate is used to retrieve a copy of an existing document. The retrieved copy will only contain the data in the document.

Input Parameters

DOCUMENT_BUFFER
is a buffer into which the Document Handler domain will place the copy of the document.

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

CLIENT_CODEPAGE
Optional Parameter

is the character encoding that the retrieved document should be converted to when it is placed in the buffer.

Output Parameters

REASON

The values for the parameter are:

```
CCSID_CONVERSION_ERROR
DOCUMENT_NOT_FOUND
INVALID_CCSID_COMBINATION
INVALID_CLIENT_CODEPAGE
INVALID_HOST_CODEPAGE
OUTPUT_BUFFER_OVERFLOW
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHDH gate, SET_PARAMETERS function

Set document handler domain parameters.

Input Parameters

DEFAULT_CODEPAGE

The default code page used by the document handler domain.

Output Parameters

REASON

The values for the parameter are:

```
INVALID_HOST_CODEPAGE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHFS gate, DELETE_HFS_FILE function

The DELETE_HFS_FILE function is used to remove a link to a z/OS UNIX file. The link may be the pathname to the file. If this is the only remaining link to the file, the file is deleted.

Input Parameters

PATHNAME

The path of the z/OS UNIX file.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
LOOP
NOT_FOUND
NOTAUTH
UNLINK_FAILED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USS_RESPONSE

Optional Parameter

The response from UNIX System Services.

DHFS gate, END_BROWSE_DIRECTORY function

The END_BROWSE_DIRECTORY function terminates the browse of the z/OS UNIX directory.

Input Parameters

BROWSE_TOKEN
A token representing the browse session.

Output Parameters

REASON
The values for the parameter are:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USS_RESPONSE
Optional Parameter

The response from UNIX System Services.

DHFS gate, GET_NEXT_IN_DIRECTORY function

The GET_NEXT_IN_DIRECTORY function returns the next file entry in the current directory buffer. If there are no file entries left, a new directory block is read in. If the number of entries read in is then zero, this indicates the end of the directory, and EXCEPTION/BROWSE_END is returned.

Input Parameters

BROWSE_TOKEN
A token representing the browse session.

FILENAME
A buffer in which the file name is returned.

Output Parameters

REASON
The values for the parameter are:
ABEND
BROWSE_END
INVALID_BROWSE_TOKEN
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USS_RESPONSE
Optional Parameter

The response from UNIX System Services.

DHFS gate, INQUIRE_HFS_FILE function

The INQUIRE_HFS_FILE routine finds the attributes of a z/OS UNIX file without opening it.

Input Parameters

PATHNAME
The path of the z/OS UNIX file.

Output Parameters

REASON
The values for the parameter are:
ABEND
FILE_TOO_LARGE

LOOP
NOT_FOUND
NOTAUTH
STAT_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LAST_MODIFIED_ABSTIME
Optional Parameter

The date and time the file was last modified, expressed in CICS ABSTIME format.

SIZE
Optional Parameter

The size of the file in bytes.

TYPE
Optional Parameter

Indicates if the PATHNAME specifies a file or a directory.

Values for the parameter are:
DIRECTORY
FILE

USS_RESPONSE
Optional Parameter

The response from UNIX System Services.

DHFS gate, MAKE_HFS_DIRECTORY function

Create a directory in z/OS UNIX.

Input Parameters

PATHNAME
The path of the z/OS UNIX directory to be created.

Output Parameters

REASON
The values for the parameter are:
ABEND
ALREADY_EXISTS
LOOP
NOTAUTH

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USS_RESPONSE
Optional Parameter

The response from UNIX System Services.

DHFS gate, READ_HFS_FILE function

The READ_HFS_FILE function is used to read an entire z/OS UNIX file into a user-specified buffer.

Input Parameters

CONTENT
A buffer into which the file is to be read.

PATHNAME
The path to the file.
CONVERT_NEWLINE
Optional Parameter

Specifies the character to which all EBCDIC newline characters ('15'x) are converted. It is typically used before converting the file to ASCII, where a newline character is not valid.

Output Parameters
REASON
The values for the parameter are:
```
ABEND
FILE_TOO_LARGE
LOOP
NOT_FOUND
NOTAUTH
OPEN_FAILED
READ_ERROR
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
LAST_MODIFIED_ABSTIME
Optional Parameter

The date and time the file was last modified, expressed in CICS ABSTIME format.
SIZE
Optional Parameter

The size of the file in bytes.
TYPE
Optional Parameter

Indicates if the PATHNAME specifies a file or a directory.

Values for the parameter are:
```
DIRECTORY
FILE
```
USS_RESPONSE
Optional Parameter

The response from UNIX System Services.

DHFS gate, START_BROWSE_DIRECTORY function
The START_BROWSE_DIRECTORY function starts a browse of the filenames recorded in the z/OS UNIX directory

Input Parameters
PATHNAME
The path of the z/OS UNIX directory to be browsed.

Output Parameters
REASON
The values for the parameter are:
```
ABEND
LOOP
NOT_DIRECTORY
NOT_FOUND
```

NOTAUTH
OPEN_FAILED
READ_ERROR

BROWSE_TOKEN
A token representing the browse session.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USS_RESPONSE
Optional Parameter

The response from UNIX System Services.

DHFS gate, WRITE_HFS_FILE function

The WRITE_HFS_FILE function is used to write an entire z/OS UNIX file from a single user-specified buffer.

Input Parameters

CONTENT
A buffer from which the file is to written.

PATHNAME
The path to the file.

APPEND
Optional Parameter

Specifies whether data is to be appended to the existing file. The default is NO: any existing data is overwritten.

Values for the parameter are:
NO
YES

CREATE_DIRECTORY
Optional Parameter

Specifies whether the directory into which the file is being written should be created if it does not exist. The default is NO: if the directory is missing, a NOT_FOUND exception is returned.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
ABEND
LOOP
NOT_FOUND
NOTAUTH
OPEN_FAILED
READ_ONLY
WRITE_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USS_RESPONSE
Optional Parameter

The response from UNIX System Services.

DHSL gate, ADD_SYMBOL_LIST function

The ADD_SYMBOL_LIST function of the DHSL gate is used to add a list of symbols to the symbol table at one time.

Input Parameters

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

SYMBOL_LIST
is a buffer containing a list of symbols to be added to the symbol table of the document.

PRIVATE_DATA
Optional Parameter

indicates that the symbols contain private data that should not be exposed in trace records.

Values for the parameter are:
NO
YES

SYMBOL_DELIMITER
Optional Parameter

is the character used to delimit symbol name-value pairs.

UNESCAPED_DATA
Optional Parameter

indicates if CICS should unescape symbol values in the data.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
DOCUMENT_NOT_FOUND
FREEMAIN_ERROR
GETMAIN_ERROR
INVALID_LENGTH
SYMBOL_NAME_INVALID
SYMBOL_VALUE_INVALID

ERROR_OFFSET
is the offset into a template where a syntax error has been detected.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHSL gate, EXPORT_SYMBOL_LIST function

The EXPORT_SYMBOL_LIST function of the DHSL gate is used to export all the symbols in the symbol table in a form that can be re-imported with IMPORT_SYMBOL_LIST.

Input Parameters

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

SYMBOL_LIST_BUFFER
is a buffer that is to contain the exported symbol list.

Output Parameters

REASON

The values for the parameter are:

DOCUMENT_NOT_FOUND
INVALID_LENGTH
OUTPUT_BUFFER_OVERFLOW

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHSL gate, IMPORT_SYMBOL_LIST function

The IMPORT_SYMBOL_LIST function of the DHSL gate is used to import all the symbols in the symbol table that were exported with EXPORT_SYMBOL_LIST.

Input Parameters

DOCUMENT_TOKEN

is the token which identifies the document into which the data will be inserted.

SYMBOL_LIST

is a buffer containing a list of symbols to be added to the symbol table of the document.

Output Parameters

REASON

The values for the parameter are:

DOCUMENT_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHSL gate, SET_SYMBOL_VALUE_BY_API function

The SET_SYMBOL_VALUE_BY_API function of the DHSL gate is used to set the value of a symbol in the symbol table. If the symbol does not exist in the table, it will be added. If the symbol does exist in the table, it will always be replaced.

Input Parameters

DOCUMENT_TOKEN

is the token which identifies the document into which the data will be inserted.

SYMBOL_NAME

is the name of the symbol in the symbol table.

VALUE

is the value to be associated with the symbol.

PRIVATE_DATA

Optional Parameter

indicates that the symbol value is private, and should not be exposed in trace records.

Values for the parameter are:

NO
YES

UNESCAPED_DATA

Optional Parameter

indicates if CICS should unescape symbol values in the data.

Values for the parameter are:

NO
YES

Output Parameters

REASON

The values for the parameter are:

DOCUMENT_NOT_FOUND
FREEMAIN_ERROR
GETMAIN_ERROR
INVALID_LENGTH
SYMBOL_NAME_INVALID

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHSL gate, SET_SYMBOL_VALUE_BY_SSI function

The SET_SYMBOL_VALUE_BY_SSI function of the DHSL gate is used to set the value of a symbol in the symbol table. If the symbol does not exist in the table, it will be added. If the symbol does exist in the table, it will only be replaced if it was previously set using the SET_SYMBOL_VALUE_BY_SSI function.

Input Parameters

DOCUMENT_TOKEN

is the token which identifies the document into which the data will be inserted.

SYMBOL_NAME

is the name of the symbol in the symbol table.

VALUE

is the value to be associated with the symbol.

Output Parameters

REASON

The values for the parameter are:

DOCUMENT_NOT_FOUND
FREEMAIN_ERROR
GETMAIN_ERROR
INVALID_LENGTH
SYMBOL_NAME_INVALID

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHTM gate, ADD_REPLACE_DOCTEMPLATE function

The ADD_REPLACE_DOCTEMPLATE function of the DHTM gate is used to install a document template into the currently executing CICS system.

Input Parameters

APPENDCRLF

specifies whether CICS is to delete trailing blanks from and append carriage-return line-feed to each logical record of the template .

Values for the parameter are:

NO
YES

CATALOG_DOC

Specifies if the changes to the doucment template are to be added to the catalog.

Values for the parameter are:

NO
YES

DOCTEMPLATE
is the name of the DOCTEMPLATE resource that is to be added.

HFSPATH
When the template resides in a z/OS UNIX System Services file, the fully qualified (absolute) or relative name of the file.

RESOURCE_NAME
is the name of the resource containing the DOCTEMPLATE.

RESOURCE_TYPE
specifies the type of resource containing the DOCTEMPLATE.

Values for the parameter are:
EXITPGM
FILE
HFSFILE
PDS_MEMBER
PROGRAM
TDQUEUE
TSQUEUE

TEMPLATE_NAME
is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

TYPE
specifies the format of the contents of the template.

Values for the parameter are:
BINARY
EBCDIC

DDNAME
Optional Parameter

is the DDNAME of the PDS containing the DOCTEMPLATE resource if the resource resides on a PDS.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DIRECTORY_ERROR
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
DDNAME_NOT_FOUND
FREEMAIN_FAILED
GETMAIN_FAILED
IO_ERROR
MEMBER_NOT_FOUND
NAME_IN_USE
NOT_FOUND
NOT_USABLE
TRUNCATED

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
INVALID_RESOURCE_TYPE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DATASET
Optional Parameter

is the data set name of the PDS containing the DOCTEMPLATE resource if the resource resides on a PDS.

DOCTEMPLATE_IN_USE
Optional Parameter

is the name of the DOCTEMPLATE definition that uses the same TEMPLATE_NAME as the resource being defined.

DHTM gate, DELETE_DOCTEMPLATE function

The DELETE_DOCTEMPLATE function of the DHTM gate deletes a previously installed DOCTEMPLATE.

Input Parameters

DOCTEMPLATE
is the name of the DOCTEMPLATE resource that is to be added.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
DIRECTORY_ERROR
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
DDNAME_NOT_FOUND
FREEMAIN_FAILED
GETMAIN_FAILED
IO_ERROR
MEMBER_NOT_FOUND
NAME_IN_USE
NOT_FOUND
NOT_USABLE
TRUNCATED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
INVALID_RESOURCE_TYPE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHTM gate, END_BROWSE function

The END_BROWSE function of the DHTM gate is used to terminate a browse of installed DOCTEMPLATE definitions.

Input Parameters

BROWSE_TOKEN
is the token identifying this browse of the DOCTEMPLATE definitions.

Output Parameters

REASON
The values for the parameter are:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHTM gate, GET_NEXT function

The GET_NEXT function of the DHTM gate returns information about the next installed DOCTEMPLATE in the browse.

Input Parameters

BROWSE_TOKEN
is the token identifying this browse of the DOCTEMPLATE definitions.

Output Parameters

REASON
The values for the parameter are:
BROWSE_END
INVALID_BROWSE_TOKEN

APPENDCRLF
specifies whether CICS is to delete trailing blanks from and append carriage-return line-feed to each logical record of the template .

Values for the parameter are:
NO
YES

DATASET
is the data set name of the PDS containing the DOCTEMPLATE resource if the resource resides on a PDS.

DDNAME
is the DDNAME of the template PDS if the RESOURCE_TYPE indicates a PDS.

DOCTEMPLATE
is the name of the DOCTEMPLATE resource as it is known to RDO.

HFSPATH
When the template resides in a z/OS UNIX System Services file, the fully qualified (absolute) or relative name of that file.

RESOURCE_NAME
is the name of the CICS or non-CICS resource.

RESOURCE_TYPE
is the CICS or non-CICS resource type associated with the template.

Values for the parameter are:
EXITPGM
FILE
HFSFILE
PDS_MEMBER
PROGRAM
TDQUEUE
TSQUEUE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TEMPLATE_NAME
is the full name of the template as known outside RDO.

TYPE
specifies the format of the contents of the template.

Values for the parameter are:
BINARY
EBCDIC

DHTM gate, INITIALIZE_DOCTEMPLATES function

The INITIALIZE_DOCTEMPLATES function of the DHSL gate is used to initialize the state required by the template manager.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DIRECTORY_ERROR
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
DDNAME_NOT_FOUND
FREEMAIN_FAILED
GETMAIN_FAILED
IO_ERROR
MEMBER_NOT_FOUND
NAME_IN_USE
NOT_FOUND
NOT_USABLE
TRUNCATED

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
INVALID_RESOURCE_TYPE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHTM gate, INQUIRE_DOCTEMPLATE function

The INQUIRE_DOCTEMPLATE function of the DHTM gate returns information about a previously installed document template.

Input Parameters

DOCTEMPLATE
is the name of the DOCTEMPLATE resource that is to be added.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DIRECTORY_ERROR
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
DDNAME_NOT_FOUND
FREEMAIN_FAILED

GETMAIN_FAILED
IO_ERROR
MEMBER_NOT_FOUND
NAME_IN_USE
NOT_FOUND
NOT_USABLE
TRUNCATED

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
INVALID_RESOURCE_TYPE

APPENDCRLF
specifies whether CICS is to delete trailing blanks from and append carriage-return line-feed to each logical record of the template .

Values for the parameter are:
NO
YES

DATASET
is the data set name of the PDS containing the DOCTEMPLATE resource if the resource resides on a PDS.

DDNAME
is the DDNAME of the template PDS if the RESOURCE_TYPE indicates a PDS.

HFSPATH
When the template resides in a z/OS UNIX System Services file, the fully qualified (absolute) or relative name of the z/OS UNIX file.

RESOURCE_NAME
is the name of the CICS or non-CICS resource.

RESOURCE_TYPE
is the CICS or non-CICS resource type associated with the template.

Values for the parameter are:
EXITPGM
FILE
HFSFILE
PDS_MEMBER
PROGRAM
TDQUEUE
TSQUEUE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TEMPLATE_NAME
is the full name of the template as known outside RDO.

TYPE
specifies the format of the contents of the template.

Values for the parameter are:
BINARY
EBCDIC

DHTM gate, INQUIRE_TEMPLATE_STATUS function

The INQUIRE_TEMPLATE_STATUS function of the DHTM gate is used to inquire the installation status of one or more templates.

Input Parameters

TEMPLATE_NAME_LIST
A list of template names whose install status is sought.

TEMPLATE_STATUS_LIST
is a list of install status indicators for the templates named in the TEMPLATE_NAME_LIST

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
DIRECTORY_ERROR
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
DDNAME_NOT_FOUND
FREEMAIN_FAILED
GETMAIN_FAILED
IO_ERROR
MEMBER_NOT_FOUND
NAME_IN_USE
NOT_FOUND
NOT_USABLE
TRUNCATED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
INVALID_RESOURCE_TYPE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DHTM gate, READ_TEMPLATE function

The READ_TEMPLATE function of the DHTM gate is used to read a named template into a buffer provided by the caller.

Input Parameters

TEMPLATE_BUFFER
is a buffer containing a template to be added to the document.

TEMPLATE_NAME
is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
DIRECTORY_ERROR
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
DDNAME_NOT_FOUND
FREEMAIN_FAILED
GETMAIN_FAILED
```

IO_ERROR
MEMBER_NOT_FOUND
NAME_IN_USE
NOT_FOUND
NOT_USABLE
TRUNCATED

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
INVALID_RESOURCE_TYPE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DOCTEMPLATE
Optional Parameter

is the name of the DOCTEMPLATE resource as it is known to RDO.

TYPE
Optional Parameter

specifies the format of the contents of the template.

Values for the parameter are:
BINARY
EBCDIC

DHTM gate, START_BROWSE function

The START_BROWSE function of the DHTM gate is used to initiate a browse of installed DOCTEMPLATE definitions.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DIRECTORY_ERROR
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
DDNAME_NOT_FOUND
FREEMAIN_FAILED
GETMAIN_FAILED
IO_ERROR
MEMBER_NOT_FOUND
NAME_IN_USE
NOT_FOUND
NOT_USABLE
TRUNCATED

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
INVALID_RESOURCE_TYPE

BROWSE_TOKEN
is a token identifying this DOCTEMPLATE browse.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Document handler domain's generic gates

Table 12 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 12. Document handler domain's generic gates

Gate	Trace	Functions	Format
APUE	DH 0D01 DH 0D02 DH 0D03 DH 0D04 DH 0D05 DH 0D06 DH 0D07 DH 0D08	SET_EXIT_STATUS	APUE
DDDM	DD 0101 DD 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

Document handler domain's call-back gates

Table 13 summarizes the domain's call-back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the call-back formats for calls to the gates.

Table 13. Document handler domain's call-back gates

Gate	Trace	Functions	Format
RMDE	DH 0301 DH 0302 DH 0303 DH 0304 DH 0306 DH 0308	START_DELIVERY DELIVER_RECOVERY END_DELIVERY	RMDE
RMKP	DH 0301 DH 0302 DH 0303 DH 0304 DH 0307 DH 0308	TAKE_KEYPOINT	RMKP
RMRO	DH 0301 DH 0302 DH 0303 DH 0304 DH 0305 DH 0308	PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT START_BACKOUT END_BACKOUT	RMRO

Modules

Module	Function
DFHDHDH	Handles the following requests: CREATE_DOCUMENT INSERT_DATA INSERT_BOOKMARK REPLACE_DATA DELETE_DOCUMENT DELETE_DATA DELETE_BOOKMARK RETRIEVE_WITH_CTLINFO RETRIEVE_WITHOUT_CTLINFO INQUIRE_DOCUMENT
DFHDHDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHDHDUF	DH domain offline dump formatting routine
DFHDHPB	Processes data supplied on the BINARY parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPD	Processes data supplied on the SOURCE_DOCUMENT parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPM	Processes data supplied on the TEMPLATE_NAME parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPR	Reads templates held as member's of partitioned data sets.
DFHDHPS	Processes data supplied on the SYMBOL parameter of INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPT	Processes data supplied on the TEXT parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPU	Processes data supplied on the TEMPLATE_BUFFER parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHPX	Processes data supplied on the RETRIEVED_DOCUMENT parameter of CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDH.
DFHDHRM	Handles the following requests: PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT START_BACKOUT END_BACKOUT START_DELIVERY DELIVER_RECOVERY END_DELIVERY TAKE_KEYPOINT

Module	Function
DFHDHSL	Handles the following requests: SET_SYMBOL_VALUE_BY_API, SET_SYMBOL_VALUE_BY_SSI, ADD_SYMBOL_LIST EXPORT_SYMBOL_LIST IMPORT_SYMBOL_LIST
DFHDHTM	Handles the following requests: INITIALIZE_DOCTEMPLATES ADD_REPLACE_DOCTEMPLATE DELETE_DOCTEMPLATE INQUIRE_DOCTEMPLATE INQUIRE_TEMPLATE_STATUS START_BROWSE GET_NEXT END_BROWSE READ_TEMPLATE
DFHDHTRI	Interprets DH domain trace entries
DFHDHUE	Handles the following requests: SET_EXIT_STATUS

Domain manager domain (DM)

The domain manager domain maintains permanent information about other domains.

Domain manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the DM domain.

DMDM gate, ADD_DOMAIN function

The ADD_DOMAIN function of the DMDM gate adds a new domain to the DM table (on the CICS(R) catalog) of all domains. Because the add is placed on the catalog, it survives system failure. A delete is required to remove the entry.

Input Parameters

DOMAIN_ID
is the unique character pair, usually an abbreviated form of the domain name.

DOMAIN_NAME
is a unique string, 1 through 8 characters, which is the name of the domain.

DOMAIN_TOKEN
is the unique index that corresponds to the new table entry for the domain.

PROGRAM_NAME
is a unique string, 1 through 8 characters, which is the name of the initialization module for the specified domain.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOADER_ERROR
```

LOOP

The following values are returned when RESPONSE is EXCEPTION:
DUPLICATE_DOMAIN_NAME
DUPLICATE_DOMAIN_TOKEN
INSUFFICIENT_STORAGE
PROGRAM_NOT_FOUND

RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR

DMDM gate, QUIESCE_SYSTEM function

The QUIESCE_SYSTEM function of the DMDM gate is used to call the domain manager to cause a normal shutdown of the system.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
INSUFFICIENT_STORAGE
LOOP

The following values are returned when RESPONSE is INVALID:
SYSTEM_INITIALISING

RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR

DMDM gate, SET_PHASE function

When a domain issues SET_PHASE during initialization, it is declaring that it is now prepared to support a given set of services.

Input Parameters

PHASE
specifies the set of services that are to be available.

STATUS
is either ACTIVE or INACTIVE.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_PHASE
SYSTEM_NOT_INITIALISING

SYSTEM_NOT_QUIESCING

RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR

DMDM gate, WAIT_PHASE function

The WAIT_PHASE function of the DMDM gate is used to wait until the services required to carry on the work are available.

Input Parameters

PHASE
specifies the set of services that are to be available.

STATUS
is either ACTIVE or INACTIVE.

DOMAIN_TOKEN
Optional Parameter

is the unique index that corresponds to the new table entry for the domain.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
DOMAIN_TOKEN_NOT_ACTIVE

The following values are returned when RESPONSE is INVALID:
INVALID_PHASE
SYSTEM_NOT_INITIALISING
SYSTEM_NOT_QUIESCING

RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR

DMEN gate, DELETE function

The DELETE function of the DMEN gate is used to deregister an interest in an ENF event.

Input Parameters

EVENT
is the event in which the caller is registering an interest

Values for the parameter are:
ENF_RACF_USERID_CHANGED
ENF_SMSVSAM_OPERATIONAL

ENF_SYSTEM_TIME_CHANGED

LISTEN_GATE
is the gate number of the gate at which the caller wants to be notified when the event occurs.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
LISTEN_NOT_ACTIVE

RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

DMEN gate, LISTEN function

The LISTEN function of the DMEN gate is issued to register an interest in an event notification facility (ENF) event. The MVS(TM) event notification facility is a generalized communication facility which allows subsystems to broadcast notification of events.

Input Parameters

EVENT
is the event in which the caller is registering an interest.

Values for the parameter are:
ENF_RACF_USERID_CHANGED
ENF_SMSVSAM_OPERATIONAL

LISTEN_GATE
is the gate number of the gate at which the caller wants to be notified when the event occurs.

Output Parameters

REASON
The values for the parameter are:
DUPLICATE_LISTEN
UNKNOWN_EVENT

RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

DMIQ gate, END_BROWSE function

The END_BROWSE function of the DMIQ gate is used to release the browse thread at any time.

Input Parameters

BROWSE_TOKEN
is the token identifying this browse session.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
BROWSE_TOKEN_NOT_FOUND
```

RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
```

DMIQ gate, GET_NEXT function

The GET_NEXT function of the DMIQ gate is used to return the next available record or an END indication.

Input Parameters

BROWSE_TOKEN
is the token identifying this browse session.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
END_LIST
```

The following values are returned when RESPONSE is INVALID:

```
BROWSE_TOKEN_NOT_FOUND
```

DOMAIN_ID
is the unique character pair, usually an abbreviated form of the domain name.

DOMAIN_NAME
is a unique string, 1 through 8 characters, which is the name of the domain.

DOMAIN_PHASE
is the current phase level for that domain.

DOMAIN_STATUS
is ACTIVE or INACTIVE.

DOMAIN_TOKEN
is the unique index that corresponds to the new table entry for the domain.

RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
```

KERNERROR

DMIQ gate, INQ_DOMAIN_BY_ID function

The INQ_DOMAIN_BY_ID function of the DMIQ gate is used to get the domain's token, name, status, and phase for the specified domain ID.

Input Parameters

DOMAIN_ID
is the unique character pair, usually an abbreviated form of the domain name.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
DOMAIN_ID_NOT_FOUND

DOMAIN_NAME
is a unique string, 1 through 8 characters, which is the name of the domain.

DOMAIN_PHASE
is the current phase level for that domain.

DOMAIN_STATUS
is ACTIVE or INACTIVE.

DOMAIN_TOKEN
is the unique index that corresponds to the new table entry for the domain.

RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR

DMIQ gate, INQ_DOMAIN_BY_NAME function

The INQ_DOMAIN_BY_NAME function of the DMIQ gate is used to get the domain's token, ID, status, and phase for the specified domain name.

Input Parameters

DOMAIN_NAME
is a unique string, 1 through 8 characters, which is the name of the domain.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
DOMAIN_NAME_NOT_FOUND

DOMAIN_ID
is the unique character pair, usually an abbreviated form of the domain name.

DOMAIN_PHASE
is the current phase level for that domain.

DOMAIN_STATUS
is ACTIVE or INACTIVE.

DOMAIN_TOKEN
is the unique index that corresponds to the new table entry for the domain.
RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:
```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
```

DMIQ gate, INQ_DOMAIN_BY_TOKEN function

The INQ_DOMAIN_BY_TOKEN function of the DMIQ gate is used to get the domain's name, ID, status, and phase for the specified domain token.

Input Parameters

DOMAIN_TOKEN
is the unique index that corresponds to the new table entry for the domain.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:
```
DOMAIN_TOKEN_NOT_FOUND
```
DOMAIN_ID
is the unique character pair, usually an abbreviated form of the domain name.
DOMAIN_NAME
is a unique string, 1 through 8 characters, which is the name of the domain.
DOMAIN_PHASE
is the current phase level for that domain.
DOMAIN_STATUS
is ACTIVE or INACTIVE.
RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:
```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
```

DMIQ gate, START_BROWSE function

The START_BROWSE function of the DMIQ gate is used to create a browse thread. The GET_NEXT function request issued after this command returns the first domain in the active domain list.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
```
BROWSE_TOKEN
is the token identifying this browse session.

RESPONSE
is DFHDMEN's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR

Domain manager domain's generic gates

Table 14 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 14. Domain manager domain's generic gates

Gate	Trace	Functions	Format
DSAT	none	TASK_REPLY	DSAT

Domain manager domain's generic formats

Table 15 describes the generic formats owned by the domain and shows the functions performed on the calls.

Table 15. Domain Manager domain's generic formats

Format	Calling module	Function
DMDM	DFHKETCB	PRE_INITIALIZE
	DFHDMDS	INITIALIZE_DOMAIN
	DFHDMDS	QUIESCE_DOMAIN
	DFHKETCB	TERMINATE_DOMAIN

Note: In the descriptions of the formats, the input parameters are input not to the Domain manager domain, but to the domain being called by the application domain. Similarly, the output parameters are output by the domain that was called by the Domain manager domain, in response to the call.

DMDM gate, INITIALISE_DOMAIN function

A generic function which the domain manager domain uses to call other domains to perform initialization.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
INSUFFICIENT_STORAGE
LOOP

The following values are returned when RESPONSE is INVALID:
ALREADY_INITIALISED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DMDM gate, PRE_INITIALISE function

A generic function which the domain manager domain uses to call other domains to perform the early stages of initialization.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
INSUFFICIENT_STORAGE
LOOP

DUMP_REQUIRED

A binary value that indicates whether a dump is required if pre-initialization failed.

Values for this parameter are

NO
YES

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DMDM gate, QUIESCE_DOMAIN function

A generic function which the domain manager domain uses to call other domains when the system is required to shut down normally.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
INSUFFICIENT_STORAGE
LOOP

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DMDM gate, TERMINATE_DOMAIN function

A generic function which the domain manager domain uses to call other domains when the system is required to shut-down quickly. The call is always made under the job step TCB.

Input Parameters

CANCEL

A binary value that indicates that the request is being issued as a result of an operator cancel. This means that attached subtasks are no longer dispatchable.

Values for the parameter are:

NO
YES

CLEAN_UP

A binary value that indicates that the request is being issued under a clean-up only ESTAE exit. This implies restrictions for terminate logic, specifically that ATTACH cannot be issued.

Values for the parameter are:

NO
YES

TERMINATION_TYPE
Indicates whether the domain is to be terminated immediately or quiesced.

Values for the parameter are:
IMMEDIATE
QUIESCE

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Domain manager domain's callback formats

The Domain manager domain can call other domains by using the callback formats provided by the Domain manager domain.

DMEN gate, PROCESS_ENF_EVENT function

Domains that registered their interest in ENF events are invoked at their identified listen gates when the ENF event occurs. A unique DMEN notify function is provided for each event so that event-specific parameters can be specified in a meaningful way.

Input Parameters

NOTIFY_PLIST
A parameter list specific to the ENF event being notified, which was supplied by the subsystem issuing the ENF signal.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
FAILED_TO_PROCESS_EVENT

RESPONSE
The response of DFHDMEN to the call.

Values for the parameter are as follows:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

Modules

Module	Function
DFHDMDM	Handles the following requests: INITIALIZE_DOMAIN PRE_INITIALIZE QUIESCE_DOMAIN QUIESCE_SYSTEM TERMINATE_DOMAIN SET_PHASE WAIT_PHASE ADD_DOMAIN
DFHDMDS	Handles the TASK_REPLY request
DFHDMDUF	Formats the DM domain control blocks in a CICS system dump
DFHDMEN	Handles LISTEN, DELETE, PROCESS_ENF_EVENT
DFHDMENF	Broadcasts ENF events to interested domains
DFHDMIQ	Handles the following requests: START_BROWSE GET_NEXT END_BROWSE INQUIRE_DOMAIN_BY_ID INQUIRE_DOMAIN_BY_NAME INQUIRE_DOMAIN_BY_TOKEN
DFHDMSVC	Provides authorized services for the DM ENF support
DFHDMTRI	Interprets DM domain trace entries
DFHDMWQ	Handles the following requests: INITIALIZE SET_UP_WAIT RESUME_WAITERS RESUME_DOMAIN_WAITERS RESUME_PHASE_WAITERS

Debugging profile domain (DP)

The Debugging profile domain manages debugging profiles.

Debugging profile domain's specific gates

The specific gates provide access for other domains to functions that are provided by the DP domain.

DPFM gate, ACTIVATE_DEBUG_PROFILE function

Activate a debugging profile.

Input Parameters

CURRENT_USERID
The userid of the user making the request

OWNER_USERID
The userid of the debugging profile's owner

PROFILE_NAME
The name of the debugging profile

SESSION_TYPE
The session type specified in the debugging profile.

Values for the parameter are:

LU3270
TCP

IP_NAME_OR_ADDR_BLOCK
Optional Parameter

A block of storage containing the IP name or IP address

LU_3270_DISPLAY
Optional Parameter

The 3270 display terminal specified in the debugging profile to be used by Debug Tool

PORT
Optional Parameter

The port number specified in the debugging profile

SOCKET_TYPE
Optional Parameter

Specifies whether the debugging client and debugging server will communicate using a single socket or more than one socket.

Values for the parameter are:
MULTIPLE
SINGLE

Output Parameters

REASON
The values for the parameter are:
ABEND
ALREADY_ACTIVE
DISASTER_PERCOLATION
FILE_ERROR
FILE_FULL
INTERNAL_ERROR
PROFILE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PATTERN_MATCH_NUMBER
Optional Parameter

A metric computed from the contents of the debugging profile, which is compared with the pattern match number form other profiles to determine which of the profiles is the best match for a program instance.

DPFM gate, DELETE_DEBUG_PROFILE function

Delete a debugging profile from the debugging profile data set.

Input Parameters

CURRENT_USERID
The userid of the user making the request

OWNER_USERID
The userid of the debugging profile's owner

PROFILE_NAME
The name of the debugging profile

Output Parameters

REASON
The values for the parameter are:

ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
PROFILE_ACTIVE
PROFILE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPFM gate, END_PM_BROWSE function

End the browse for pattern matching.

Output Parameters

REASON
The values for the parameter are:
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPFM gate, GET_DEBUG_PROFILE function

Retrieve a debugging profile from the debugging profile data set.

Input Parameters

OWNER_USERID
The userid of the debugging profile's owner

PROFILE_NAME
The name of the debugging profile

BEAN_BLOCK
Optional Parameter

A block of storage containing the bean name

CLASS_BLOCK
Optional Parameter

A block of storage containing the class name

IP_NAME_OR_ADDR_BLOCK
Optional Parameter

A block of storage containing the IP name or IP address

LE_OPTIONS_BLOCK
Optional Parameter

A block of storage containing Language Environment options

METHOD_BLOCK
Optional Parameter

A block of storage containing the method name

Output Parameters

REASON
The values for the parameter are:
ABEND
DISASTER_PERCOLATION
FILE_ERROR

INTERNAL_ERROR
PROFILE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACTIVATE_USERID
Optional Parameter

For an active debugging profile, the user ID of the user who made it active.

APPLID
Optional Parameter

The APPLID specified in the debugging profile

COMMAND_FILE
Optional Parameter

The command file specified in the debugging profile

COMP_UNIT
Optional Parameter

The compilation unit name specified in the debugging profile

JVM_PROFILE
Optional Parameter

The JVM profile specified in the debugging profile

LU_3270_DISPLAY
Optional Parameter

The 3270 display terminal to be used by Debug Tool

NETNAME
Optional Parameter

The terminal's network name specified in the debugging profile

PATTERN_MATCH_NUMBER
Optional Parameter

A metric computed from the contents of the debugging profile, which is compared with the pattern match number form other profiles to determine which of the profiles is the best match for a program instance.

PORT
Optional Parameter

The port number specified in the debugging profile

PREFERENCE_FILE
Optional Parameter

The preference file specified in the debugging profile

PROGRAM
Optional Parameter

The program name specified in the debugging profile

PROMPT
Optional Parameter

The prompt specified in the debugging profile

SESSION_TYPE
Optional Parameter

The session type specified in the debugging profile.

Values for the parameter are:
LU3270
TCP

SOCKET_TYPE
Optional Parameter

Specifies whether the debugging client and debugging server will communicate using a single socket or more than one socket.

Values for the parameter are:
MULTIPLE
SINGLE

STATUS
Optional Parameter

The status of the debugging profile.

Values for the parameter are:
ACTIVE
INACTIVE

TERMID
Optional Parameter

The terminal ID specified in the debugging profile

TEST_LEVEL
Optional Parameter

The test level specified in the debugging profile.

Values for the parameter are:
ALL
ERROR
NONE

TRANID
Optional Parameter

The transaction ID specified in the debugging profile

TYPE
Optional Parameter

The type of debugging profile.

Values for the parameter are:
C
E
J
LE

USERID
Optional Parameter

The user ID specified in the debugging profile

DPFM gate, INACTIVATE_DEBUG_PROFILE function

Inactivate a debug profile on the debugging profile data set.

Input Parameters

CURRENT_USERID
The userid of the user making the request

OWNER_USERID
The userid of the debugging profile's owner

PROFILE_NAME
The name of the debugging profile

Output Parameters

REASON
The values for the parameter are:
ABEND
ALREADY_INACTIVE
DISASTER_PERCOLATION
FILE_ERROR
FILE_FULL
INTERNAL_ERROR
PROFILE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPFM gate, READNEXT_PM_PROFILE function

Read the next profile on the debugging profile data set for pattern match.

Input Parameters

BEAN_BLOCK
Optional Parameter

A block of storage containing the bean name

CLASS_BLOCK
Optional Parameter

A block of storage containing the class name

IP_NAME_OR_ADDR_BLOCK
Optional Parameter

A block of storage containing the IP name or IP address

LE_OPTIONS_BLOCK
Optional Parameter

A block of storage containing Language Environment options

MANGLED_METHOD_BLOCK
Optional Parameter

A block of storage containing the mangled method name

Output Parameters

REASON
The values for the parameter are:
ABEND
DISASTER_PERCOLATION
END_OF_PROFILES
FILE_ERROR
INTERNAL_ERROR

APPLID
The APPLID specified in the debugging profile

COMMAND_FILE
The command file specified in the debugging profile

COMP_UNIT
The compilation unit name specified in the debugging profile

JVM_PROFILE
The JVM profile specified in the debugging profile

LU_3270_DISPLAY
The 3270 display terminal to be used by Debug Tool

NETNAME
The terminal's network name specified in the debugging profile

OWNER_USERID
The userid of the profile's owner
PATTERN_MATCH_NUMBER
A metric computed from the contents of the debugging profile, which is compared with the pattern match number form other profiles to determine which of the profiles is the best match for a program instance.
PORT
The port number specified in the debugging profile
PREFERENCE_FILE
The preference file specified in the debugging profile
PROFILE_NAME
The name of the debugging profile
PROGRAM
The program name specified in the debugging profile
PROMPT
The prompt specified in the debugging profile
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
SESSION_TYPE
The session type specified in the debugging profile.

Values for the parameter are:
LU3270
TCP
SOCKET_TYPE
Specifies whether the debugging client and debugging server will communicate using a single socket or more than one socket.

Values for the parameter are:
MULTIPLE
SINGLE
TERMID
The terminal ID specified in the debugging profile
TEST_LEVEL
The test level specified in the debugging profile.

Values for the parameter are:
ALL
ERROR
NONE
TRANID
The transaction ID specified in the debugging profile
TYPE
The type of debugging profile.

Values for the parameter are:
C
E
J
LE
USERID
The user ID specified in the debugging profile
ACTIVATE_USERID
Optional Parameter

For an active debugging profile, the user ID of the user who made it active.

DPFM gate, REPLACE_DEBUG_PROFILE function

Replace a debug_profile on the debugging profile data set.

Input Parameters

OWNER_USERID
The userid of the debugging profile's owner

PROFILE_NAME
The name of the debugging profile

APPLID
Optional Parameter

The APPLID specified in the debugging profile

BEAN_BLOCK
Optional Parameter

A block of storage containing the bean name

CLASS_BLOCK
Optional Parameter

A block of storage containing the class name

COMMAND_FILE
Optional Parameter

The command file specified in the debugging profile

COMP_UNIT
Optional Parameter

The compilation unit name specified in the debugging profile

IP_NAME_OR_ADDR_BLOCK
Optional Parameter

A block of storage containing the IP name or IP address

JVM_PROFILE
Optional Parameter

The JVM profile specified in the debugging profile

LE_OPTIONS_BLOCK
Optional Parameter

A block of storage containing Language Environment options

LU_3270_DISPLAY
Optional Parameter

The 3270 display terminal specified in the debugging profile to be used by Debug Tool

METHOD_BLOCK
Optional Parameter

A block of storage containing the method name

NETNAME
Optional Parameter

The terminal's network name specified in the debugging profile

PORT
Optional Parameter

The port number specified in the debugging profile

PREFERENCE_FILE
Optional Parameter

The preference file specified in the debugging profile

PROGRAM
Optional Parameter

The program name specified in the debugging profile

PROMPT
Optional Parameter

The prompt specified in the debugging profile

SESSION_TYPE
Optional Parameter

The session type specified in the debugging profile.

Values for the parameter are:
- LU3270
- TCP

SOCKET_TYPE
Optional Parameter

Specifies whether the debugging client and debugging server will communicate using a single socket or more than one socket.

Values for the parameter are:
- MULTIPLE
- SINGLE

TERMID
Optional Parameter

The terminal ID specified in the debugging profile

TEST_LEVEL
Optional Parameter

The test level specified in the debugging profile.

Values for the parameter are:
- ALL
- ERROR
- NONE

TRANID
Optional Parameter

The transaction ID specified in the debugging profile

TYPE
Optional Parameter

The type of debugging profile.

Values for the parameter are:
- C
- E
- J
- LE

USERID
Optional Parameter

The user ID specified in the debugging profile

Output Parameters

REASON
The values for the parameter are:
- ABEND
- APPLID_INVALID

BEAN_INVAL_FOR_TYPE_C
BEAN_INVAL_FOR_TYPE_J
BEAN_INVALID
CLASS_INVAL_FOR_TYPE_E
CLASS_INVALID
CMD_FILE_INVALID
COMP_UNIT_INVALID
DISASTER_PERCOLATION
FILE_ERROR
FILE_FULL
INTERNAL_ERROR
JVM_PROFILE_INVALID
METHOD_INVAL_FOR_TYPE_J
METHOD_INVALID
NETNAME_INVALID
PREF_FILE_INVALID
PROFILE_NAME_BLANK
PROFILE_NAME_INVALID
PROGRAM_INVALID
PROMPT_INVALID
TERMID_INVALID
TRANID_INVALID
USERID_INVALID

NEW_PROFILE_CREATED
Indicates whether a new profile was created.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MANGLE_CODE
Optional Parameter

Indicates how a method or class name was mangled.

Values for the parameter are:
IDL_KEYWORD
MANGLED_TO_SELF
PROPERTY_ACC
UNDERSCORE

DPFM gate, SAVE_DEBUG_PROFILE function
Save a debug profile on the debug profile data set.

Input Parameters
OWNER_USERID
The userid of the debugging profile's owner

PROFILE_NAME
The name of the debugging profile

APPLID
Optional Parameter

The APPLID specified in the debugging profile

BEAN_BLOCK
Optional Parameter

A block of storage containing the bean name

CLASS_BLOCK
Optional Parameter

A block of storage containing the class name

COMMAND_FILE
Optional Parameter

The command file specified in the debugging profile

COMP_UNIT
Optional Parameter

The compilation unit name specified in the debugging profile

IP_NAME_OR_ADDR_BLOCK
Optional Parameter

A block of storage containing the IP name or IP address

JVM_PROFILE
Optional Parameter

The JVM profile specified in the debugging profile

LE_OPTIONS_BLOCK
Optional Parameter

A block of storage containing Language Environment options

LU_3270_DISPLAY
Optional Parameter

The 3270 display terminal specified in the debugging profile to be used by Debug Tool

METHOD_BLOCK
Optional Parameter

A block of storage containing the method name

NETNAME
Optional Parameter

The terminal's network name specified in the debugging profile

PORT
Optional Parameter

The port number specified in the debugging profile

PREFERENCE_FILE
Optional Parameter

The preference file specified in the debugging profile

PROGRAM
Optional Parameter

The program name specified in the debugging profile

PROMPT
Optional Parameter

The prompt specified in the debugging profile

SESSION_TYPE
Optional Parameter

The session type specified in the debugging profile.

Values for the parameter are:

```
LU3270
TCP
```

SOCKET_TYPE
Optional Parameter

Specifies whether the debugging client and debugging server will communicate using a single socket or more than one socket.

Values for the parameter are:

MULTIPLE
SINGLE

TERMID

Optional Parameter

The terminal ID specified in the debugging profile

TEST_LEVEL

Optional Parameter

The test level specified in the debugging profile.

Values for the parameter are:

ALL
ERROR
NONE

TRANID

Optional Parameter

The transaction ID specified in the debugging profile

TYPE

Optional Parameter

The type of debugging profile.

Values for the parameter are:

C
E
J
LE

USERID

Optional Parameter

The user ID specified in the debugging profile

Output Parameters

REASON

The values for the parameter are:

ABEND
APPLID_INVALID
BEAN_INVAL_FOR_TYPE_C
BEAN_INVAL_FOR_TYPE_J
BEAN_INVALID
CLASS_INVAL_FOR_TYPE_E
CLASS_INVALID
CMD_FILE_INVALID
COMP_UNIT_INVALID
DISASTER_PERCOLATION
DUPLICATE_PROFILE
FILE_ERROR
FILE_FULL
INTERNAL_ERROR
JVM_PROFILE_INVALID
METHOD_INVAL_FOR_TYPE_J
METHOD_INVALID
NETNAME_INVALID
PREF_FILE_INVALID

PROFILE_NAME_BLANK
PROFILE_NAME_INVALID
PROGRAM_INVALID
PROMPT_INVALID
TERMID_INVALID
TRANID_INVALID
USERID_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MANGLE_CODE
Optional Parameter

Indicates how a method or class name was mangled.

Values for the parameter are:
IDL_KEYWORD
MANGLED_TO_SELF
PROPERTY_ACC
UNDERSCORE

DPFM gate, START_PM_BROWSE function
Start a browse for pattern matching.

Input Parameters
MATCH_TYPE
Optional Parameter

The type of debugging profile to match during the browse operation.

Values for the parameter are:
TYPE_J
TYPE_LE

Output Parameters
REASON
The values for the parameter are:
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
NO_PROFILES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPIQ gate, INQUIRE_DEBUG_TASK function
Inquire DP domain debug settings.

Output Parameters
REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
OUT_OF_RANGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DEBUG_TASK
Optional Parameter

Specifies whether Debug Tool is to be used to debug an application.

Values for the parameter are:
NO
YES

DPIQ gate, INQUIRE_PARAMETERS function

Inquire DP domain parameters.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
OUT_OF_RANGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DEBUGTOOL
Optional Parameter

The value of the DEBUGTOOL system initialization parameter.

Values for the parameter are:
DEBUGTOOL_NO
DEBUGTOOL_YES

DTLEVEL
Optional Parameter

Specifies whether the level of Debug Tool supports the CADP transaction.

Values for the parameter are:
DTNEW_NO
DTNEW_YES

DPIQ gate, SET_DEBUG_PROFILE function

Set DP domain parameters.

Input Parameters

DEBUG_PROFILE
Optional Parameter

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
OUT_OF_RANGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPIQ gate, SET_DEBUGGING function

Sets the state of the debugging profile domain.

Input Parameters

DOMAIN_STATE

The intended state of the domain.

Values for the parameter are:

DISABLED
ENABLED

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
OUT_OF_RANGE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPIQ gate, SET_PARAMETERS function

Set DP domain parameters.

Input Parameters

DEBUGTOOL

Optional Parameter

The value of the DEBUGTOOL system initialization parameter.

Values for the parameter are:

DEBUGTOOL_NO
DEBUGTOOL_YES

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
OUT_OF_RANGE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPLM gate, ENDBR_DEBUG_PROFILES function

End the browse for pattern matching.

Input Parameters

BROWSE_LIST_TOKEN

A token which uniquely identifies the list of profiles.

Output Parameters

REASON

The values for the parameter are:

ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CURRENT_PAGE

Optional Parameter

Specifies which page of the list of profiles is currently displayed

DPLM gate, READNEXT_DEBUG_PROFILE function

Returns one profile to the caller for display on the screen. Largely for the benefit of the 3270 version of CADP, the readnext can optionally position itself based on a page size parameter so that it is possible to easily implement scrolling up and down. The default if no position is specified is to return the next profile.

Input Parameters

BROWSE_LIST_TOKEN

A token which uniquely identifies the list of profiles.

BEAN_BLOCK

Optional Parameter

A block of storage containing the bean name

CLASS_BLOCK

Optional Parameter

A block of storage containing the class name

LE_OPTIONS_BLOCK

Optional Parameter

A block of storage containing Language Environment options

MANGLED_METHOD_BLOCK

Optional Parameter

A block of storage containing the mangled method name

METHOD_BLOCK

Optional Parameter

A block of storage containing the method name

PAGE_SIZE

Optional Parameter

The number of profiles which can be shown on a page of the display

POSITION

Optional Parameter

Specifies the position in the list of the next profile to be read.

Values for the parameter are:

```
NEXT_PROFILE
PAGE_BACK
PAGE_FORWARD
TOP
TOP_CURRENT_PAGE
```

Output Parameters

REASON

The values for the parameter are:

```
ABEND
ALREADY_AT_BOTTOM
ALREADY_AT_TOP
DISASTER_PERCOLATION
END_OF_PROFILES
INTERNAL_ERROR
```

APPLID

The Applid specified in the debugging profile

COMMAND_FILE

The command file specified in the debugging profile

COMP_UNIT
The compilation unit name specified in the debugging profile
INPUT
The action specified for the profile.

Values for the parameter are:
ACTIVATE
CLEAR
COPY
DELETE
INACTIVATE
JVM_PROFILE
The JVM profile specified in the debugging profile
NETNAME
The terminal's network name specified in the debugging profile
OWNER_USERID
The userid of the profile's owner
PREFERENCE_FILE
The preference file specified in the debugging profile
PROFILE_NAME
The name of the debugging profile
PROGRAM
The program name specified in the debugging profile
PROMPT
The prompt specified in the debugging profile
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
STATUS
The status of the debugging profile.

Values for the parameter are:
ACTIVE
INACTIVE
TERMID
The terminal ID specified in the debugging profile
TEST_LEVEL
The test level specified in the debugging profile.

Values for the parameter are:
ALL
ERROR
NONE
TRANID
The transaction ID specified in the debugging profile
TYPE
The type of debugging profile.

Values for the parameter are:
C
E
J
N
USERID
The user ID specified in the debugging profile
ACTIVATE_USERID
Optional Parameter

For an active debugging profile, the user ID of the user who made it active.

CURRENT_PAGE
Optional Parameter

Specifies which page of the list of profiles is currently displayed

INVALID_INPUT
Optional Parameter

Whatever was (invalidly) typed as an input

PATTERN_MATCH_NUMBER
Optional Parameter

A metric computed from the contents of the debugging profile, which is compared with the pattern match number form other profiles to determine which of the profiles is the best match for a program instance.

PROFILE_NUMBER
Optional Parameter

The position of the current profile in the list

DPLM gate, READNEXT_INPUT function

When inputs are typed in against profiles they are saved with the profile in the linked list so that they are still retrievable for redisplay after scrolling up and down. READNEXT_INPUT allows easy retrieval of just those profiles with inputs against them so that they can be processed when enter is pressed. All the data in the profile is returned as it is required if the input to be processed is COPY.

Input Parameters

BROWSE_LIST_TOKEN
A token which uniquely identifies the list of profiles.

INPUT_FILTER
Specifies profiles of interest, based on any actions that have been specified for the profile.

Values for the parameter are:

```
ACTIVATES
ALL_INPUTS
COPIES
DELETES
INACTIVATES
```

BEAN_BLOCK
Optional Parameter

A block of storage containing the bean name

CLASS_BLOCK
Optional Parameter

A block of storage containing the class name

LE_OPTIONS_BLOCK
Optional Parameter

A block of storage containing Language Environment options

MANGLED_METHOD_BLOCK
Optional Parameter

A block of storage containing the mangled method name

METHOD_BLOCK
Optional Parameter

A block of storage containing the method name

POSITION
Optional Parameter

Specifies the position in the list of the next profile to be read.

Values for the parameter are:
NEXT_PROFILE
TOP

Output Parameters

REASON
The values for the parameter are:
ABEND
DISASTER_PERCOLATION
END_OF_INPUTS
INTERNAL_ERROR

APPLID
The Applid specified in the debugging profile

COMMAND_FILE
The command file specified in the debugging profile

COMP_UNIT
The compilation unit name specified in the debugging profile

INPUT
The action specified for the profile.

Values for the parameter are:
ACTIVATE
CLEAR
COPY
DELETE
INACTIVATE

JVM_PROFILE
The JVM profile specified in the debugging profile

NETNAME
The terminal's network name specified in the debugging profile

OWNER_USERID
The userid of the profile's owner

PREFERENCE_FILE
The preference file specified in the debugging profile

PROFILE_NAME
The name of the debugging profile

PROGRAM
The program name specified in the debugging profile

PROMPT
The prompt specified in the debugging profile

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATUS
The status of the debugging profile.

Values for the parameter are:
ACTIVE
INACTIVE

TERMID
The terminal ID specified in the debugging profile

TEST_LEVEL
The test level specified in the debugging profile.

Values for the parameter are:

ALL
ERROR
NONE

TRANID
The transaction ID specified in the debugging profile

TYPE
The type of debugging profile.

Values for the parameter are:

C
E
J
N

USERID
The user ID specified in the debugging profile

ACTIVATE_USERID
Optional Parameter

For an active debugging profile, the user ID of the user who made it active.

CURRENT_PAGE
Optional Parameter

Specifies which page of the list of profiles is currently displayed

INVALID_INPUT
Optional Parameter

Whatever was (invalidly) typed as an input

PATTERN_MATCH_NUMBER
Optional Parameter

A metric computed from the contents of the debugging profile, which is compared with the pattern match number form other profiles to determine which of the profiles is the best match for a program instance.

DPLM gate, RESTARTBR_DEBUG_PROFILES function

Resume browsing a list of debugging profiles.

Input Parameters

BROWSE_LIST_TOKEN
A token which uniquely identifies the list of profiles.

CURRENT_USERID
The userid of the user making the request

Output Parameters

REASON
The values for the parameter are:

ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
NO_PROFILES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CURRENT_PAGE
Optional Parameter

Specifies which page of the list of profiles is currently displayed

NUMBER_IN_LIST
Optional Parameter

The number of profiles in the list

DPLM gate, STARTBR_DEBUG_PROFILES function

Start browsing a list of debug profiles.

Input Parameters

CURRENT_USERID
The userid of the user making the request

FILTER_ACTIVE
Specifies whether the list contains active profiles only, or active and inactive profiles.

Values for the parameter are:
```
ACTIVE_P
ALL_P
```

FILTER_USER
Specifies whether the list contains profiles for just the current user, or all users.

Values for the parameter are:
```
ALL_U
CURRENT_USER
```

SORT_TYPE
Specifies the field used to sort the list.

Values for the parameter are:
```
APPL
COMP_U
NAME
NETN
OWNER
PROG
STAT
TERM
TRAN
TYP
USER
```

Output Parameters

REASON
The values for the parameter are:
```
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
NO_PROFILES
```

BROWSE_LIST_TOKEN
A token which uniquely identifies the list of profiles.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CURRENT_PAGE
Optional Parameter

Specifies which page of the list of profiles is currently displayed

NUMBER_IN_LIST
Optional Parameter

The number of profiles in the list

DPLM gate, UPDATE_PROFILE_IN_LIST function

Update the specified in-memory linked list element with the input supplied so that it may be kept until ready to process later. CLEAR may be used to clear an input that has been handled.

Input Parameters

BROWSE_LIST_TOKEN

A token which uniquely identifies the list of profiles.

INPUT

The action specified for the profile.

Values for the parameter are:

```
ACTIVATE
CLEAR
COPY
DELETE
INACTIVATE
```

OWNER_USERID

The userid of the debugging profile's owner

PROFILE_NAME

The name of the debugging profile

INVALID_INPUT

Optional Parameter

An invalid action character that cannot be interpreted as one of the values of the INPUT parameter.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
DISASTER_PERCOLATION
INTERNAL_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CURRENT_PAGE

Optional Parameter

Specifies which page of the list of profiles is currently displayed

DPPM gate, PATTERN_MATCH_PROFILE function

Determines if an active debugging profile matches the parameters supplied.

Input Parameters

MATCH_TYPE

The type of debugging profile.

Values for the parameter are:

```
LE
NON_LE
```

APPLID

Optional Parameter

The Applid specified in the debugging profile

BEAN_BLOCK

Optional Parameter

A block of storage containing the bean name

CLASS_BLOCK
Optional Parameter

A block of storage containing the class name

COMP_UNIT
Optional Parameter

The compilation unit name specified in the debugging profile

IP_NAME_OR_ADDR_BLOCK
Optional Parameter

A block of storage containing the IP name or IP address

LE_OPTIONS_BLOCK
Optional Parameter

A block of storage containing Language Environment options

MANGLED_METHOD_BLOCK
Optional Parameter

A block of storage containing the mangled method name

NETNAME
Optional Parameter

The terminal's network name specified in the debugging profile

PROGRAM
Optional Parameter

The program name specified in the debugging profile

TERMID
Optional Parameter

The terminal ID specified in the debugging profile

TRANID
Optional Parameter

The transaction ID specified in the debugging profile

USERID
Optional Parameter

The user ID specified in the debugging profile

Output Parameters

REASON
The values for the parameter are:

```
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
NO_MATCH
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

COMMAND_FILE
Optional Parameter

The command file specified in the debugging profile

JVM_PROFILE
Optional Parameter

The JVM profile specified in the debugging profile

LU_3270_DISPLAY
Optional Parameter

The 3270 display terminal to be used by Debug Tool

PORT
Optional Parameter

The port number specified in the debugging profile

PREFERENCE_FILE
Optional Parameter

The preference file specified in the debugging profile

PROFILE_APPLID
Optional Parameter

The Applid specified in the matching profile

PROFILE_COMP_UNIT
Optional Parameter

The compilation unit name specified in the matching profile

PROFILE_NETNAME
Optional Parameter

The terminal's network name specified in the matching profile

PROFILE_PROGRAM
Optional Parameter

The program name specified in the matching profile

PROFILE_TERMID
Optional Parameter

The terminal ID specified in the matching profile

PROFILE_TRANID
Optional Parameter

The transaction ID specified in the matching profile

PROFILE_USERID
Optional Parameter

The user ID specified in the matching profile

PROMPT
Optional Parameter

The prompt specified in the debugging profile

SESSION_TYPE
Optional Parameter

The session type specified in the debugging profile.

Values for the parameter are:

```
LU3270
TCP
```

SOCKET_TYPE
Optional Parameter

Specifies whether the debugging client and debugging server will communicate using a single socket or more than one socket.

Values for the parameter are:

```
MULTIPLE
SINGLE
```

TEST_LEVEL
Optional Parameter

The test level specified in the debugging profile.

Values for the parameter are:
ALL
ERROR
NONE

DPPM gate, PATTERN_MATCH_TASK function

Determines if an active debugging profile matches the parameters supplied.

Input Parameters

APPLID
The Applid specified in the debugging profile

NETNAME
The terminal's network name specified in the debugging profile

TERMID
The terminal ID specified in the debugging profile

TRANID
The transaction ID specified in the debugging profile

USERID
The user ID specified in the debugging profile

Output Parameters

REASON
The values for the parameter are:
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
NO_MATCH

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPUM gate, GET_USER_DEFAULTS function

Get user defaults. If none already, returns global defaults.

Input Parameters

CURRENT_SESSION_TYPE
The session type specified for the current user.

Values for the parameter are:
LU3270
TCP

CURRENT_USERID
The userid of the user making the request

CURRENT_TERMID
Optional Parameter

The TERMID of the terminal making the request.

IP_NAME_OR_ADDR_BLOCK
Optional Parameter

A block of storage containing the IP name or IP address

LE_OPTIONS_BLOCK
Optional Parameter

A block of storage containing Language Environment options

Output Parameters

REASON

The values for the parameter are:

```
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

COMMAND_FILE

Optional Parameter

The command file specified in the debugging profile

FILTER_ACTIVE

Optional Parameter

Specifies whether the list contains active profiles only, or active and inactive profiles.

Values for the parameter are:

```
ACTIVE_P
ALL_P
```

FILTER_USER

Optional Parameter

Specifies whether the list contains profiles for just the current user, or all users.

Values for the parameter are:

```
ALL_U
CURRENT_USER
```

JVM_PROFILE

Optional Parameter

The JVM profile specified in the debugging profile

LU_3270_DISPLAY

Optional Parameter

The 3270 display terminal to be used by Debug Tool

PORT

Optional Parameter

The port number specified in the debugging profile

PREFERENCE_FILE

Optional Parameter

The preference file specified in the debugging profile

PROMPT

Optional Parameter

The prompt specified in the debugging profile

SESSION_TYPE

Optional Parameter

The session type specified in the debugging profile.

Values for the parameter are:

```
LU3270
TCP
```

SOCKET_TYPE

Optional Parameter

Specifies whether the debugging client and debugging server will communicate using a single socket or more than one socket.

Values for the parameter are:
MULTIPLE
SINGLE

SORT_TYPE
Optional Parameter

Specifies the field used to sort the list.

Values for the parameter are:
APPL
COMP_U
NAME
NETN
OWNER
PROG
STAT
TERM
TRAN
TYP
USER

SUPPRESS_PANEL
Optional Parameter

Specifies whether the debugging device panel is to be suppressed.

Values for the parameter are:
NOSUPPRESS
SUPPRESS

TEST_LEVEL
Optional Parameter

The test level specified in the debugging profile.

Values for the parameter are:
ALL
ERROR
NONE

TYPE
Optional Parameter

The type of debugging profile.

Values for the parameter are:
C
E
J
LE

DPUM gate, SAVE_USER_DEFAULTS function

Save user defaults. Never returns duplicate response - saves or updates.

Input Parameters

CURRENT_USERID
The userid of the user making the request

COMMAND_FILE
Optional Parameter

The command file specified in the debugging profile

FILTER_ACTIVE
Optional Parameter

Specifies whether the list contains active profiles only, or active and inactive profiles.

Values for the parameter are:
ACTIVE_P
ALL_P

FILTER_USER
Optional Parameter

Specifies whether the list contains profiles for just the current user, or all users.

Values for the parameter are:
ALL_U
CURRENT_USER

IP_NAME_OR_ADDR_BLOCK
Optional Parameter

A block of storage containing the IP name or IP address

JVM_PROFILE
Optional Parameter

The JVM profile specified in the debugging profile

LE_OPTIONS_BLOCK
Optional Parameter

A block of storage containing Language Environment options

LU_3270_DISPLAY
Optional Parameter

The 3270 display terminal specified in the debugging profile to be used by Debug Tool

PORT
Optional Parameter

The port number specified in the debugging profile

PREFERENCE_FILE
Optional Parameter

The preference file specified in the debugging profile

PROMPT
Optional Parameter

The prompt specified in the debugging profile

SESSION_TYPE
Optional Parameter

The session type specified in the debugging profile.

Values for the parameter are:
LU3270
TCP

SOCKET_TYPE
Optional Parameter

Specifies whether the debugging client and debugging server will communicate using a single socket or more than one socket.

Values for the parameter are:
MULTIPLE
SINGLE

SORT_TYPE
Optional Parameter

Specifies the field used to sort the list.

Values for the parameter are:
- APPL
- COMP_U
- NAME
- NETN
- OWNER
- PROG
- STAT
- TERM
- TRAN
- TYP
- USER

SUPPRESS_PANEL
Optional Parameter

Specifies whether the debugging device panel is to be suppressed.

Values for the parameter are:
- NOSUPPRESS
- SUPPRESS

TEST_LEVEL
Optional Parameter

The test level specified in the debugging profile.

Values for the parameter are:
- ALL
- ERROR
- NONE

TYPE
Optional Parameter

The type of debugging profile.

Values for the parameter are:
- C
- E
- J
- LE

Output Parameters

REASON
The values for the parameter are:
- ABEND
- CMD_FILE_INVALID
- DISASTER_PERCOLATION
- FILE_ERROR
- FILE_FULL
- INTERNAL_ERROR
- IP_BLANK
- IP_INVALID
- JVM_PROFILE_INVALID
- PORT_BLANK
- PORT_INVALID
- PREF_FILE_INVALID

PROMPT_INVALID
3270_DISPLAY_BLANK
3270_DISPLAY_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPWD gate, PROCESS_PAGE function

Process a request for an html page in the following format:

Input Parameters

ITOKEN
A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

PAGE
The page to be processed

MSG_INSERT1
Optional Parameter

An insert for the message. If this field is null there is no first insert.

MSG_INSERT2
Optional Parameter

An insert for the message. If this field is null there is no second insert.

MSG_NUM
Optional Parameter

The message number of a message to be displayed when the page is formatted.

MSG_TYPE
Optional Parameter

The type of message to be displayed when the page is formatted, in the absence of a more serious message. If this value is not present then by default no message is displayed.

Values for the parameter are:
ERROR
INFO

Output Parameters

REASON
The values for the parameter are:
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR

OTOKEN
A token representing a chain of output html tags.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPWD gate, PROCESS_SUBMIT function

Process a submitted form request. The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input Parameters

BUTTON
The action button used to submit the form.

ITOKEN
A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
```

OTOKEN
A token representing a chain of output html tags.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPWE gate, PROCESS_PAGE function

Process a request for an html page in the following format:

Input Parameters

ITOKEN
A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

PAGE
The page to be processed

MSG_INSERT1
Optional Parameter

An insert for the message. If this field is null there is no first insert.

MSG_INSERT2
Optional Parameter

An insert for the message. If this field is null there is no second insert.

MSG_NUM
Optional Parameter

The message number of a message to be displayed when the page is formatted.

MSG_TYPE
Optional Parameter

The type of message to be displayed when the page is formatted, in the absence of a more serious message. If this value is not present then by default no message is displayed.

Values for the parameter are:

```
ERROR
INFO
```

Output Parameters

REASON
The values for the parameter are:

```
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
```

OTOKEN
A token representing a chain of output html tags.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPWE gate, PROCESS_SUBMIT function

Process a submitted form request. The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input Parameters

BUTTON
The action button used to submit the form.
ITOKEN
A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
```

OTOKEN
A token representing a chain of output html tags.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPWJ gate, PROCESS_PAGE function

Process a request for an html page in the following format:

Input Parameters

ITOKEN
A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.
PAGE
The page to be processed
MSG_INSERT1
Optional Parameter

An insert for the message. If this field is null there is no first insert.
MSG_INSERT2
Optional Parameter

An insert for the message. If this field is null there is no second insert.
MSG_NUM
Optional Parameter

The message number of a message to be displayed when the page is formatted.
MSG_TYPE
Optional Parameter

The type of message to be displayed when the page is formatted, in the absence of a more serious message. If this value is not present then by default no message is displayed.

Values for the parameter are:
ERROR
INFO

Output Parameters

REASON

The values for the parameter are:
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR

OTOKEN

A token representing a chain of output html tags.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPWJ gate, PROCESS_SUBMIT function

Process a submitted form request. The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input Parameters

BUTTON

The action button used to submit the form.

ITOKEN

A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

Output Parameters

REASON

The values for the parameter are:
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR

OTOKEN

A token representing a chain of output html tags.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPWL gate, PROCESS_PAGE function

Process a request for an html page in the following format:

Input Parameters

ITOKEN

A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

PAGE

The page to be processed

MSG_INSERT1

Optional Parameter

An insert for the message. If this field is null there is no first insert.

MSG_INSERT2

Optional Parameter

An insert for the message. If this field is null there is no second insert.

MSG_NUM

Optional Parameter

The message number of a message to be displayed when the page is formatted.

MSG_TYPE

Optional Parameter

The type of message to be displayed when the page is formatted, in the absence of a more serious message. If this value is not present then by default no message is displayed.

Values for the parameter are:

```
ERROR
INFO
```

Output Parameters

REASON

The values for the parameter are:

```
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
```

OTOKEN

A token representing a chain of output html tags.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPWL gate, PROCESS_SUBMIT function

Process a submitted form request. The input options will be read by the page processor from ITOKEN. The page processor will generate an output page request in OTOKEN.

Input Parameters

BUTTON

The action button used to submit the form.

ITOKEN

A token representing a chain of input values. These are name-value pairs from either the page options, or from the form.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
DISASTER_PERCOLATION
FILE_ERROR
INTERNAL_ERROR
```

OTOKEN

A token representing a chain of output html tags.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPXM gate, BIND_XM_CLIENT function

The BIND_XM_CLIENT call flows from the transaction manager to the DP Domain during transaction initialization after Recovery Manager initialization is complete. The DP domain does a scan of the active debugging profiles to determine if it is

possible that debugging could be required in this transaction. If it is not, DP domain is not called again until transaction termination.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPXM gate, INIT_XM_CLIENT function

The INIT_XM_CLIENT call flows from the transaction manager to the DP Domain during transaction initialization. The DP domain allocates the DP domain transaction lifetime control block, and anchors it in the AP domain's transaction token.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DPXM gate, RELEASE_XM_CLIENT function

The RELEASE_XM_CLIENT call is made from the transaction manager to the DP Domain during transaction termination. DP domain transaction lifetime resources are released.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Debugging profile domain's generic gates

Table 16 on page 454 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 16. Debugging profile domain's generic gates

Gate	Trace	Functions	Format
DPDM	DP 0101 DP 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DPDM

Dispatcher domain (DS)

The dispatcher domain is concerned with the attaching, running, and detaching of tasks, and the posting of TCBs.

The domain posts TCBs with the following modes:

Table 17. TCB modes

Mode	Description
CO	Concurrent
D2	DB2
EP	Event processing
FO	File-owning
L8	CICS key OPENAPI programs
L9	User key OPENAPI programs
QR	Quasi-reentrant
RO	Resource-owning
RP	ONC/RPC-owning
SO	Sockets
SL	Sockets listener
S8	Secure sockets key 8
SP	SSL pool owner
SZ	Secondary LU usage
TP	JVM server thread pool owner
T8	JVM server threads
X8	XPLINK CICS key
X9	XPLINK user key

Dispatcher domain's specific gates

The specific gates provide access for other domains to functions that are provided by the DS domain.

DSAC gate, ASSOCIATE function

Initializes the dispatcher association functions.

Input Parameters

ASSOCIATION_TOKEN
The association token

ASSOC_FLAT_TOKEN
The flat version of the association token

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_ASSOCIATION_TOKEN
INVALID_ASSOCIATION_STATE
INCORRECT_TCB
WAIT_TASK_CANCELLED
WAIT_TASK_TIMED_OUT
ERROR_IN_OTHER_FUNCTION
DISASSOCIATE_INVOKED
FUNCTION_ALREADY_INVOKED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see RESPONSE parameter on domain interfaces.

DSAC gate, DEFAULT_TASK_ASSOC function

Performs default task processing for association.

Input Parameters

ASSOCIATION_TOKEN

The association token

ASSOCIATION_TYPE

The type of association, either

NONE, WAIT or ASSOC

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see RESPONSE parameter on domain interfaces.

DSAC gate, ASSOCIATE_INIT function

Initialises the dispatcher association functions.

Input Parameters

None

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see RESPONSE parameter on domain interfaces.

DSAC gate, PREPARE_FOR_ASSOCIATION function

Prepares the task for association.

Input Parameters

PURGEABLE

The purgeable status of the task

Values for the parameter are:
NO
YES

INTERVAL
Optional Parameter

An interval (in units as specified by TIME_UNIT) after which the task is given back control if the association has not been successful.

TIME_UNIT
Optional Parameter

Identifies the time units specified on the INTERVAL and DELAY parameters where present.

Values for the parameter are:
MILLI_SECOND
SECOND

Output Parameters

ASSOCIATION_TOKEN
The association token

ASSOC_FLAT_TOKEN
The flat version of the association token.

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE

The following value are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see RESPONSE parameter on domain interfaces.

DSAC gate, WAIT_FOR_ASSOCIATION function

Waits for the association function to complete.

Input Parameters

ASSOCIATION_TOKEN
The association token

ASSOC_FLAT_TOKEN
The flat version of the association token

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_ASSOCIATION_TOKEN
INVALID_ASSOCIATION_STATE
ERROR_IN_OTHER_FUNCTION
DISASSOCIATE_INVOKED
FUNCTION_ALREADY_INVOKED
WAIT_TASK_CANCELLED
WAIT_TASK_TIMED_OUT

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see RESPONSE parameter on domain interfaces.

DSAT gate, DISASSOCIATE function

Disassociates the wait_for_association task from its TCB.

Input Parameters

ASSOCIATION_TOKEN
The association token

ASSOC_FLAT_TOKEN
The flat version of the association token

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_ASSOCIATION_TOKEN
INVALID_ASSOCIATION_STATE
INCORRECT_TCB
WAIT_TASK_CANCELLED
WAIT_TASK_TIMED_OUT
ERROR_IN_OTHER_FUNCTION
DISASSOCIATE_INVOKED
FUNCTION_ALREADY_INVOKED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see RESPONSE parameter on domain interfaces.

DSAT gate, ATTACH function

The ATTACH function of the DSAT gate is used to attach a new task.

Input Parameters

PRIORITY
affects a task's dispatching precedence. It can have a value in the range 0 (low priority) through 255 (high priority).

TYPE
is the type of task.

Values for the parameter are:

```
NON_SYSTEM
SYSTEM
```

USER_TOKEN
is the token by which the task to be attached is known to the caller.

MODE
Optional Parameter

specifies the mode in which the task is to run.

Values for the parameter are:

```
CO
FO
QR
RO
RP
```

SZ

SPECIAL_TYPE
Optional Parameter

identifies the special task SMSY.

Values for the parameter are:
SMSY

TASK_REPLY_GATE_INDEX
Optional Parameter

is used when a gate other than the attaching domain's default gate is to receive a resultant TASK_REPLY.

TIMEOUT
Optional Parameter

is the deadlock timeout interval, in milliseconds.

TRANSACTION_TOKEN
Optional Parameter

identifies the transaction associated with the attached task.

TRANSACTION_BUILD
Optional Parameter

Values for the parameter are:
BUILD
TERMINATE

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP
TRANSACTION_ABEND
TRANSACTION_ERROR_MSG
USER_TASK_SLOT_UNAVAILABLE

The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TASK_TOKEN
is the token by which the attached task is known to the dispatcher.

DSAT gate, CANCEL_TASK function

The CANCEL_TASK function of DSAT gate causes a specified task to be canceled. The task is cancelled when in a suitable suspend or when a deferred abend can be delivered to the task.

Input Parameters

CANCEL_TYPE
Specifies when the task can be canceled having regard to system integrity and data integrity.

Values for the parameter are:
FORCE_CANCEL
KILL_CANCEL
NORMAL_CANCEL

DEFERRED_ABEND_CODE
is the abend code to be used when the task is abended during deferred abend processing.

TASK_TOKEN
identifies the task whose priority is to be changed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CANCEL_INHIBITED
INVALID_STATE
INVALID_STATE_PURGE
INVALID_TASK_TOKEN
NOT_PURGEABLE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSAT gate, CHANGE_MODE function

The CHANGE_MODE function of DSAT gate is used to move a task from one CICS-managed TCB to another, or to select the mode in which the task is to run.

Input Parameters

MODE
specifies the mode in which the task is to run:

CO concurrent mode
FO file-owning mode
QR quasi-reentrant mode
RO resource-owning mode
RP ONC/RPC-owning mode
SZ secondary LU usage mode

MODENAME
2-character mode name.

MODENAME_TOKEN
token representing modename. More efficient than using MODENAME. The token is returned by ACTIVATE_MODE and by CHANGE_MODE (see OLD_MODENAME_TOKEN)

TCB_TOKEN
token representing the TCB instance to which to switch. The token is returned by CHANGE_MODE (see OLD_TCB_TOKEN)

CONDITIONAL
Optional Parameter

states whether the CHANGE_MODE should be conditional on the current load on the CPU.

Values for the parameter are:

```
NO
YES
```

DISASSOCIATE_TCB
Optional Parameter

indicates whether to disassociate the task from the TCB from which the switch is made.

Values for the parameter are:

```
NO
YES
```

FRESH_TCB
Optional Parameter

indicates whether a fresh TCB is required.

Values for the parameter are:
NO
YES

MATCH_STRATEGY
Optional Parameter

the strategy to be followed if a TCB instance that satisfies the PRIMARY_MATCH and SECONDARY_MATCH values is not found.

Values for the parameter are:
EXACT_THEN_NEW_THEN_BEST

PRIMARY_MATCH
Optional Parameter

an 8-byte token to be used to search for a matching free TCB instance to which to switch.

SECONDARY_MATCH
Optional Parameter

an 8-byte token to be used to search for a matching free TCB instance to which to switch.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ACTIVATE_MODE_FAILED
ADD_TCB_FAILED
LOCK_FAILED
SUSPEND_FAILED

The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
MODE_NOT_ACTIVE
NO_TCBS_ACTIVE
NOT_OPEN_MODE_TCB
TCB_FAILED
TOO_FEW_TCBS

The following values are returned when RESPONSE is INVALID:
INVALID_FRESH_TCB_USAGE
INVALID_MODENAME
INVALID_MODENAME_TOKEN
INVALID_TCB_TOKEN

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MATCH_RESULT
Optional Parameter

indicates the level of success of the matching process.

Values for the parameter are:

EXACT_MATCH
NO_MATCH
NOT_APPLIC
PRIM_NOT_SEC_MATCH

NEW_TCB_TOKEN
Optional Parameter

token representing the TCB instance returned by the matching process.

OLD_MODE
Optional Parameter

is the mode used by the task when the CHANGE_MODE request was issued.

Values for the parameter are:
CO
FO
QR
RO
RP
SZ

OLD_MODENAME
Optional Parameter

is the mode used by the task when the CHANGE_MODE request was issued. It can have the same values as OLD_MODE. OLD_MODENAME is preferred to OLD_MODE.

OLD_MODENAME_TOKEN
Optional Parameter

is a token representing the mode used by the task when the CHANGE_MODE request was issued.

OLD_TCB_TOKEN
Optional Parameter

is a token representing the TCB used by the task when the CHANGE_MODE request was issued.

DSAT gate, CHANGE_PRIORITY function

The CHANGE_PRIORITY function of DSAT gate has two effects. It changes the dispatch priority of the issuing task, and it causes control to be given up to another task.

Input Parameters

PRIORITY
Optional Parameter

The dispatching precedence of a task. This parameter can have a value in the range 0 (low priority) through 255 (high priority).

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON
Optional Parameter

The values for the parameter are as follows:
ABEND
ACTIVATE_MODE_FAILED
ADD_TCB_FAILED

CANCEL_INHIBITED
INSUFFICIENT_STORAGE
INVALID_FORMAT
INVALID_FRESH_TCB_USAGE
INVALID_FUNCTION
INVALID_MODENAME
INVALID_MODENAME_TOKEN
INVALID_STATE
INVALID_STATE_PURGE
INVALID_TASK_TOKEN
INVALID_TCB_TOKEN
LOCK_FAILED
LOOP
MODE_NOT_ACTIVE
NO_TCBS_ACTIVE
NOT_OPEN_MODE_TCB
NOT_PURGEABLE
NOT_SUBSPACE_ELIGIBLE
SUSPEND_FAILED
TASK_CANCELLED
TCB_FAILED
TCB_NOT_OWNED
TIMED_OUT
TOO_FEW_TCBS
USER_TASK_SLOT_UNAVAILABLE

OLD_PRIORITY
Optional Parameter

The former priority of the task. This parameter can have a value in the range 0 (low priority) through 255 (high priority).

DSAT gate, CLEAR_MATCH function

The CLEAR_MATCH function of the DSAT gate causes all match tokens associated with the calling TCB to be discarded.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSAT gate, DELETE_SUBSPACE_TCBS function

The DELETE_SUBSPACE_TCBS function of DSAT gate deletes any open TCBs associated with the given subspace.

Input Parameters

SUBSPACE_TOKEN
indicates the subspace whose associated open TCBs are to be deleted

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOCK_FAILED

The following values are returned when RESPONSE is EXCEPTION:
TOO_FEW_TCBS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSAT gate, FREE_SUBSPACE_TCBS function

The FREE_SUBSPACE_TCBS function of DSAT gate releases any open subspace TCBs owned by the task, and makes them available for use by another task executing with the same subspace, or deletes the TCBs if the task is 'unclean'.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOCK_FAILED

The following values are returned when RESPONSE is INVALID:
NOT_SUBSPACE_ELIGIBLE

OPEN_TCBS_USED_AND_KEPT
is a bit string indicating which TCB modes were used by the task, of and are now available to other tasks

OPEN_TCBS_USED_AND_LOST
is a bit string indicating which TCB modes were used by the task, of and have now been deleted because the task was 'unclean'

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSAT gate, RELEASE_OPEN_TCB function

The RELEASE_OPEN_TCB function of DSAT gate frees the TCB from the calling task's ownership.

Input Parameters

TCB_TOKEN
token representing the TCB instance to which to switch. The token is returned by CHANGE_MODE (see OLD_TCB_TOKEN)

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOCK_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_TCB_TOKEN
TCB_NOT_OWNED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSAT gate, SET_PRIORITY function

The SET_PRIORITY function of DSAT gate changes the priority of the issuing task, or the task specified by the TASK_TOKEN parameter.

Input Parameters

PRIORITY
affects a task's dispatching precedence. It can have a value in the range 0 (low priority) through 255 (high priority).

SPECIAL_TYPE
Optional Parameter

identifies the special task IMMEDIATE_SHUTDOWN_TASK.

Values for the parameter are:
IMMEDIATE_SHUTDOWN_TASK

TASK_TOKEN
Optional Parameter

identifies the task whose priority is to be changed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_TASK_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OLD_PRIORITY
Optional Parameter

is the task's former priority. It can have a value in the range 0 (low priority) through 255 (high priority).

DSAT gate, SET_TRANSACTION_TOKEN function

The SET_TRANSACTION_TOKEN function of DSAT gate sets the XM domain transaction token of the transaction associated with the currently dispatched task.

Input Parameters

TRANSACTION_TOKEN
identifies the transaction associated with the attached task.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON
Optional Parameter

The values for the parameter are:
ABEND
ACTIVATE_MODE_FAILED
ADD_TCB_FAILED
CANCEL_INHIBITED
INSUFFICIENT_STORAGE
INVALID_FORMAT
INVALID_FRESH_TCB_USAGE
INVALID_FUNCTION
INVALID_MODENAME
INVALID_MODENAME_TOKEN
INVALID_STATE
INVALID_STATE_PURGE
INVALID_TASK_TOKEN
INVALID_TCB_TOKEN
LOCK_FAILED
LOOP
MODE_NOT_ACTIVE
NO_TCBS_ACTIVE
NOT_OPEN_MODE_TCB
NOT_PURGEABLE

NOT_SUBSPACE_ELIGIBLE
SUSPEND_FAILED
TASK_CANCELLED
TCB_FAILED
TCB_NOT_OWNED
TIMED_OUT
TOO_FEW_TCBS
USER_TASK_SLOT_UNAVAILABLE

DSAT gate, TCB_POOL_MANAGEMENT function

The TCB_POOL_MANAGEMENT function of DSAT gate deletes unallocated TCBs which are excess to current requirements.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOCK_FAILED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSBR gate, END_BROWSE function

The END_BROWSE function of DSBR gate ends a browse session with the dispatcher.

Input Parameters

BROWSE_TOKEN

is the token identifying the browse session to be ended.

Output Parameters

REASON

The values for the parameter are:

INVALID_BROWSE_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSBR gate, GET_NEXT function

The GET_NEXT function of DSBR gate returns information about the next task.

Input Parameters

BROWSE_TOKEN

is the token identifying the browse session to be ended.

Output Parameters

REASON

The values for the parameter are:

END
INVALID_BROWSE_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DOMAIN_INDEX

Optional Parameter

is the 2-character index identifying the domain that made the ATTACH call for the task.

ESSENTIAL_TCB
Optional Parameter

indicates whether the TCB is an essential TCB or not.

Values for the parameter are:
ESSENTIAL_NO
ESSENTIAL_YES

KERNEL_TOKEN
Optional Parameter

is the token by which the task is known to the kernel.

MODE
Optional Parameter

is the mode in which the task is to run.

Values for the parameter are:
CO
FO
QR
RO
RP
SZ

OPEN_MODES
Optional Parameter

is a 32-bit string which indicates which modes of open TCBs were used by this task.

PRIORITY
Optional Parameter

is the task's dispatch priority. It can have a value in the range 0 (low priority) through 255 (high priority).

RESOURCE_NAME
Optional Parameter

is the name of the resource that the task is waiting for, if the task is suspended.

RESOURCE_TIME
Optional Parameter

is the interval of time that has passed since the task last issued a suspend or wait.

RESOURCE_TYPE
Optional Parameter

is the type of resource that the task is waiting for, if the task is suspended.

STATE
Optional Parameter

is the state of the task.

Values for the parameter are:
READY
RUNNING
SUSPENDED

SUSPEND_TOKEN
Optional Parameter

is the token by which the dispatcher recognizes a task to be suspended or resumed.

TASK_TOKEN
Optional Parameter

is the token by which the attached task is known to the dispatcher.

TCB_TOKEN
Optional Parameter

is the TCB token associated with the task.

TCB_TYPE
Optional Parameter

is the type of TCB that the task is executing on.

Values for the parameter are:
- CKOPEN_TCB
- INTERNAL_TCB
- QR_TCB
- UKOPEN_TCB

TYPE
Optional Parameter

is the type of task.

Values for the parameter are:
- NON_SYSTEM
- SYSTEM

USER_TOKEN
Optional Parameter

is the token by which the task is known to the caller that made the ATTACH request for the task.

DSBR gate, INQUIRE_TASK function

The INQUIRE_TASK function of DSBR gate returns information about a specified task.

Input Parameters

INPUT_TASK_TOKEN
Optional Parameter

is the token for the task to be inquired on.

Output Parameters

REASON
The values for the parameter are:
- INVALID_TASK_TOKEN
- NOT_SUPPORTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CANCEL_PENDING
Optional Parameter

Not supported by domain gate function.

Values for the parameter are:
- CLEARED
- FORCE
- KILL
- NONE
- NORMAL

DEFERRED_ABEND_CODE
Optional Parameter

Not supported by domain gate function.

DOMAIN_INDEX
Optional Parameter

is the 2-character index identifying the domain that made the ATTACH call for the task.

ESSENTIAL_TCB
Optional Parameter

indicates whether the TCB is an essential TCB or not.

Values for the parameter are:
```
ESSENTIAL_NO
ESSENTIAL_YES
```

KERNEL_TOKEN
Optional Parameter

is the token by which the task is known to the kernel.

MODE
Optional Parameter

is the mode in which the task is to run.

Values for the parameter are:
```
CO
FO
QR
RO
RP
SZ
```

OPEN_MODES
Optional Parameter

is a 32-bit string which indicates which modes of open TCBs were used by this task.

PRIORITY
Optional Parameter

is the task's dispatch priority. It can have a value in the range 0 (low priority) through 255 (high priority).

RESOURCE_NAME
Optional Parameter

is the name of the resource that the task is waiting for, if the task is suspended.

RESOURCE_TIME
Optional Parameter

is the interval of time that has passed since the task last issued a suspend or wait.

RESOURCE_TYPE
Optional Parameter

is the type of resource that the task is waiting for, if the task is suspended.

STATE
Optional Parameter

is the state of the task.

Values for the parameter are:
```
READY
```

RUNNING
SUSPENDED

SUSPEND_TOKEN
Optional Parameter

is the token by which the dispatcher recognizes a task to be suspended or resumed.

TASK_TOKEN
Optional Parameter

is the token by which the attached task is known to the dispatcher.

TCB_TOKEN
Optional Parameter

is the TCB token associated with the task.

TCB_TYPE
Optional Parameter

is the type of TCB that the task is executing on.

Values for the parameter are:
CKOPEN_TCB
INTERNAL_TCB
QR_TCB
UKOPEN_TCB

TYPE
Optional Parameter

is the type of task.

Values for the parameter are:
NON_SYSTEM
SYSTEM

USER_TOKEN
Optional Parameter

is the token by which the task is known to the caller that made the ATTACH request for the task.

DSBR gate, INQUIRE_TCB function

The INQUIRE_TCB function of the DSBR gate returns the AP TCB-related token associated with the specified DS TCB_TOKEN. If the AP token has not yet been set by SET_TCB, then the function returns an AP_TCB_TOKEN value of zero.

Input Parameters

TCB_TOKEN
Optional Parameter

token representing the TCB instance to which to switch. The token is returned by CHANGE_MODE (see OLD_TCB_TOKEN)

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_TCB_TOKEN

OWNER_TCB_TOKEN
token, provided by the TCB's owning domain, associated with the TCB instance defined by TCB_TOKEN.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSBR gate, SET_TASK function

The SET_TASK function of DSBR gate marks the task as "unclean" so that open TCBs will be freed at task termination.

Input Parameters

ABTERM_ALLOWED

Optional Parameter

Not supported by domain gate function.

Values for the parameter are:

ABTERM_NO
ABTERM_YES

CANCEL_STATE

Optional Parameter

Not supported by domain gate function.

Values for the parameter are:

FORCE
KILL
NONE
NORMAL

CLEANLINESS

Optional Parameter

specifies that the task is to be marked "unclean".

Values for the parameter are:

UNCLEAN

CLEAR_CANCEL_PENDING

Optional Parameter

Not supported by domain gate function.

Values for the parameter are:

YES

INPUT_TASK_TOKEN

Optional Parameter

is the token for the task to be inquired on.

WAIT

Optional Parameter

Not supported by domain gate function.

Values for the parameter are:

WAIT_NO
WAIT_YES

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

INVALID_TASK_TOKEN
NOT_SUPPORTED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACTION

Optional Parameter

Not supported by domain gate function.

Values for the parameter are:
ACTION_ABEND
ACTION_ABTERM
ACTION_NONE

CANCEL_PENDING
Optional Parameter

Not supported by domain gate function.

Values for the parameter are:
CLEARED
FORCE
KILL
NONE
NORMAL

DEFERRED_ABEND_CODE
Optional Parameter

Not supported by domain gate function.

DSBR gate, SET_TCB function

The SET_TCB function of the DSBR gate sets the AP TCB-related token to be associated with the running TCB.

Input Parameters

OWNER_TCB_TOKEN
token, provided by the TCB's owning domain, to be associated with the running TCB.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON
Optional Parameter

The values for the parameter are:
ABEND
END
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
INVALID_TASK_TOKEN
INVALID_TCB_TOKEN
LOOP
NOT_SUPPORTED

DSBR gate, START_BROWSE function

The START_BROWSE function of DSBR gate starts a browse session with the dispatcher.

Output Parameters

BROWSE_TOKEN
is the token representing this browse session.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSIT gate, ACTIVATE_MODE function

The ACTIVATE_MODE function creates a mode to which TCBs can be added (by ADD_TCB) so that tasks can CHANGE_MODE to the TCBs.

Input Parameters

ESSENTIAL_TCB
indicates whether CICS is to be brought down if a TCB in this mode suffers a non recoverable abend.

Values for the parameter are:
ESSENTIAL_NO
ESSENTIAL_YES

EXEC_CAPABLE
indicates whether TCBs in this mode are to be set up to support the use of EXEC CICS commands by code running on them.

Values for the parameter are:
EXEC_NO
EXEC_YES

IDENTITY
is the name of the mode to be activated. It is a two byte character string.

INHERIT_SUBSPACE
indicates whether TCBs in this mode will be able to run application code in a subspace.

Values for the parameter are:
INHERIT_NO
INHERIT_YES

LE_ENVIRONMENT
indicates whether Language Environment® is to run in native MVS mode or in CICS mode on TCBs in this mode.

Values for the parameter are:
LE_CICS
LE_MVS

MODE
specifies the mode in which the task is to run.

MODENAME
2-character mode name.

MULTIPLE_TCBS
indicates whether this mode allows more than one TCB.

Values for the parameter are:
MULTIPLE_NO
MULTIPLE_YES

OPEN
indicates whether TCBs in this mode are to be managed by the Dispatcher domain as "Open TCBs".

Values for the parameter are:
OPEN_NO
OPEN_YES

PARENT_MODENAME
the mode of the TCB that issued the request.

PRTY_RELATIVE_TO_QR
allows TCBs in this mode to have a different priority to that of the QR TCB.

TCB_KEY
indicates the key to be specified on ATTACHes of TCBs in this mode.

Values for the parameter are:
KEY8
KEY9

DEPENDENT_ON
Optional Parameter

indicates that TCBs of the mode being activated depend on the existence of TCBs of another mode.

NOTIFY_DELETE
Optional Parameter

indicates which domain, if any, to notify when a DELETE_TCB is issued. It is the binary domain index for the domain.

OPEN_POOL_NUMBER
Optional Parameter

is the number of the open TCB pool which is to contain TCBs of the newly-activated mode.

PTHREAD
Optional Parameter

indicates whether to create a protected thread.

Values for the parameter are:
PTHREAD_NO
PTHREAD_YES

SZERO
Optional Parameter

indicates whether TCBs of the new mode should be attached with SZERO(YES) or SZERO(NO).

Values for the parameter are:
SZERO_NO
SZERO_YES

WAIT_FOR_MATCH
Optional Parameter

indicates if a CHANGE_MODE should consider waiting for a suitable TCB rather than using a free TCB.

Values for the parameter are:
NEVER
NO_MODE
NO_PRIMARY

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
MODE_ALREADY_ACTIVE
MODE_LIMIT_REACHED
MODENAME_ALREADY_ACTIVE
RESERVED_MODENAME
TOO_MANY_MULTI

The following values are returned when RESPONSE is INVALID:
INVALID_MODE
INVALID_POOL_NUMBER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MODENAME_TOKEN
Optional Parameter

is a token that identifies this modename.

DSIT gate, ADD_TCB function

The ADD_TCB function adds a TCB to a particular mode.

Input Parameters

IDENTITY
is the name of the mode to be activated. It is a two byte character string.

MODENAME
2-character mode name.

MODENAME_TOKEN
token representing modename. More efficient than using MODENAME. The token is returned by ACTIVATE_MODE and by CHANGE_MODE (see OLD_MODENAME_TOKEN)

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INSUFFICIENT_STORAGE
MODE_LIMIT_REACHED
MODE_NOT_ACTIVE
RESERVED_MODENAME
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_MODENAME
INVALID_MODENAME_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TCB_TOKEN
is the TCB token associated with the task.

DSIT gate, DELETE_ALL_OPEN_TCBS function

DELETE_ALL_OPEN_TCBS schedules the termination of all open TCBs with a given modename. For TCBs that are currently in use, the termination will occur when the owning task terminates. The function does not prevent new TCBs of the given mode from being created.

Input Parameters

MODENAME
2-character mode name.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
MODE_NOT_ACTIVE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_MODENAME
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSIT gate, DELETE_OPEN_TCB function

DELETE_OPEN_TCB schedules the termination of an open TCB. If the TCB is currently in use, the termination will occur when the owning task terminates.

Input Parameters

TCB_TOKEN
is a token provided by DS that uniquely identifies the TCB.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
`INVALID_TCB_TOKEN`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSIT gate, DELETE_TCB function

The DELETE_TCB function is used by the caller to tell the Dispatcher that the TCB is to be shutdown and that the associated control blocks can be freed. If an attempt is made to shut down an essential TCB, an EXCEPTION response is returned with a reason of NOT_SUPPORTED.

Input Parameters

TCB_TOKEN
token representing the TCB instance to which to switch. The token is returned by CHANGE_MODE (see OLD_TCB_TOKEN)

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`NOT_SUPPORTED`
`TCB_IN_USE`

The following values are returned when RESPONSE is INVALID:
`INVALID_TCB_TOKEN`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSIT gate, FREE_TCB function

The FREE_TCB function is issued by the Kernel and tells the Dispatcher that a given TCB has terminated and been DETACHed.

Input Parameters

TCB_TOKEN
token representing the TCB instance to which to switch. The token is returned by CHANGE_MODE (see OLD_TCB_TOKEN)

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`INVALID_TCB_TOKEN`
`TASK_NOT_TERMINATED`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSIT gate, INQUIRE_DISPATCHER function

The INQUIRE_DISPATCHER function of DSIT gate returns information about the current state of the dispatcher.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACTOPENTCBS

Optional Parameter

is the number of TCBs in the TCB pool known as the *open pool* which are being used by current tasks.

ACTSSLTCBS

Optional Parameter

is the number of TCBs in the SSL TCB pool which are being used by current tasks.

ACTXPTCBS

Optional Parameter

is the number of TCBs in the XPLINK TCB pool which are being used by current tasks.

MAXIMUM_WAIT_INTERVAL

Optional Parameter

is the maximum delay before terminal control is dispatched.

MAXOPENTCBS

Optional Parameter

is the maximum number of TCBs in the TCB pool known as the *open pool*.

MAXSSLTCBS

Optional Parameter

is the maximum number of TCBs in the SSL TCB pool.

MAXXPTCBS

Optional Parameter

is the maximum number of TCBs in the XPLINK TCB pool.

NUMBER_OF_SUBTASKS

Optional Parameter

is the number of subtasks for concurrent mode.

PRIORITY_MULTIPLIER

Optional Parameter

determines how the priority of new tasks is to be penalized in 'storage getting short' and 'storage critical' situations.

QR_BATCHING_VALUE

Optional Parameter

is the number of POSTs for BATCH=YES waits in quasi-reentrant mode.

RP_TCB_ATTACHED

Optional Parameter

indicates whether or not the RP TCB is attached.

Values for the parameter are:

NO
YES

SCAN_DELAY_INTERVAL

Optional Parameter

is the delay before terminal control is dispatched after a terminal is posted by the access method.

SZ_TCB_ATTACHED
Optional Parameter

indicates whether or not the SZ TCB is attached.

Values for the parameter are:
NO
YES

DSIT gate, PROCESS_DEAD_TCBS function

The PROCESS_DEAD_TCBS function is issued by the SM system task each time it runs to tell the Dispatcher to process any TCBs it finds on its dead TCB chain. Such TCBs will be in an MVS WAIT issued by their ESTAE exit after suffering a non recoverable abend.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_FUNCTION
MAXOPENTCBS_OUT_OF_RANGE
MAXSSLTCBS_OUT_OF_RANGE
MAXWAIT_LESSTHAN_SCANDELAY
MAXXPTCBS_OUT_OF_RANGE
MODE_ALREADY_ACTIVE
MODE_LIMIT_REACHED
MODE_NOT_ACTIVE
MODENAME_ALREADY_ACTIVE
NOT_SUPPORTED
RESERVED_MODENAME
TASK_NOT_TERMINATED
TCB_IN_USE

The following values are returned when RESPONSE is EXCEPTION:
TOO_LATE_TO_SET_SUBTASKS
TOO_MANY_MULTI

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT

The following values are returned when RESPONSE is INVALID:
EXEC_LE_CLASH
INVALID_MODE
INVALID_MODENAME
INVALID_MODENAME_TOKEN
INVALID_POOL_NUMBER
INVALID_TCB_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSIT gate, SET_DISPATCHER function

The SET_DISPATCHER function of DSIT gate sets the state of the dispatcher.

Input Parameters

MAXIMUM_WAIT_INTERVAL
Optional Parameter

is the maximum delay before terminal control is dispatched.

MAXOPENTCBS
Optional Parameter

is the maximum number of TCBs in the TCB pool known as the *open pool*.

MAXSSLTCBS
Optional Parameter

is the maximum number of TCBs in the SSL TCB pool.

MAXXPTCBS
Optional Parameter

is the maximum number of TCBs in the XPLINK TCB pool.

NUMBER_OF_SUBTASKS
Optional Parameter

is the number of subtasks for concurrent mode.

PRIORITY_MULTIPLIER
Optional Parameter

determines how quickly a task's priority increases as it waits to be dispatched. The faster it increases the less likely a low priority task is to be held up for long periods by higher priority tasks in a busy system.

QR_BATCHING_VALUE
Optional Parameter

is the number of POSTs for BATCH=YES waits in quasi reentrant mode.

SCAN_DELAY_INTERVAL
Optional Parameter

is the delay before terminal control is dispatched after a terminal is posted by the access method.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
MAXOPENTCBS_OUT_OF_RANGE
MAXSSLTCBS_OUT_OF_RANGE
MAXWAIT_LESSTHAN_SCANDELAY
MAXXPTCBS_OUT_OF_RANGE
TOO_LATE_TO_SET_SUBTASKS
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSMT gate, END_BROWSE_MVSTCB function

End a browse operation on the MVS TCBs

Input Parameters

BROWSE_TOKEN
The token that represents the browse session.

Output Parameters

REASON
The values for the parameter are:

```
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSMT gate, GET_NEXT_MVSTCB function

During a browse session, return information about an MVS TCB.

Input Parameters

BROWSE_TOKEN
The token that represents the browse session.

ELEMENT_BUFFER
Optional Parameter

a buffer in which the dispatcher domain returns a list of the addresses of all areas of private storage owned by this TCB.

LENGTH_BUFFER
Optional Parameter

a buffer in which the dispatcher domain returns a list of the lengths of all areas of private storage owned by this TCB.

SUBPOOL_BUFFER
Optional Parameter

a buffer in which the dispatcher domain returns a list of the subpools of all areas of private storage owned by this TCB.

Output Parameters

REASON
The values for the parameter are:

```
BUFFER_NOT_BIG_ENOUGH
END_OF_BROWSE
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TCB_ADDRESS
The address of the MVS TCB.

TCB_NAME
The name of the MVS TCB.

NUMBER_OF_ELEMENTS
Optional Parameter

The number of elements in the three lists of information about the private storage owned by this TCB.

DSMT gate, INQUIRE_MVSTCB function

Return information about an MVS TCB.

Input Parameters

TCB_ADDRESS
The address of the MVS TCB.

ELEMENT_BUFFER
Optional Parameter

a buffer in which the dispatcher domain returns a list of the addresses of all areas of private storage owned by this TCB.

LENGTH_BUFFER
Optional Parameter

a buffer in which the dispatcher domain returns a list of the lengths of all areas of private storage owned by this TCB.

SUBPOOL_BUFFER
Optional Parameter

a buffer in which the dispatcher domain returns a list of the subpools of all areas of private storage owned by this TCB.

Output Parameters

REASON
The values for the parameter are:
```
BUFFER_NOT_BIG_ENOUGH
INVALID_FORMAT
INVALID_FUNCTION
NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TCB_NAME
The name of the MVS TCB.

NUMBER_OF_ELEMENTS
Optional Parameter

The number of elements in the three lists of information about the private storage owned by this TCB.

DSMT gate, SNAPSHOT_MVSTCBS function

Take a snapshot of the state of all MVS TCBs in the CICS address space.

Output Parameters

REASON
The values for the parameter are:
```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TASK_STG_USED
Optional Parameter

indicates if the snapshot was captured in task storage.

Values for the parameter are:
```
TASK_STG_NO
TASK_STG_YES
```

DSMT gate, START_BROWSE_MVSTCB function

Start a browse operation on the MVS TCBs

Output Parameters

REASON
The values for the parameter are:
```
INVALID_FORMAT
INVALID_FUNCTION
```

BROWSE_TOKEN
A token that represents the browse session.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSSR gate, ADD_SUSPEND function

The ADD_SUSPEND function of DSSR gate returns a suspend token which is used to identify a task to be suspended or resumed.

Input Parameters

RESOURCE_NAME
Optional Parameter

is the name of the resource that the task is suspended on.

RESOURCE_TYPE
Optional Parameter

is the type of resource that the task is suspended on.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
INSUFFICIENT_STORAGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SUSPEND_TOKEN
is the token by which the dispatcher recognizes a task to be suspended or resumed.

DSSR gate, DELETE_SUSPEND function

The DELETE_SUSPEND function of DSSR gate discards a suspend token.

Input Parameters

SUSPEND_TOKEN
is the suspend token to be deleted.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_SUSPEND_TOKEN
SUSPEND_TOKEN_IN_USE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSSR gate, RESUME function

The RESUME function of DSSR gate causes a suspended task to be resumed.

Input Parameters

SUSPEND_TOKEN
is the suspend token to be deleted.

COMPLETION_CODE
Optional Parameter

is a completion code to be passed from the resumed task to the suspended task.

ASSOCIATION_TOKEN
Optional parameters

The association token

ASSOCIATION_TYPE

Optional parameters

The association type. either NONE, WAIT or ASSOC

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
TASK_CANCELLED
TIMED_OUT
```

The following values are returned when RESPONSE is INVALID:

```
ALREADY_RESUMED
INVALID_SUSPEND_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see RESPONSE parameter on domain interfaces.

DSSR gate, SUSPEND function

The SUSPEND function of DSSR gate causes a running task to be suspended.

Input Parameters

PURGEABLE

The purgeable status of the task.

Values for the parameter are as follows:

```
NO
YES
```

SUSPEND_TOKEN

The suspend token to be deleted.

DEADLOCK_ACTION

Optional Parameter

Describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged.

DELAY

Optional Parameter

An interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

DISPATCH_BEFORE_WAIT

Optional Parameter

Indicates whether the suspended task is prepared to wait across a partition exit

Values for the parameter are as follows:

```
NO
YES
```

INTERVAL

Optional Parameter

An interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

RESOURCE_NAME

Optional Parameter

The name of the resource that the task is suspended on.

RESOURCE_TYPE

Optional Parameter

The type of resource that the task is suspended on.

RETRY

Optional Parameter

Indicates whether the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation.

Values for the parameter are as follows:

NO
YES

TEMP_HIGH_PRIORITY

Optional Parameter

Indicates whether the task is to get a temporary priority boost at the completion of the suspend.

Values for the parameter are as follows:

NO
YES

TIME_UNIT

Optional Parameter

Identifies the time units specified on the INTERVAL and DELAY parameters where present.

Values for the parameter are as follows:

MILLI_SECOND
SECOND

WLM_WAIT_TYPE

Optional Parameter

Indicates the reason for the wait state of the task to the z/OS Workload Manager.

Values for the parameter are as follows:

CMDRESP
CONV
DISTRIB
IDLE
IO
LOCK
MISC
OTHER_PRODUCT
SESS_LOCALMVS
SESS_NETWORK
SESS_SYSPLEX
TIMER

ASSOCIATION_TOKEN

Optional parameters

The association token

ASSOCIATION_TYPE

Optional parameters

The association type. either NONE, WAIT or ASSOC

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

ALREADY_SUSPENDED
CLEAN_UP_PENDING

INVALID_SUSPEND_TOKEN

The following values are returned when RESPONSE is PURGED:

TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see RESPONSE parameter on domain interfaces.

COMPLETION_CODE
Optional Parameter

is a completion code supplied by the resumed task.

DSSR gate, WAIT_MVS function

The WAIT_MVS function of DSSR gate causes a task to wait on an ECB, or list of ECBs, to be posted via the MVS POST service.

Input Parameters

ECB_ADDRESS
The address of the ECB for the task.

ECB_LIST_ADDRESS
The address of a list of ECBs for the task.

PURGEABLE
The purgeable status of the task.

Values for the parameter are as follows:

NO
YES

BATCH
Optional Parameter

States whether requests are to be batched.

Values for the parameter are as follows:

NO
YES

DEADLOCK_ACTION
Optional Parameter

Describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged.

DELAY
Optional Parameter

An interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

DISPATCH_BEFORE_WAIT
Optional Parameter

Indicates whether the suspended task is prepared to wait across a partition exit

Values for the parameter are as follows:

NO
YES

INTERVAL
Optional Parameter

An interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

RESOURCE_NAME
Optional Parameter

The name of the resource that the task is suspended on.

RESOURCE_TYPE
Optional Parameter

The type of resource that the task is suspended on.

RETRY
Optional Parameter

Indicates whether the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation.

Values for the parameter are as follows:
```
NO
YES
```

TEMP_HIGH_PRIORITY
Optional Parameter

Indicates whether the task is to get a temporary priority boost at the completion of the suspend.

Values for the parameter are as follows:
```
NO
YES
```

TIME_UNIT
Optional Parameter

Identifies the time units specified on the INTERVAL and DELAY parameters where present.

Values for the parameter are as follows:
```
MILLI_SECOND
SECOND
```

WLM_WAIT_TYPE
Optional Parameter

Indicates the reason for wait state of the task to the z/OS Workload Manager (WLM).

Values for the parameter are as follows:
```
CMDRESP
CONV
DISTRIB
IDLE
IO
LOCK
MISC
OTHER_PRODUCT
SESS_LOCALMVS
SESS_NETWORK
SESS_SYSPLEX
TIMER
```

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
```
ALREADY_WAITING
INVALID_ECB_ADDR
```

The following values are returned when RESPONSE is PURGED:
```
TASK_CANCELLED
TIMED_OUT
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSSR gate, WAIT_OLDC function

The WAIT_OLDC function of DSSR gate causes a task to wait on an ECB that must be posted by setting the X'40' bit rather than via the MVS POST service. This function is supported only in QR mode.

Input Parameters

ECB_ADDRESS
The address of the ECB for the task.

PURGEABLE
The purgeable status of the task.

Values for the parameter are as follows:
NO
YES

DEADLOCK_ACTION
Optional Parameter

Describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged.

DELAY
Optional Parameter

An interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

DISPATCH_BEFORE_WAIT
Optional Parameter

Indicates whether the suspended task is prepared to wait across a partition exit

Values for the parameter are as follows:
NO
YES

INTERVAL
Optional Parameter

An interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

RESOURCE_NAME
Optional Parameter

The name of the resource that the task is suspended on.

RESOURCE_TYPE
Optional Parameter

The type of resource that the task is suspended on.

RETRY
Optional Parameter

Indicates whether the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation.

Values for the parameter are as follows:
NO
YES

TEMP_HIGH_PRIORITY
Optional Parameter

Indicates whether the task is to get a temporary priority boost at the completion of the suspend.

Values for the parameter are as follows:

NO
YES

TIME_UNIT

Optional Parameter

Identifies the time units specified on the INTERVAL and DELAY parameters where present.

Values for the parameter are as follows:

MILLI_SECOND
SECOND

WLM_WAIT_TYPE

Optional Parameter

Indicates the reason for the wait state of the task to the z/OS Workload Manager (WLM).

Values for the parameter are as follows:

CMDRESP
CONV
DISTRIB
IDLE
IO
LOCK
MISC
OTHER_PRODUCT
SESS_LOCALMVS
SESS_NETWORK
SESS_SYSPLEX
TIMER

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

ALREADY_WAITING
INVALID_ECB_ADDR
INVALID_MODE

The following values are returned when RESPONSE is PURGED:

TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSSR gate, WAIT_OLDW function

The WAIT_OLDW function of DSSR gate causes a task to wait on an ECB, or list of ECBs, that can be posted via the MVS POST service or by setting the POST bit (X'40' in the first byte). This function is supported only in QR mode.

Input Parameters

ECB_ADDRESS

The address of the ECB for the task.

ECB_LIST_ADDRESS

The address of a list of ECBs for the task.

PURGEABLE
The purgeable status of the task.

Values for the parameter are as follows:
NO
YES

DEADLOCK_ACTION
Optional Parameter

Describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged.

DELAY
Optional Parameter

An interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

DISPATCH_BEFORE_WAIT
Optional Parameter

Indicates whether the suspended task is prepared to wait across a partition exit.

Values for the parameter are as follows:
NO
YES

INTERVAL
Optional Parameter

An interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

RESOURCE_NAME
Optional Parameter

The name of the resource that the task is suspended on.

RESOURCE_TYPE
Optional Parameter

The type of resource that the task is suspended on.

RETRY
Optional Parameter

Indicates whether the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation.

Values for the parameter are as follows:
NO
YES

SPECIAL_TYPE
Optional Parameter

Identifies the special task CSTP.

Values for the parameter are as follows:
CSTP

TEMP_HIGH_PRIORITY
Optional Parameter

Indicates whether the task is to get a temporary priority boost at the completion of the suspend.

Values for the parameter are as follows:
NO
YES

TIME_UNIT
Optional Parameter

Identifies the time units specified on the INTERVAL and DELAY parameters where present.

Values for the parameter are as follows:
MILLI_SECOND
SECOND

WLM_WAIT_TYPE
Optional Parameter

Indicates the reason for the wait state of the task to the z/OS Workload Manager.

Values for the parameter are as follows:
CMDRESP
CONV
DISTRIB
IDLE
IO
LOCK
MISC
OTHER_PRODUCT
SESS_LOCALMVS
SESS_NETWORK
SESS_SYSPLEX
TIMER

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
ALREADY_WAITING
INVALID_ECB_ADDR
INVALID_MODE

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Dispatcher domain's generic gates

Table 18 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 18. Dispatcher domain's generic gates

Gate	Trace	Functions	Format
APUE	DS 0121 DS 0122	SET_EXIT_STATUS	APUE
DMDM	DS 0006 DS 0007	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

Table 18. Dispatcher domain's generic gates (continued)

Gate	Trace	Functions	Format
KEDS	DS 0012 DS 0013	TCB_REPLY TASK_REPLY	KEDS
SMNT	DS 0145 DS 0113	STORAGE_NOTIFY	SMNT
STST	DS 0020 DS 0021	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

Dispatcher domain's generic formats

Table 19 describes the generic formats owned by the domain and shows the functions performed on the calls.

Table 19. Dispatcher domain's generic formats

Format	Calling modules	Functions
DSAT	DFHDSKE	TASK_REPLY
	DFHDSDS4	PURGE_INHIBIT_QUERY
	DFHSJIN	FORCE_PURGE_INHIBIT_QUERY
	DFHSMVN	NOTIFY_DELETE_TCB

Note: In the descriptions of the formats, the input parameters are input not to the dispatcher domain, but to the domain being called by the dispatcher domain. Similarly, the output parameters are output by the domain that was called by the dispatcher domain, in response to the call.

DSAT gate, TASK_REPLY function

The TASK_REPLY function of DSAT format is used to notify the domain that attached a task that the task has had its first dispatch.

Input Parameters

SUSPEND_TOKEN
is the suspend token that the task can be suspended against by default.

TASK_TOKEN
is the token by which the task that has been dispatched is known to the dispatcher.

TRANSACTION_BUILD
Optional Parameter

Values for the parameter are:

```
BUILD
TERMINATE
```

USER_TOKEN
is the token by which the task that has been dispatched is known to the called domain.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSAT gate, PURGE_INHIBIT_QUERY function

The PURGE_INHIBIT_QUERY function of DSAT format is used by the dispatcher to see if a task selected for purge can be purged. Its main purpose is to find out from the AP domain whether the task is currently purgeable by the system.

Input Parameters

TASK_TOKEN
is the token by which the task that has been dispatched is known to the dispatcher.

USER_TOKEN
is the token by which the task that has been dispatched is known to the called domain.

Output Parameters

PURGE_INHIBITED_RESPONSE
states whether the task can be purged.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSAT gate, FORCE_PURGE_INHIBIT_QUERY function

The FORCE_PURGE_INHIBIT_QUERY function of DSAT format is used by the dispatcher to see if a task selected for purge can be force purged. Its main purpose is to find out from the AP domain whether the task is currently purgeable by the system.

Input Parameters

TASK_TOKEN
is the token by which the task that has been dispatched is known to the dispatcher.

USER_TOKEN
is the token by which the task that has been dispatched is known to the called domain.

Output Parameters

PURGE_INHIBITED_RESPONSE
states whether the task can be purged.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSAT gate, NOTIFY_DELETE_TCB function

The NOTIFY_DELETE function of DSAT format notifies the interested domain (as specified in the NOTIFY_DELETE parameter on the DSIT ACTIVATE_MODE request for the mode) that a DELETE_TCB request is in progress.

Input Parameters

TCB_TOKEN

The DS token representing the TCB instance for which notification is required when deleted.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHDSAT	Handles the following requests: ATTACH CHANGE_MODE CHANGE_PRIORITY SET_PRIORITY CANCEL_TASK
DFHDSBR	Handles the following requests: START_BROWSE GET_NEXT END_BROWSE INQUIRE_TASK
DFHDSDM	Handles the following requests: DMDM PRE_INITIALISE DMDM INITIALISE_DOMAIN DMDM QUIESCE_DOMAIN DMDM TERMINATE_DOMAIN
DFHDSIT	Handles the following requests: INQUIRE_DISPATCHER SET_DISPATCHER
DFHDSKE	Handles kernel DS requirements, and handles the following requests: KEDS TCB_REPLY KEDS TASK_REPLY
DFHDSSM	Receives the STORAGE_NOTIFY call from the storage manager domain.
DFHDSSR	Handles the following requests: ADD_SUSPEND DELETE_SUSPEND INQUIRE_SUSPEND_TOKEN SUSPEND RESUME WAIT_MVS WAIT_OLDW WAIT_OLDC
DFHDSST	Receives statistics calls from the ST domain
DFHDSUE	Receives the user exit gate call from the AP domain

Exits

There are two global user exit points in the dispatcher domain, XDSAWT and XDSBWT.

Dump Domain (DU)

The dump domain is responsible for producing storage dumps and for handling the associated data sets and status.

Dump Domain's specific gates

The specific gates provide access for other domains to functions that are provided by the DU domain.

DUDT gate, ADD_SYSTEM_DUMPCODE function

The ADD_SYSTEM_DUMPCODE function of the DUDT gate is invoked to add a new dump code to the system dump table.

Input Parameters

DAEOPTION
states whether a dump produced for this dumpcode is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component.

Values for the parameter are:
NO
YES

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued.

Values for the parameter are:
LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems
RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

MAXIMUM_DUMPS
is the maximum number of times the dump code action can be taken.

SYSTEM_DUMP
states whether a system dump is required for this dump code.

Values for the parameter are:
NO
YES

SYSTEM_DUMPCODE
is the system dump code.

TERMINATE_CICS
states whether CICS is to be terminated for this dump code.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CATALOG_FULL
DUPLICATE_DUMPCODE

INSUFFICIENT_STORAGE
INVALID_DUMPCODE
IO_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDT gate, ADD_TRAN_DUMPCODE function

The ADD_TRAN_DUMPCODE function of the DUDT gate is invoked to add a new dump code to the transaction dump table.

Input Parameters

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued.

Values for the parameter are:

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

MAXIMUM_DUMPS
is the maximum number of times the dump code action can be taken.

SYSTEM_DUMP
states whether a system dump is required for this dump code.

Values for the parameter are:
NO
YES

TERMINATE_CICS
states whether CICS is to be terminated for this dump code.

Values for the parameter are:
NO
YES

TRANSACTION_DUMP
states whether a transaction dump is required for this dump code.

Values for the parameter are:
NO
YES

TRANSACTION_DUMPCODE
is the transaction dump code.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CATALOG_FULL
DUPLICATE_DUMPCODE
INSUFFICIENT_STORAGE
INVALID_DUMPCODE
IO_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDT gate, DELETE_SYSTEM_DUMPCODE function

The DELETE_SYSTEM_DUMPCODE function of the DUDT gate is invoked to delete an existing dump code from the system dump table.

Input Parameters

SYSTEM_DUMPCODE
is the system dump code.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DUMPCODE_NOT_FOUND
IO_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDT gate, DELETE_TRAN_DUMPCODE function

The DELETE_TRAN_DUMPCODE function of the DUDT gate is invoked to delete an existing dump code from the transaction dump table.

Input Parameters

TRANSACTION_DUMPCODE
is the transaction dump code.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DUMPCODE_NOT_FOUND
IO_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDT gate, ENDBR_SYSTEM_DUMPCODE function

The ENDBR_SYSTEM_DUMPCODE function of the DUDT gate is invoked to end a browse on the system dump table.

Input Parameters

BROWSE_TOKEN
is the token identifying the browse session.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDT gate, ENDBR_TRAN_DUMPCODE function

The ENDBR_TRAN_DUMPCODE function of the DUDT gate is invoked to end a browse session on the transaction dump table.

Input Parameters

BROWSE_TOKEN
is the token identifying the browse session.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDT gate, GETNEXT_SYSTEM_DUMPCODE function

The GETNEXT_SYSTEM_DUMPCODE function of the DUDT gate is invoked in a browse session to get the next entry in the system dump table.

Input Parameters

BROWSE_TOKEN
is the token identifying the browse session.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
END_BROWSE

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

COUNT
Optional Parameter

is the number of times the dump code action has been taken.

DAEOPTION
Optional Parameter

states whether a dump produced for this dumpcode is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component.

Values for the parameter are:
NO
YES

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued.

Values for the parameter are:

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request

is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

MAXIMUM_DUMPS
Optional Parameter

is the maximum number of times the dump code action can be taken.

SYSTEM_DUMP
Optional Parameter

states whether a system dump is required for this dump code.

Values for the parameter are:
NO
YES

SYSTEM_DUMPCODE
Optional Parameter

is the system dump code.

TERMINATE_CICS
Optional Parameter

states whether CICS is to be terminated for this dump code.

Values for the parameter are:
NO
YES

DUDT gate, GETNEXT_TRAN_DUMPCODE function

The GETNEXT_TRAN_DUMPCODE function of the DUDT gate is invoked in a browse session to get the next entry in the transaction dump table.

Input Parameters

BROWSE_TOKEN
is the token identifying the browse session.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
END_BROWSE

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

COUNT
Optional Parameter

is the number of times the dump code action has been taken.

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued.

Values for the parameter are:

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request

is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

MAXIMUM_DUMPS
Optional Parameter

is the maximum number of times the dump code action can be taken.

SYSTEM_DUMP
Optional Parameter

states whether a system dump is required for this dump code.

Values for the parameter are:
NO
YES

TERMINATE_CICS
Optional Parameter

states whether CICS is to be terminated for this dump code.

Values for the parameter are:
NO
YES

TRANSACTION_DUMP
Optional Parameter

states whether a transaction dump is required for this dump code.

Values for the parameter are:
NO
YES

TRANSACTION_DUMPCODE
Optional Parameter

is the transaction dump code.

DUDT gate, INQUIRE_SYSTEM_DUMPCODE function

The INQUIRE_SYSTEM_DUMPCODE function of the DUDT gate is invoked to inquire on a dump code in the system dump table.

Input Parameters

SYSTEM_DUMPCODE
is the system dump code.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DUMPCODE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

COUNT
Optional Parameter

is the number of times the dump code action has been taken.

DAEOPTION
Optional Parameter

states whether a dump produced for this dumpcode is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component.

Values for the parameter are:

NO
YES

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued.

Values for the parameter are:

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

MAXIMUM_DUMPS
Optional Parameter

is the maximum number of times the dump code action can be taken.

SYSTEM_DUMP
Optional Parameter

states whether a system dump is required for this dump code.

Values for the parameter are:
NO
YES

TERMINATE_CICS
Optional Parameter

states whether CICS is to be terminated for this dump code.

Values for the parameter are:
NO
YES

DUDT gate, INQUIRE_TRAN_DUMPCODE function

The INQUIRE_TRAN_DUMPCODE function of the DUDT gate is invoked to inquire on a dump code in the transaction dump table.

Input Parameters

TRANSACTION_DUMPCODE
is the transaction dump code.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DUMPCODE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

COUNT
Optional Parameter

is the number of times the dump code action has been taken.

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued.

Values for the parameter are:

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

MAXIMUM_DUMPS
Optional Parameter

is the maximum number of times the dump code action can be taken.

SYSTEM_DUMP
Optional Parameter

states whether a system dump is required for this dump code.

Values for the parameter are:
NO
YES

TERMINATE_CICS
Optional Parameter

states whether CICS is to be terminated for this dump code.

Values for the parameter are:
NO
YES

TRANSACTION_DUMP
Optional Parameter

states whether a transaction dump is required for this dump code.

Values for the parameter are:
NO
YES

DUDT gate, SET_SYSTEM_DUMPCODE function

The SET_SYSTEM_DUMPCODE function of the DUDT gate is invoked to set options for a dump code in the system dump table.

Input Parameters

SYSTEM_DUMPCODE
is the system dump code.

DAEOPTION
Optional Parameter

states whether a dump produced for this dumpcode is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component.

Values for the parameter are:
NO
YES

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued.

Values for the parameter are:

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

MAXIMUM_DUMPS
Optional Parameter

is the maximum number of times the dump code action can be taken.

RESET_COUNT
Optional Parameter

states whether COUNT is to be reset to zero.

Values for the parameter are:
NO
YES

SYSTEM_DUMP
Optional Parameter

states whether a system dump is required for this dump code.

Values for the parameter are:
NO
YES

TERMINATE_CICS
Optional Parameter

states whether CICS is to be terminated for this dump code.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CATALOG_FULL
DUMPCODE_NOT_FOUND
IO_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDT gate, SET_TRAN_DUMPCODE function

The SET_TRAN_DUMPCODE function of the DUDT gate is invoked to set options for a dump code in the transaction dump table.

Input Parameters

TRANSACTION_DUMPCODE
is the transaction dump code.

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued.

Values for the parameter are:

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

MAXIMUM_DUMPS
Optional Parameter

is the maximum number of times the dump code action can be taken.

RESET_COUNT
Optional Parameter

states whether COUNT is to be reset to zero.

Values for the parameter are:
NO
YES

SYSTEM_DUMP
Optional Parameter

states whether a system dump is required for this dump code.

Values for the parameter are:
NO
YES

TERMINATE_CICS
Optional Parameter

states whether CICS is to be terminated for this dump code.

Values for the parameter are:
NO
YES

TRANSACTION_DUMP
Optional Parameter

states whether a transaction dump is required for this dump code.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CATALOG_FULL
DUMPCODE_NOT_FOUND
IO_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDT gate, STARTBR_SYSTEM_DUMPCODE function

The STARTBR_SYSTEM_DUMPCODE function of the DUDT gate is invoked to start a browse session on the system dump table.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE

BROWSE_TOKEN
is the token identifying the browse session.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDT gate, STARTBR_TRAN_DUMPCODE function

The STARTBR_TRAN_DUMPCODE function of the DUDT gate is invoked to start a browse session on the transaction dump table.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE

BROWSE_TOKEN
is the token identifying the browse session.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDU gate, SYSTEM_DUMP function

The SYSTEM_DUMP function of the DUDU gate is invoked to take a system dump.

Input Parameters

SYSTEM_DUMPCODE
is the system dump code.

CALLER
Optional Parameter

specifies the address and length of a character string to appear as the caller of this dump.

INDIRECT_CALL
Optional Parameter

states whether the call is indirect, that is, whether the actual requester of the dump is not the immediate caller of the dump domain.

Values for the parameter are:
NO
YES

MESSAGE_TEXT
Optional Parameter

specifies the address and length of the message text associated with this system dump.

SYMPTOM_RECORD
Optional Parameter

specifies the address and length of the symptom record associated with this dump.

SYMPTOM_STRING
Optional Parameter

specifies the address and length of the symptom string associated with this dump.

TERMINATE_CICS
Optional Parameter

states whether CICS is to be terminated for this dump code.

Values for the parameter are:
NO
YES

TITLE
Optional Parameter

specifies the address and length of a title to be associated with this dump.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
FESTAE_FAILED
INSUFFICIENT_STORAGE
INVALID_DUMPCODE
IWMWQWRK_FAILED
NO_DATASET
PARTIAL_SYSTEM_DUMP
SDUMP_BUSY
SDUMP_FAILED
SDUMP_NOT_AUTHORIZED
SUPPRESSED_BY_DUMPOPTION
SUPPRESSED_BY_DUMPTABLE
SUPPRESSED_BY_USEREXIT

The following values are returned when RESPONSE is INVALID:
INVALID_PROBDESC
INVALID_SVC_CALL

DUMPID
is a character string of the form "rrrr/cccc" giving a unique identification to this dump request. "rrrr" is the run number of this CICS instance. Leading zeros are removed. The run number is incremented every time CICS is initialized. "cccc" is the count of this dump request within this CICS run.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUDU gate, TRANSACTION_DUMP function

The TRANSACTION_DUMP function of the DUDU gate is invoked to take a transaction dump.

Input Parameters

TRANSACTION_DUMPCODE
is the transaction dump code.

CSA
Optional Parameter

- common system area

Values for the parameter are:
NO
YES

DCT
Optional Parameter

- destination control table (for transient data queue resource definitions)

Values for the parameter are:
NO
YES

FCT
Optional Parameter

- file control table (for file resource definitions)

Values for the parameter are:
NO
YES

INDIRECT_CALL
Optional Parameter

states whether the call is indirect, that is, whether the actual requester of the dump is not the immediate caller of the dump domain.

Values for the parameter are:
NO
YES

PCT
Optional Parameter

- program control table (for transaction resource definitions)

Values for the parameter are:
NO
YES

PPT
Optional Parameter

- processing program table (for program resource definitions)

Values for the parameter are:
NO
YES

PROGRAM
Optional Parameter

- program storage

Values for the parameter are:
NO
YES

SEGMENT
Optional Parameter

specifies the address and length of a single block of storage to be dumped.

SEGMENT_LIST
Optional Parameter

specifies the address and length of a list of length-address pairs of storage blocks to be dumped. SEGMENT and SEGMENT_LIST may not be specified together.

SIT
Optional Parameter

- system initialization table

Values for the parameter are:
NO
YES

TCA
Optional Parameter

- task control area

Values for the parameter are:
NO
YES

TCT
Optional Parameter

- terminal control table (for terminal resource definitions)

Values for the parameter are:
NO
YES

TERMINAL
Optional Parameter

- terminal-related storage areas

Values for the parameter are:
NO
YES

TRANSACTION
Optional Parameter

- transaction-related storage areas

Values for the parameter are:
NO
YES

TRT
Optional Parameter

- internal trace table

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
FESTAE_FAILED
INSUFFICIENT_STORAGE
INVALID_DUMPCODE
IWMWQWRK_FAILED
NOT_OPEN
OPEN_ERROR
PARTIAL_SYSTEM_DUMP
PARTIAL_TRANSACTION_DUMP
SDUMP_BUSY
SDUMP_FAILED
SDUMP_NOT_AUTHORIZED
SUPPRESSED_BY_DUMPOPTION
SUPPRESSED_BY_DUMPTABLE
SUPPRESSED_BY_USEREXIT

The following values are returned when RESPONSE is INVALID:

INVALID_PROBDESC
INVALID_SVC_CALL

DUMPID
is a character string of the form "rrrr/cccc" giving a unique identification to this dump request. "rrrr" is the run number of this CICS instance. Leading zeros are removed. The run number is incremented every time CICS is initialized. "cccc" is the count of this dump request within this CICS run.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUFT gate, DEREGISTER function

Deregister a feature with the dump domain

Input Parameters

COMPANY_NAME
The name of the company providing the feature.

FEATURE_LEVEL
The level number of the feature.

FEATURE_NAME
The name of the feature.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

FEATURE_NOT_FOUND

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUFT gate, INQUIRE_FEATURE function

Inquire about a feature that is registered with the dump domain.

Input Parameters

COMPANY_NAME
The name of the company providing the feature.

FEATURE_LEVEL
The level number of the feature.

FEATURE_NAME
The name of the feature.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

DEREGISTERED_FEATURE
FEATURE_NOT_FOUND

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUMP_FORMATTING_ROUTINE
Optional Parameter

The dump formatting routine provided by the feature.
FEATURE_TOKEN
Optional Parameter

The token that identifies the registered feature.
FEATURE_TRACE_TOKEN
The token that the feature uses to identify itself to the CICS trace domain.
TRACE_ABBREVIATED_NAME
Optional Parameter

The abbreviated name that the feature uses in the trace.
TRACE_FORMATTING_ROUTINE
Optional Parameter

The trace formatting routine provided by the feature.

DUFT gate, REGISTER function

Register a feature with the dump domain.

Input Parameters

COMPANY_NAME
The name of the company providing the feature.
FEATURE_LEVEL
The level number of the feature.
FEATURE_NAME
The name of the feature.
DUMP_FORMATTING_ROUTINE
Optional Parameter

The dump formatting routine provided by the feature.
FEATURE_TOKEN
Optional Parameter

The token that identifies the registered feature.
TRACE_ABBREVIATED_NAME
Optional Parameter

The abbreviated name that the feature uses in the trace.
TRACE_FORMATTING_ROUTINE
Optional Parameter

The trace formatting routine provided by the feature.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
DUPLICATE_DUMP_ROUTINE
DUPLICATE_FEATURE
DUPLICATE_TRACE_ROUTINE
INSUFFICIENT_STORAGE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

FEATURE_TRACE_TOKEN
The token that the feature uses to identify itself to the CICS trace domain.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUFT gate, UPDATE_FEATURE function

Update information about a feature that is registered with the dump domain.

Input Parameters

COMPANY_NAME
The name of the company providing the feature.

FEATURE_LEVEL
The level number of the feature.

FEATURE_NAME
The name of the feature.

DUMP_FORMATTING_ROUTINE
Optional Parameter

The dump formatting routine provided by the feature.

FEATURE_TOKEN
Optional Parameter

The token that identifies the registered feature.

TRACE_ABBREVIATED_NAME
Optional Parameter

The abbreviated name that the feature uses in the trace.

TRACE_FORMATTING_ROUTINE
Optional Parameter

The trace formatting routine provided by the feature.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
DEREGISTERED_FEATURE
DUPLICATE_DUMP_ROUTINE
DUPLICATE_TRACE_ROUTINE
FEATURE_NOT_FOUND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, CROSS_SYSTEM_DUMP_AVAIL function

The CROSS_SYSTEM_DUMP_AVAIL function of the DUSR gate is used to inform the dump domain about the DUMP_AVAIL token which links CICS with the z/OS Workload Manager.

Output Parameters

REASON
The values for the parameter are:

```
NOT_OPEN
OPEN_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, DUMPDS_CLOSE function

The DUMPDS_CLOSE function of the DUSR gate is invoked to close the CICS dump data set.

Output Parameters

REASON

The values for the parameter are:

NOT_OPEN

OPEN_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, DUMPDS_OPEN function

The DUMPDS_OPEN function of the DUSR gate is invoked to open the CICS dump data set.

Output Parameters

REASON

The values for the parameter are:

OPEN_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, DUMPDS_SWITCH function

The DUMPDS_SWITCH function of the DUSR gate is invoked to switch to the alternate CICS dump data set.

Output Parameters

REASON

The values for the parameter are:

OPEN_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, INQUIRE_CURRENT_DUMPDS function

The INQUIRE_CURRENT_DUMPDS function of the DUSR gate returns the name of the current dump data set.

Output Parameters

REASON

The values for the parameter are:

NOT_OPEN

OPEN_ERROR

CURRENT_DUMPDS

is the name of the current dump data set.

Values for the parameter are:

DFHDMPA

DFHDMPB

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, INQUIRE_DUMPDS_AUTOSWITCH function

The INQUIRE_DUMPDS_AUTOSWITCH function of the DUSR gate returns an indication of whether autoswitching is active or not.

Output Parameters

REASON
The values for the parameter are:
NOT_OPEN
OPEN_ERROR

AUTOSWITCH
is the dump data set autoswitch status.

Values for the parameter are:
OFF
ON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, INQUIRE_DUMPDS_OPEN_STATUS function

The INQUIRE_DUMPDS_OPEN_STATUS function of the DUSR gate returns an indication of whether the current dump data set is open or closed.

Output Parameters

REASON
The values for the parameter are:
NOT_OPEN
OPEN_ERROR

OPEN_STATUS
is the open status of the current dump data set.

Values for the parameter are:
CLOSED
OPEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, INQUIRE_INITIAL_DUMPDS function

The INQUIRE_INITIAL_DUMPDS function of the DUSR gate returns the setting of the initial dump data set.

Output Parameters

REASON
The values for the parameter are:
NOT_OPEN
OPEN_ERROR

INITIAL_DUMPDS
is the initial dump data set.

Values for the parameter are:
AUTO
DFHDMPA
DFHDMPB

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, INQUIRE_RETRY_TIME function

The INQUIRE_RETRY_TIME function of the DUSR gate returns the value of the SDUMP retry time.

Output Parameters

REASON

The values for the parameter are:

NOT_OPEN
OPEN_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RETRY_TIME

is the value in seconds of the time interval for which CICS should retry SDUMP requests that fail because another SDUMP is in progress within the MVS system. The SDUMP is retried at intervals of five seconds for the specified total time.

DUSR gate, INQUIRE_SYSTEM_DUMP function

The INQUIRE_SYSTEM_DUMP function of the DUSR gate returns the setting of the system dump suppression flag.

Output Parameters

REASON

The values for the parameter are:

NOT_OPEN
OPEN_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SYSTEM_DUMP

states whether a system dump is required for this dump code.

Values for the parameter are:

NO
YES

DUSR gate, SET_DUMPDS_AUTOSWITCH function

The SET_DUMPDS_AUTOSWITCH function of the DUSR gate is used to set autoswitching on or off.

Input Parameters

AUTOSWITCH

is the dump data set autoswitch status.

Values for the parameter are:

OFF
ON

Output Parameters

REASON

The values for the parameter are:

NOT_OPEN
OPEN_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, SET_DUMPTABLE_DEFAULTS function

The SET_DUMPTABLE_DEFAULTS function of the DUSR gate is invoked during system initialization tp update the DUA with the DAE option specified in a SIT or as a SIT override.

Input Parameters

DAE_DEFAULT

Optional Parameter

indicates whether temporary dump table entries added by CICS will indicate DAE (dump eligible for DAE suppression) or NODAE (dump will not be suppressed by DAE).

Values for the parameter are:

```
DAE
NODAE
```

SYDUMAX_DEFAULT

Optional Parameter

is taken from system initialization parameter (SIT=SYDUMAX), which specifies the maximum number of system dumps which can be taken per dump table entry.

TRDUMAX_DEFAULT

Optional Parameter

is taken from system initialization parameter (SIT=TRDUMAX), which specifies the maximum number of transaction dumps which can be taken per dump table entry.

Output Parameters

REASON

The values for the parameter are:

```
NOT_OPEN
OPEN_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, SET_INITIAL_DUMPDS function

The SET_INITIAL_DUMPDS function of the DUSR gate is used to change the setting of the initial dump data set.

Input Parameters

INITIAL_DUMPDS

is the initial dump data set.

Values for the parameter are:

```
AUTO
DFHDMPA
DFHDMPB
```

Output Parameters

REASON

The values for the parameter are:

```
NOT_OPEN
OPEN_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, SET_RETRY_TIME function

The SET_RETRY_TIME function of the DUSR gate is invoked to set the SDUMP retry time.

Input Parameters

RETRY_TIME
is the value in seconds of the time interval for which CICS should retry SDUMP requests that fail because another SDUMP is in progress within the MVS system. The SDUMP is retried at intervals of five seconds for the specified total time.

Output Parameters

REASON
The values for the parameter are:
NOT_OPEN
OPEN_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, SET_SYSTEM_DUMP function

The SET_SYSTEM_DUMP function of the DUSR gate is used to change the setting of the system dump suppression flag.

Input Parameters

SYSTEM_DUMP
states whether a system dump is required for this dump code.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
NOT_OPEN
OPEN_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, SET_TRANTABLESIZE function

Set the size of the transaction dump trace table.

Input Parameters

TRAN_TABLE_SIZE
the intended size of the transaction dump trace table.

Output Parameters

REASON
The values for the parameter are:
NOT_OPEN
OPEN_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUSR gate, SET_TRANTABLETYPE function

Specify which trace entries should be copied from the internal trace table to the transaction dump trace table.

Input Parameters

TRAN_TABLE_TYPE
indicates which trace entries should be copied.

Values for the parameter are:
```
ALL
TRAN
```

Output Parameters

REASON
The values for the parameter are:
```
NOT_OPEN
OPEN_ERROR
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Dump domain's generic gates

Table 20 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 20. Dump domain's generic gates

Gate	Trace	Functions	Format
APUE	DU 0301 DS 0302	SET_EXIT_STATUS	APUE
DMDM	DU 0001 DU 0002	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	DS 0500 DS 0501	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

Initialization and termination

In preinitialization processing, the dump domain establishes the initial dumping status:

- System dumping is enabled or suppressed, as required.
- The next transaction dump data set to be used is flagged.
- The transaction dump data set autoswitch status is set on or off, as required.
- The dump retry interval is established.
- The system dump table is initialized to empty.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

In initialization processing, the dump domain loads the transaction dump table and the system dump table from the global catalog. In quiesce processing, the dump domain performs only internal routines.

In termination processing, the dump domain closes the transaction dump data set.

DMDM PRE_INITIALIZE function

The PRE_INITIALIZE function of the DMDM gate performs the following functions:

1. Issue MVS(TM) GETMAIN for DU anchor block (DUA) and initialize it.
2. Read DU state record from the local catalog and set values in the DUA.
3. Initialize to empty the system dump table.
4. Issue MVS GETMAIN for DU statistics buffer.
5. Acquire startup information from the parameter manager (PA) domain and set it in the DUA.
6. Inform the kernel that DU system dump is available by issuing KEDD ADD_GATE for the DFHDUDU gate.

DMDM INITIALIZE_DOMAIN function

The INITIALIZE_DOMAIN function of the DMDM gate performs the following functions:

1. Load the system dump table from the global catalog.
2. Load the transaction dump table from the global catalog.
3. Issue LMLM ADD_LOCK for the dump data set lock (DUDATSET).
4. Issue LMLM ADD_LOCK for the dump table lock (DUTABLE).
5. Issue LMLM UNLOCK for DUTABLE lock.
6. Issue KEDD ADD_GATE for the DU STST, DUDT, and APUE gates.
7. Initialize transaction dump, including loading DFHDUIO, and indicate that the dump table is available to the DUDU TRANSACTION_DUMP function.
8. Update DU state record on catalog.
9. Issue LMLM UNLOCK for DUDATSET lock, thereby making the transaction dump function available.

DMDM QUIESCE_DOMAIN function

The QUIESCE_DOMAIN function of the DMDM gate issues a DMDM WAIT_PHASE function request to ensure all statistics are collected.

DMDM TERMINATE_DOMAIN function

The TERMINATE_DOMAIN function of the DMDM gate issues a DUSU CLOSE request to close the transaction dump data set.

APUE SET_EXIT_STATUS function

The SET_EXIT_STATUS function of the APUE gate sets the exit status flag in the DUA for the specified exit.

STST COLLECT_STATISTICS function

The COLLECT_STATISTICS function of the STST gate is called from the statistics domain. The process flow is:

1. Issue LMLM LOCK for DUTABLE lock on the transaction dump table.
2. Acquire KE system dump lock.
3. Issue STST COLLECT_STATISTICS call to DFHDUTM.

   If the COLLECT_STATISTICS parameters requested DATA, the following statistics records are written to the statistics domain:

   a. If the RESOURCE_TYPE is not specified or is SYSDUMP, a DFHSDGPS global system dump statistics record is created, using global system dump counts (taken and suppressed) from the DUA. The KE system lock is released while a STATS_PUT request is made to the statistics domain. The lock is obtained again on successful completion of the STATS_PUT.
   b. If the RESOURCE_TYPE is not specified or is TRANDUMP, a DFHTDGPS global transaction dump statistics record is created, using global transaction dump counts (taken and suppressed) from the DUA. The DUTABLE lock is released while a RECORD_STATISTICS request is made to the statistics domain. The lock is obtained again on successful completion of the RECORD_STATISTICS.
   c. If the RESOURCE_TYPE is not specified or is SYSDUMP, a DFHSDRPS statistics detail record is written for every dump code found on the system dump table. The records contain the statistics for that dump code held on the dump table entry. The DFHSDRPS records are buffered and full buffers are written out using a RECORD_STATISTICS call to the statistics domain.
   d. If the RESOURCE_TYPE is not specified or is TRANDUMP, a DFHTDRPS statistics detail record is written for every dump code found on the transaction dump table. The records contain the statistics for that dump code held on the dump table entry. The DFHTDRPS records are buffered and full buffers are written out using a RECORD_STATISTICS call to the statistics domain.

   The global system and transaction dump counts (taken and suppressed) in the DUA are also reset to zero. The last_reset_time is also updated in the DUA at this time.
4. Release DUTABLE lock and system dump lock.

STST COLLECT_RESOURCE_STATS function

The COLLECT_RESOURCE_STATS function of the STST gate is called from an **EXEC CICS** command. The process flow is:

1. Issue LMLM LOCK for DUTABLE lock on the transaction dump table.
2. Acquire KE system dump lock.
3. Issue STST COLLECT_RESOURCE_STATS call to DFHDUTM.
   a. Validate RESOURCE_TYPE for either SYSDUMP or TRANDUMP. Perform error processing and return INVALID to the caller if it is neither of these.
   b. If the RESOURCE_ID has not been passed, format a global statistics record, using counts of dumps taken and suppressed from the DUA, for either system or transaction dumps, depending on the RESOURCE_TYPE. Return this record to the caller in the RESOURCE_STATISTICS_DATA parameter.
   c. If the RESOURCE_ID is present, it should contain a dump code. Search the relevant dump table (depending on RESOURCE_TYPE). Return ID_NOT_FOUND exception to the caller if the dump code cannot be found.

If the dump code is found, format either a DFHTDRPS or a DFHSDRPS statistics record using the dumps taken and suppressed statistics on the dump table entry. This record is formatted in the next available space in the RESOURCE_STATISTICS_DATA buffer.

4. Release DUTABLE lock and system dump lock.

Modules

Module	Function
DFHDUDM	Processes requests to the DMDM gate of the dump domain
DFHDUDT	Processes requests to the DUDT gate of the dump domain
DFHDUDU	Processes requests to the DUDU gate of the dump domain
DFHDUIO	Processes domain subroutine requests of format DUIO
DFHDUPH	Writes line to dump index for each dump header record encountered. On first entry, opens the index file DFHTINDX.
DFHDUPM	Invoked for each module index entry found to save information. Invoked when dump trailer record found to format and print the complete module index.
DFHDUPP	Is responsible for all access to the print file DFHPRINT, namely for OPEN, CLOSE, and PUT requests.
DFHDUPR	Controlling routine, responsible for reading information from the dump data set DFHDMPDS.
DFHDUPS	Receives the address of a dump header record from the dump data set, and decides whether this dump fulfils the criteria for printing. On first entry, reads and stores the selective print parameters from SYSIN.
DFHDUSR	Processes requests to the DUSR and APUE gates of the dump domain
DFHDUSU	Processes domain subroutine requests of format DUSU
DFHDUSVC	System dump
DFHDUTM	Dump table manager
DFHDUXD	Invoked by DFHDUDU with a DUDD format parameter list to control the transaction dump process
DFHDUXW	Processes domain subroutine requests of format DUXW

Transaction dump formatting routines

The following routines are invoked by DFHDUXD to dump the storage areas associated with a particular CICS component. They are passed a DUXF format parameter list. They are all part of the DFHSIP load module.

Module	Function
DFHDLXDF	DL/I related areas
DFHFCXDF	File control related areas
DFHPCXDF	Program related areas
DFHSAXDF	Common areas such as CSA, TCA, and so on
DFHSMXDF	Task subpools
DFHTCXDF	Terminal control related areas
DFHTRXDF	The internal trace table
DFHXDXDF	Information such as register contents, headers, and so on

Module	Function
DFHXRXDF	XRF related areas

Exits

There are four user exit points in the dump domain, XDUCLSE, XDUOUT, XDUREQ and XDUREQC.

Enterprise Java domain (EJ)

The Enterprise Java domain provides support for Web Services Atomic Transactions (WS-AT) only.

Enterprise Java domain's specific gates

The specific gates provide access for other domains to functions that are provided by the EJ domain.

EJDU gate, DUMP_DATA function

The DUMP_DATA function of the EJDU gate is used to collect data from a dumping class. It will be placed in the chain of data collected by EJDU and formatted out when a CICS dump occurs.

Input Parameters

DATA

A pointer and length pair containing the data to be stored for inclusion in a dump.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
INSUFFICIENT_STORAGE
INTERNAL_ERROR
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EJDU gate, DUMP_STACK function

The DUMP_STACK function of the EJDU gate is used to collect the stack of a running JVM. The stack is passed as a string to EJDU and will be formatted out separately from the other data collected by EJDU's DUMP_DATA function. This function should be called before DUMP_DATA as it will free any existing data gathered for the running task.

Input Parameters

DATA

A pointer and length pair containing the data to be stored for inclusion in a dump.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
```

INSUFFICIENT_STORAGE
INTERNAL_ERROR
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EJDU gate, INQUIRE_TRACE_FLAGS function

The INQUIRE_TRACE_FLAGS function of the EJDU gate is used to return the current settings of all the trace flags. It takes into account the master trace flag setting when returning the result. The trace flags are returned as a continuous block of storage with 2 bytes for each flag, in domain order.

Input Parameters

TRACE_DATA
A block of data containing the trace flags in domain order, where each trace flag takes up 2 bytes

Output Parameters

REASON
The values for the parameter are:
ABEND
BAD_DOMAIN_TOKEN
INTERNAL_ERROR
INVALID_FORMAT
INVALID_FUNCTION
TRACE_BUFFER_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

AUX_ON
Optional Parameter

Indicates whether auxiliary trace is turned on.

Values for the parameter are:
NO
YES

Enterprise Java domain's generic gates

Table 21 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 21. Enterprise Java domain's generic gates

Gate	Trace	Functions	Format
EJDM	EJ 01*nn*	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

Modules

Module	Function
DFHEJDU	EJ Dump Interface EJDU Gate

Event manager domain (EM)

The event manager domain manages event and timer objects created within CICS BTS activities.

Event manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the EM domain.

EMBR gate, END_BROWSE_EVENT function

The END_BROWSE_EVENT function ends the event browse identified by the browse token.

Input Parameters

BROWSE_TOKEN
is a token which identifies the browse.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMBR gate, END_BROWSE_TIMER function

The END_BROWSE_TIMER function ends the timer browse identified by the browse token.

Input Parameters

BROWSE_TOKEN
is a token which identifies the browse.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMBR gate, GET_NEXT_EVENT function

The GET_NEXT_EVENT function returns the next name in the browse specified by the browse token, and returns the attributes associated with the event.

Input Parameters

BROWSE_TOKEN
is a token which identifies the browse.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
INVALID_BROWSE_TOKEN

EVENT
is the name of the retrieved reattach event.
EVENT_TYPE
is the type of the retrieved reattach event.

Values for the parameter are:
ACTIVITY
COMPOSITE
INPUT
SYSTEM
TIMER
FIRED
returns the fire status of the event.

Values for the parameter are:
NO
YES
PARENT
is the name of the parent (if the event is a subevent).
PREDICATE
is the predicate type (for composite events only).

Values for the parameter are:
AND
OR
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
TIMER_NAME
is the name of the associated timer (if the event is of type timer).

EMBR gate, GET_NEXT_TIMER function

The GET_NEXT_TIMER function returns the next name in the browse specified by the browse token, and returns the attributes associated with the timer.

Input Parameters

BROWSE_TOKEN
is a token which identifies the browse.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
INVALID_BROWSE_TOKEN
ABSTIME
returns the timer's expiry time in ABSTIME format.
EVENT
is the name of the retrieved reattach event.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
TIMER_NAME
is the name of the associated timer (if the event is of type timer).
TIMER_STATUS
returns the status of the timer.

Values for the parameter are:
EXPIRED

FORCED
UNEXPIRED

EMBR gate, INQUIRE_EVENT function

The INQUIRE_EVENT function returns information about the named event.

Input Parameters

EVENT
is the name of the composite event.

ACTIVITY_ID
Optional Parameter

is an optional activity id for the activity whose event pool is to be browsed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
EVENT_NOT_FOUND
FILE_NOT_AUTH
FILE_UNAVAILABLE
INVALID_ACTIVITY_ID
NO_CURRENT_ACTIVITY
READ_FAILURE

EVENT_TYPE
is the type of the retrieved reattach event.

Values for the parameter are:
ACTIVITY
COMPOSITE
INPUT
SYSTEM
TIMER

FIRED
returns the fire status of the event.

Values for the parameter are:
NO
YES

PARENT
is the name of the parent (if the event is a subevent).

PREDICATE
is the predicate type (for composite events only).

Values for the parameter are:
AND
OR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TIMER_NAME
is the name of the associated timer (if the event is of type timer).

EMBR gate, INQUIRE_TIMER function

The INQUIRE_TIMER function returns information about the named timer.

Input Parameters

TIMER_NAME
is the name of the timer.

ACTIVITY_ID
Optional Parameter

is an optional activity id for the activity whose event pool is to be browsed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
FILE_NOT_AUTH
FILE_UNAVAILABLE
INVALID_ACTIVITY_ID
NO_CURRENT_ACTIVITY
READ_FAILURE
TIMER_NOT_FOUND
```

ABSTIME
returns the timer's expiry time in ABSTIME format.

EVENT
is the name of the retrieved reattach event.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TIMER_STATUS
returns the status of the timer.

Values for the parameter are:

```
EXPIRED
FORCED
UNEXPIRED
```

EMBR gate, START_BROWSE_EVENT function

The START_BROWSE_EVENT function starts an event browse and returns a token to be used for the browse.

Input Parameters

ACTIVITY_ID
Optional Parameter

is an optional activity id for the activity whose event pool is to be browsed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
FILE_NOT_AUTH
FILE_UNAVAILABLE
INVALID_ACTIVITY_ID
NO_CURRENT_ACTIVITY
READ_FAILURE
```

BROWSE_TOKEN
returns a token which is used to identify the browse.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMBR gate, START_BROWSE_TIMER function

The START_BROWSE_TIMER function starts a timer browse and returns a token to be used for the browse.

Input Parameters

ACTIVITY_ID

Optional Parameter

is an optional activity id for the activity whose event pool is to be browsed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
FILE_NOT_AUTH
FILE_UNAVAILABLE
INVALID_ACTIVITY_ID
NO_CURRENT_ACTIVITY
READ_FAILURE
```

BROWSE_TOKEN

returns a token which is used to identify the browse.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, ADD_SUBEVENT function

The ADD_SUBEVENT function adds a subevent to an existing composite event.

Input Parameters

EVENT

is the name of the composite event.

SUBEVENT

is the name of the subevent.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
EVENT_NOT_FOUND
INVALID_EVENT_TYPE
INVALID_SUBEVENT
NO_CURRENT_ACTIVITY
SUBEVENT_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, CHECK_TIMER function

The CHECK_TIMER function returns the status of a timer.

Input Parameters

TIMER_NAME

is the name of the timer.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
NO_CURRENT_ACTIVITY
TIMER_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TIMER_STATUS
returns the status of the timer.

Values for the parameter are:
EXPIRED
FORCED
UNEXPIRED

EMEM gate, DEFINE_ATOMIC_EVENT function

The DEFINE_ATOMIC_EVENT function defines an atomic event of type ACTIVITY or INPUT.

Input Parameters

EVENT
is the name of the composite event.

EVENT_TYPE
is the type of the event.

Values for the parameter are:
ACTIVITY
INPUT

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DUPLICATE_EVENT
INVALID_EVENT_NAME
NO_CURRENT_ACTIVITY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, DEFINE_COMPOSITE_EVENT function

The DEFINE_COMPOSITE_EVENT function defines a composite event with an associated predicate which may be AND or OR. Up to eight subevents may be provided.

Input Parameters

EVENT
is the name of the composite event.

PREDICATE
is the predicate type.

Values for the parameter are:
AND
OR

SUBEVENT_LIST
Optional Parameter

is an optional list of up to 8 subevents.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DUPLICATE_EVENT
INVALID_EVENT_NAME
INVALID_SUBEVENT
NO_CURRENT_ACTIVITY

SUBEVENT_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SUBEVENT_IN_ERROR
returns the number of the first subevent which is in error (if any).

EMEM gate, DEFINE_TIMER function

The DEFINE_TIMER function defines a timer.

Input Parameters

TIMER_NAME
is the name of the timer.

AFTER
Optional Parameter

indicates whether or not the timer is an interval.

Values for the parameter are:
NO
YES

AT Optional Parameter

indicates whether or not the timer is a time.

Values for the parameter are:
NO
YES

DAYOFMONTH
Optional Parameter

is the day of the month.

DAYOFYEAR
Optional Parameter

is the day of the year.

DAYS
Optional Parameter

is the number of days for an interval.

EVENT
Optional Parameter

is the name of the composite event.

HOURS
Optional Parameter

is the number of hours for an interval or time.

MINUTES
Optional Parameter

is the number of minutes for an interval or time.

MONTH
Optional Parameter

is the month.

ON Optional Parameter

indicates whether or not a date has been specified.

Values for the parameter are:
NO
YES

SECONDS
Optional Parameter

is the number of seconds for an interval or time.

YEAR
Optional Parameter

is the year.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
DUPLICATE_EVENT
DUPLICATE_TIMER
INVALID_EVENT_NAME
INVALID_INTERVAL
INVALID_TIME
INVALID_TIMER_NAME
NO_CURRENT_ACTIVITY
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, DELETE_EVENT function

The DELETE_EVENT function deletes an event.

Input Parameters

EVENT
is the name of the composite event.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
EVENT_NOT_FOUND
INVALID_EVENT_TYPE
NO_CURRENT_ACTIVITY
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, DELETE_TIMER function

The DELETE_TIMER function deletes a timer.

Input Parameters

TIMER_NAME
is the name of the timer.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NO_CURRENT_ACTIVITY
TIMER_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, FIRE_EVENT function

The FIRE_EVENT function causes an event to fire.

Input Parameters

EVENT

is the name of the composite event.

EVENT_VERSION

Optional Parameter

is an optional version number for the event.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ALREADY_FIRED
EVENT_NOT_FOUND
INVALID_EVENT_TYPE
NO_CURRENT_ACTIVITY
VERSION_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, FORCE_TIMER function

The FORCE_TIMER function causes a timer to expire early.

Input Parameters

TIMER_NAME

is the name of the timer.

ACQUIRED_ACTIVITY

Optional Parameter

indicates whether or not the timer to be forced is owned by the acquired activity.

Values for the parameter are:

```
NO
YES
```

ACQUIRED_PROCESS

Optional Parameter

indicates whether or not the timer to be forced is owned by the acquired process.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_ACTIVITY
NO_ACQUIRED_ACTIVITY
NO_ACQUIRED_PROCESS
NO_CURRENT_ACTIVITY
TIMER_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, INQUIRE_STATUS function

The INQUIRE_STATUS function returns the status of the event pool for the current activity.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NO_CURRENT_ACTIVITY

EVENTS_PROCESSED
indicates whether any events were processed during this activation.

Values for the parameter are:
NO
YES

PENDING_ACTIVITY_EVENTS
indicates whether any activity events are pending.

Values for the parameter are:
NO
YES

PENDING_EVENTS
indicates whether any events are pending.

Values for the parameter are:
NO
YES

REATTACH
indicates whether the task should be reattached.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, REMOVE_SUBEVENT function

The REMOVE_SUBEVENT function removes a subevent from the named composite event.

Input Parameters

EVENT
is the name of the composite event.

SUBEVENT
is the name of the subevent.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
EVENT_NOT_FOUND
INVALID_EVENT_TYPE
INVALID_SUBEVENT
NO_CURRENT_ACTIVITY
SUBEVENT_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, RETRIEVE_REATTACH_EVENT function

The RETRIEVE_REATTACH_EVENT function retrieves the next event from the current activity's reattach queue.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
END_EVENTS
NO_CURRENT_ACTIVITY
```
EVENT
is the name of the retrieved reattach event.

EVENT_TYPE
is the type of the retrieved reattach event.

Values for the parameter are:
```
ACTIVITY
COMPOSITE
INPUT
SYSTEM
TIMER
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EMEM gate, RETRIEVE_SUBEVENT function

The RETRIEVE_SUBEVENT function retrieves the next event from the named composite event's subevent queue.

Input Parameters

EVENT
is the name of the composite event.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
END_SUBEVENTS
EVENT_NOT_FOUND
INVALID_EVENT_TYPE
NO_CURRENT_ACTIVITY
NO_SUBEVENTS
```
EVENT_TYPE
is the type of the retrieved reattach event.

Values for the parameter are:
```
ACTIVITY
INPUT
TIMER
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SUBEVENT
is the name of the subevent.

EMEM gate, TEST_EVENT function

The TEST_EVENT function returns the fire status of the named event.

Input Parameters

EVENT
is the name of the composite event.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
EVENT_NOT_FOUND
NO_CURRENT_ACTIVITY

FIRED
returns the fire status of the event.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Event manager domain's generic gates

Table 22 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 22. Event manager domain's generic gates

Gate	Trace	Functions	Format
DMDM	EM 0101 EM 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
EMBA	EM 0401 EM 0402	INQUIRE_DATA_LENGTH GET_DATA DESTROY_TOKEN	BAGD

Modules

Module	Function
DFHEMBA	Handles the following requests: INQUIRE_DATA_LENGTH GET_DATA DESTROY_TOKEN
DFHEMBR	Handles the following requests: INQUIRE_EVENT START_BROWSE_EVENT GET_NEXT_EVENT END_BROWSE_EVENT INQUIRE_TIMER START_BROWSE_TIMER GET_NEXT_TIMER END_BROWSE_TIMER

Module	Function
DFHEMDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHEMDUF	Formats the EM domain control blocks
DFHEMEM	Handles the following requests: ADD_SUBEVENT CHECK_TIMER DEFINE_ATOMIC_EVENT DEFINE_COMPOSITE_EVENT DEFINE_TIMER DELETE_EVENT DELETE_TIMER FIRE_EVENT FORCE_TIMER INQUIRE_STATUS REMOVE_SUBEVENT RESET_EVENT RETRIEVE_REATTACH_EVENT RETRIEVE_SUBEVENT TEST_EVENT
DFHEMTRI	Interprets EM domain trace entries

Event processing domain (EP)

The Event processing domain manages events captured as a result of an installed event binding.

Event processing domain's specific gates

The specific gates provide access for other domains to functions that are provided by the EP domain.

EPAD gate, ADD_EPADAPTER function

ADD_EPADAPTER adds a previously created EP adapter to a running CICS system.

Input Parameters

EPADAPTER_NAME
The name of the EPADAPTER resource.

EPADAPTER
A token identifying the EPADAPTER instance.

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
`DUPLICATE_EPADAPTER`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPAD gate, DISCARD_EPADAPTER function

DISCARD_EPADAPTER removes a previously installed EP adapter from a running CICS system.

Input Parameters

EPADAPTER_NAME
The name of the EPADAPTER resource.

BUNDLE_NAME
The name of the bundle from which the EPADAPTER was installed.

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:

```
IN_USE
NOT_FOUND
BUNDLE_MISMATCH
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPAD gate, FREE_EPADAPTER function

FREE_EPADAPTER frees the storage and deletes any related channels and containers for an EPADAPTER.

Input Parameters

EPADAPTER
A token identifying the EPADAPTER instance.

BUNDLE_NAME
The name of the bundle from which the EPADAPTER was installed.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPAD gate, RECOVER_EPADAPTERS function

RECOVER_EPADAPTERS recovers all EPADAPTERs by using the global catalog.

Input Parameters

None

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPAD gate, SET_EPADAPTER function

SET_EPADAPTER sets the status of the specified EP adapter.

Input Parameters

EPADAPTER_NAME
The name of the EPADAPTER resource.

STATUS
The new status of the EPADAPTER resource.

The values of this parameter are:

DISABLED
ENABLED

BUNDLE_NAME
Optional parameter

The name of the bundle from which the EPADAPTER was installed.

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPAS gate, FORMAT_EVENT function

FORMAT_EVENT formats a CICS event object, in the form passed to an EP adapter by the EP dispatcher, into the required EP event_format. The formatted event is returned to the caller in one container (or more than one container in the case of CCE events) in the out_pool container provided by the caller.

Input Parameters

event_format
The parameter which controls how the event is formatted.

The values of this parameter are:
CBE
CBER
CCE
CFE
DSIE
WBE

in_pool_token
a container pool containing the CICS Event Object

out_pool_token
a container pool to contain the formatted event

container_name
The name of the data container, if applicable, in the formatted event container pool. This is not set for CCE events where the container pool IS the formatted event.

Output Parameters

REASON
The following value is returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is EXCEPTION:
DATA_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_EVENT_FORMAT
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPEV gate, COMMIT_EVENT function

COMMIT_EVENT performs commit processing for transactional events.

Input Parameters

ACTION

Specifies the direction of COMMIT_EVENT.

The values of this parameter are:

BACKWARD
FORWARD

CORRELATION_FACTOR

The unit of work ID (UOWID) that is used to correlate transactional events.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPEV gate, DISPATCH_EVENT function

DISPATCH_EVENT performs the event dispatcher function.

Input Parameters

DISPATCHER

A token which identifies the event dispatcher (CEPD) task that processes the request.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPEV gate, DRAIN_EVENT_ADAPTER function

DRAIN_EVENT_ADAPTER starts draining the use of the specified adapter type.

Input Parameters

ADAPTER_TYPE

Specifies the type of adapter to start draining.

The values of this parameter are:

CUSTOM
TDQ
TSQ
WMQ
XACTION

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPEV gate, END_DRAIN_EVENT_ADAPTER function

END_DRAIN_EVENT_ADAPTER stops draining the use of the specified adapter type.

Input Parameters

ADAPTER_TYPE

Specifies the type of adapter to stop draining.

The values of this parameter are:
CUSTOM
TDQ
TSQ
WMQ
XACTION

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPEV gate, PUT_EVENT function

PUT_EVENT passes a captured event to the EP domain.

Input Parameters

ADAPTER_NAME
The name of the EP adapter used to emit the event.
CHANNEL_TOKEN
A token for the channel associated with the PUT_EVENT request.
CORRELATION_FACTOR
Optional parameter

The unit of work ID (UOWID) used to correlate transactional events.
EVBNAME
The event binding name that is inserted into a message.
EVENT_TYPE
Specifies the type of event. The values of this parameter are:
APPLICATION
SYSTEM

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
ADAPTER_NOT_FOUND
SYNC_EMIT_FAILED
SYNC_EVENT_DISCARDED
TRANS_EVENT_DISCARDED
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPEV gate, QUEUE_EVENT function

QUEUE_EVENT performs the event queue manager function.

Input Parameters

None.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPIS gate, END_BROWSE_EPADAPTER function

END_BROWSE_EPADAPTER ends a browse of EP adapters.

Input Parameters

BROWSE_TOKEN
The token that identifies the browse operation as returned by the associated START_BROWSE_EPADAPTER function call.

Output Parameters

REASON
The following value is returned when RESPONSE is EXCEPTION:
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPIS gate, GET_NEXT_EPADAPTER function

GET_NEXT_EPADAPTER returns information about the next EP adapter in the browse.

Input Parameters

BROWSE_TOKEN
The token that identifies the browse operation as returned by the associated START_BROWSE_EPADAPTER function call.

RESOURCE_SIGNATURE
Optional parameter

The resource_signature from the installation.

Output Parameters

ADAPTER_NAME
The name of the EP adapter resource.

ADAPTER_STATUS
Optional parameter

The status of the EP adapter resource.

The values of this parameter are:
DISABLED
ENABLED

ADAPTER_TYPE
Optional parameter

The type of adapter.

The values of this parameter are:
CUSTOM
HTTP
TDQ
TRANSTART
TSQ
WMQ

AUTHORITY
Optional parameter

The user ID used to run the EP adapter.

The values of this parameter are:
CONTEXT
REGION
USERID

AUTH_USERID
Optional parameter

The user ID used to attach the EP adapter transaction. This attribute is set only if **AUTHORITY** is USERID.

CONFIG_DATA1
Optional parameter

The primary configuration data item for the EP adapter. If the primary configuration data item is less than 64 bytes, the field is padded with blanks. The data item returned is dependent upon the **ADAPTER_TYPE**.

DATA_FORMAT
Optional parameter

The format of events emitted by the EP adapter.

The values of this parameter are:
CBE
CBER
CCE
CFE
DSIE
USER
WBE

EMIT_MODE
Optional parameter

The emission mode of the EP adapter.

The values of this parameter are:
ASYNC
SYNC

INVOKE_TYPE
Optional parameter

Specifies how the EP adapter is started.

The values of this parameter are:
ATTACH
LINK

PRIORITY
Optional parameter

The dispatching priority of event emission for the EP adapter.

The values of this parameter are:
HIGH
NORMAL

PROGRAM
Optional parameter

The program name for a linked adapter. If the **ADAPTER_TYPE** is CUSTOM, this attribute is applicable only if **INVOKE_TYPE** is LINK.

TRANSACTION
Optional parameter

The transaction ID for an attached adapter. This attribute is applicable only if **INVOKE_TYPE** is ATTACH.

TRANSMODE
Optional parameter

Specifies whether the event is transactional, or not.

The values of this parameter are:
NONTRANS

TRANS

REASON
The following value is returned when RESPONSE is EXCEPTION:
BROWSE_END

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPIS gate, INQ_EPADAPTER function

INQ_EPADAPTER retrieves information about a specified EP adapter.

Input Parameters

ADAPTER_NAME
The name of the EPADAPTER resource.

RESOURCE_SIGNATURE
Optional parameter

The resource_signature from the installation.

Output Parameters

ADAPTER_STATUS
Optional parameter

The status of the EP adapter resource.

The values of this parameter are:
DISABLED
ENABLED

ADAPTER_TYPE
Optional parameter

The type of EP adapter.

The values of this parameter are:
CUSTOM
HTTP
TDQ
TRANSTART
TSQ
WMQ

AUTHORITY
Optional parameter

The user ID used to run the EP adapter.

The values of this parameter are:
CONTEXT
REGION
USERID

AUTH_USERID
Optional parameter

The user ID used to attach the EP adapter transaction. This attribute is set only if **AUTHORITY** is USERID.

CONFIG_DATA1
Optional parameter

The primary configuration data item for the EP adapter. If the primary configuration data item is less than 64 bytes, the field is padded with blanks. The data item returned is dependent upon the **ADAPTER_TYPE**.

DATA_FORMAT
Optional parameter

The format of events emitted by the EP adapter.

The values of this parameter are:
CBE
CBER
CCE
CFE
DSIE
USER
WBE

EMIT_MODE
Optional parameter

The emission mode of the EP adapter.

The values of this parameter are:
ASYNC
SYNC

INVOKE_TYPE
Optional parameter

Specifies how the EP adapter is started.

The values of this parameter are:
ATTACH
LINK

PRIORITY
Optional parameter

The dispatching priority of event emission for the EP adapter.

The values of this parameter are:
HIGH
NORMAL

PROGRAM
Optional parameter

The program name for a linked adapter. If the **ADAPTER_TYPE** is CUSTOM, this attribute is applicable only if **INVOKE_TYPE** is LINK.

TRANSACTION
Optional parameter

The transaction ID for an attached adapter. This attribute is applicable only if **INVOKE_TYPE** is ATTACH.

TRANSMODE
Optional parameter

Specifies whether the event is transactional, or not.

The values of this parameter are:
NONTRANS
TRANS

REASON
The following value is returned when RESPONSE is EXCEPTION:
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPIS gate, SET_EPADAPTER function

SET_EPADAPTER sets the status of the specified EP adapter.

Input Parameters

ADAPTER_NAME

The name of the EPADAPTER resource.

ADAPTER_STATUS

The new status of the EP adapter resource.

The values of this parameter are:

```
DISABLED
ENABLED
```

Output Parameters

REASON

The following value is returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPIS gate, START_BROWSE_EPADAPTER function

START_BROWSE_EPADAPTER starts a browse of EP adapters.

Input Parameters

None

Output Parameters

BROWSE_TOKEN

The token that identifies the browse operation.

REASON

The following value is returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EPIS gate, SET_EVENT_PROCESSING function

SET_EVENT_PROCESSING sets the status of event processing.

Input Parameters

STATUS

Sets the status of event processing to be either enabled or disabled.

The values of this parameter are:

```
ACTIVE
INACTIVE
```

ACTION

Optional parameter

Instructs the EP domain to either phase out or purge the events on the event queue.

The values of this parameter are:

```
PHASEOUT
PURGE
```

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Event processing domain's generic gates

Table 23 summarizes the Event processing domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 23. Event processing domain's generic gates

Gate	Trace	Function	Format
EPDM	EP 0101 EP 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

Modules

Module	Function
DFHEPAS	EP domain adapter services
DFHEPDM	Domain initialization and termination program
DFHEPDS	EP dispatcher program
DFHEPDUF	EP domain dump formatting program
DFHEPEV	EP domain event gate functions
DFHEPIS	EP domain inquire and set program
DFHEPSS	EP domain statistics and monitoring program
DFHEPSY	EP queue manager program
DFHEPTRI	EP domain trace formatting program
DFHEPXM	EP domain XM Attach client

IP ECI (IE) domain

The IP ECI domain provides services that are used by the CICS EPI protocol over IP connections.

IP ECI domain's specific gates

The specific gates provide access for other domains to functions that are provided by the IE domain.

IEIE gate, PROCESS_ECI_FLOW function

Initiates processing of a flow from an ECI client, either by attaching a new mirror task, or by posting an existing mirror task.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
FREEMAIN_FAILURE
INSTALL_FAILED
```

INVALID_FLOW
INVALID_FORMAT
INVALID_FUNCTION
NOT_INSTALLED
RECEIVE_FAILURE
SEND_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

IEIE gate, RECEIVE function

Receives input from an ECI client.

Output Parameters

REASON
The values for the parameter are:
ABEND
CLIENT_NOT_RESPONDING
FREEMAIN_FAILURE
INVALID_FORMAT
INVALID_FUNCTION
INVALID_REQUEST
REQUEST_PURGED
WAIT_FAILURE

BINARY_FORMAT
Optional Parameter

The binary format in which numeric data is represented.

Values for the parameter are:
BIG_ENDIAN
LITTLE_ENDIAN

CLIENT_CCSID
Optional Parameter

The Coded Character Set Identifier (CCSID) of the code page used by the client.

CLIENT_INDEX
Optional Parameter

Specifies the conversion table associated with the **CLIENT_CCSID** parameter.

CODEPAGE
The code page of the request

DATA_ADDRESS
The address of the buffer containing the data received.

DATA_LENGTH
The length of the data received.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

IEIE gate, SEND function

Sends a reply to an ECI client.

Input Parameters

DATA_ADDRESS
The address of the buffer containing the data to be sent.

DATA_LENGTH
The length of the data to be sent.
LAST
This is the last send in this conversation, or not.

Values for the parameter are:
LAST_NO
LAST_YES

Output Parameters
REASON
The values for the parameter are:
ABEND
FREEMAIN_FAILURE
INVALID_FORMAT
INVALID_FUNCTION
INVALID_REQUEST
REQUEST_PURGED
SEND_FAILURE
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

IEIE gate, SEND_ERROR function
Sends an FMH7 to an ECI client.

Input Parameters
MESSAGE_NUMBER
The number of the IE component message to be sent to the client.
INSERT1
Optional Parameter

The first message insert
INSERT2
Optional Parameter

The second message insert
INSERT3
Optional Parameter

The third message insert
INSERT4
Optional Parameter

The fourth message insert

Output Parameters
REASON
The values for the parameter are:
ABEND
FREEMAIN_FAILURE
INVALID_FORMAT
INVALID_FUNCTION
INVALID_REQUEST
SEND_FAILURE
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

IP ECI domain's generic gates

Table 24 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 24. IP ECI domain's generic gates

Gate	Trace	Functions	Format
DMDM	IE 0100 IE 0101	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

Modules

Module	Function
DFHIEDM	IE domain initialization and termination.
DFHIEIE	The main part of IE domain. Processes all DFHIEIE_GATE functions.

Inter-system (IS) domain

The IS domain manages the resources, and the sending and receiving of requests and responses for IP interconnectivity (IPIC) connections.

IS domain specific gates

The specific gates provide access for other domains to functions that are provided by the IS domain.

ISCO gate, ACQUIRE_CONNECTION function

Acquire a connection to the partner CICS system named in the IPCONN parameter. The IPCONN must be INS, REL before this function is called. It opens a web session, sends a capability exchange to the partner, and waits for a response before setting the IPCONN connection status to ACQUIRED. A web session is provided to the IS domain, by the WB domain, when a socket is opened or closed for an IPIC connection.

Input Parameters

IPCONN

Optional Parameter

The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

TCPIPSERVICE

Optional Parameter

The name of the PROTOCOL(IPIC) TCPIPSERVICE definition that defines the attributes of the inbound processing for this connection.

Output Parameters

REASON

The values for the parameter are:

```
INVALID_FORMAT
INVALID_FUNCTION
INVALID_IPCONN_STATE
IPCONN_NOT_FOUND
```

ISCER_BAD_RESPONSE
ISCER_ERROR
ISCER_HTTP_ERROR
ISCER_TIMED_OUT
NO_IPCONN
SESSION_OPEN_FAILED
SHUTDOWN
TCPIP_CLOSED
TCPIPSERVICE_NOT_FOUND
TCPIPSERVICE_NOT_OPEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISCO gate, INITIALIZE_CONNECTION function

Accept an incoming connection from a partner CICS system. Called by the TCPIPSERVICE transaction, which is attached in response to a new PROTOCOl(IPIC) connection.

CICS reads the initial capability exchange, locates or creates an IPCONN to service further incoming IPIC requests from the partner and sends a response.

If a callback port is specified in the capability exchange, a connection is first made back to the client to allow outbound IPIC requests from this CICS system.

Output Parameters

REASON
The values for the parameter are:
AUTOINSTALL_FAILED
INVALID_FORMAT
INVALID_FUNCTION
INVALID_IPCONN_STATE
INVALID_PARTNER_STATE
IPCONN_NOT_FOUND
ISCE_BAD_RECOV
ISCE_ERROR
ISCE_INVALID_APPLID
ISCE_TIMED_OUT
ISCER_BAD_RESPONSE
ISCER_ERROR
ISCER_HTTP_ERROR
ISCER_TIMED_OUT
NO_IPCONN
ONE_WAY_IPCONN
SESSION_OPEN_FAILED
SHUTDOWN
TCPIP_CLOSED
TCPIPSERVICE_MISMATCH
TCPIPSERVICE_NOT_FOUND

TCPIPSERVICE_NOT_OPEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISCO gate, RELEASE_CONNECTION function

Rejects new work for the named IPCONN.

Input Parameters

DRAIN

Optional Parameter

Values for the parameter are:

YES

When YES, CICS performs the following actions:

Notifies the partner to do likewise.
Waits for work in progress, and queued work, to complete but will not allow new work to the partner to be initiated. Work for which an allocate_send has completed or is queued is allowed to complete but new allocate requests are rejected.

NO

When NO, queued work is cancelled and the partner is only notified when it attempts to send new work to this IPCONN.

When all work associated with the server is complete, the server web session is closed. A web session is provided to the IS domain, by the WB domain, when a socket is opened or closed for an IPIC connection.

The client is normally closed by the partner by passing a session_closed notification.

Once both client and server are released, the IPCONN is released.

For JCA, where there is no server session with which to notify the partner of the need to drain, incoming new work is rejected and, once the last work in progress (as indicated by the presence of an active receive session) is complete, the client session is closed.

IPCONN

Optional Parameter

The name of the IPCONN definition; that is, the name by which CICS knows the remote system.
If neither IPCONN nor TCPIPSERVICE is specified, all IPCONNs are released.

TCPIPSERVICE

Optional Parameter

The name of the PROTOCOL(IPIC) TCPIPSERVICE definition that defines the attributes of the inbound processing for this connection.
If specified, any IPCONNs referencing the given TCPIPSERVICE are released.
If neither IPCONN nor TCPIPSERVICE is specified, all IPCONNs are released.

Output Parameters

REASON

The values for the parameter are:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_IPCONN_STATE
IPCONN_NOT_FOUND
NO_IPCONN
TCPIPSERVICE_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISCO gate, TERMINATE_CONNECTION function

Release the IPCONN web sessions immediately, without waiting for any work in progress to complete. Used for error processing or when it is known that IS sessions (ISSBs) are no longer active.

Input Parameters

IPCONN

The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

SESSION_TYPE

Optional Parameter

Restricts the command to the client or the server.

Values for the parameter are:

CLIENT
SERVER

Output Parameters

REASON

The values for the parameter are:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISIC gate, ADD_IPCONN function

Create and install an IPCONN in the running system.

Input Parameters

HOST
The host name of the remote system (for example, `abc.example.com`), or its dotted decimal IP address (for example, `9.20.181.3`)

INSTALL_TYPE
IPCONN installation method.

Values for the parameter are:
`GRPLIST`
`ONLINE`
`WARM_AUTOINSTALLED`
`WARM_EXPLICIT`

IPCONN
The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

PORTNUMBER
The port number used for outbound requests on this connection; that is, the number of the port on which the remote system will listen.

TCPIPSERVICE
The name of the TCPIPSERVICE that defines the attributes of the inbound processing for this connection.

APPLID
Optional Parameter

The application identifier (applid) of the remote system. (If the remote system is a CICS region, its applid is specified on the APPLID parameter of its system initialization table.)

AUTOCONNECT
Optional Parameter

Values for the parameter are:
`AUTOCONNECT_NO`
`AUTOCONNECT_YES`

CERTIFICATE
Optional Parameter

CIPHER_LIST
Optional Parameter

INSERVICE
Optional Parameter

Values for the parameter are:
`INSERVICE_NO`
`INSERVICE_YES`

MAXQTIME
Optional Parameter

The maximum time, in seconds, for which allocate requests may be queued on this connection.

NETWORKID
Optional Parameter

The network ID of the remote system.

QUEUELIMIT
Optional Parameter

The maximum number of allocate requests that can be queued for this connection.

RECEIVECOUNT
Optional Parameter

The number of receive sessions for this connection

SECURITYNAME

Optional Parameter

The security name of the remote system.

SENDCOUNT

Optional Parameter

The number of send sessions for this connection

SSLTYPE

Optional Parameter

Whether to use secure socket layer (SSL) authentication.

Values for the parameter are:

SSL_NO
SSL_YES

USERAUTH

Optional Parameter

Type of user authentication to use.

Values for the parameter are:

CERTIFICAUTH
IDENTIFY
LOCAL
USERAUTH_NO
VERIFY

XLNACTION

Optional Parameter

Values for the parameter are:

FORCE
KEEP

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

UNLOCK_ERROR

The following values are returned when RESPONSE is EXCEPTION:

CERTIFICATE_ERROR
CIPHER_LIST_REDUCED
CIPHER_LIST_REJECTED
CONNECTION_MISMATCH
DUPLICATE_APPLID
IN_USE
NO_DEFAULT_CERTIFICATE

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID

KERNERROR
PURGED

ISIC gate, AUTOINSTALL_IPCONN function

Attempt to create an IPCONN to an unknown NETWORKID or APPLID. This function always runs on the QR TCB.

Input Parameters

APPLID
The application identifier (applid) of the remote system. (If the remote system is a CICS region, its applid is specified on the APPLID parameter of its system initialization table.)

HOST
The host name of the remote system (for example, `abc.example.com`), or its dotted decimal IP address (for example, `9.20.181.3`).

NETWORKID
The network ID of the remote system.

PORTNUMBER
The port number used for outbound requests on this connection; that is, the number of the port on which the remote system is to listen.

RECOVERY
Recovery method.

Values for the parameter are:
CICS
NON_CICS

REQUESTED_SESSIONS
The number of sessions for this connection.

TCPIPSERVICE
The name of the PROTOCOL(IPIC) TCPIPSERVICE definition that defines the attributes of the inbound processing for this connection

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
AUP_ABENDED
AUP_AMODE_ERROR
AUP_NOT_AVAILABLE
AUP_NOT_KNOWN
AUP_NOT_SPECIFIED
AUP_VETO
CONNECTION_MISMATCH
DUPLICATE_APPLID
NAME_IN_USE
NAME_INVALID
PORT_INVALID
TEMPLATE_NOT_FOUND
TEMPLATE_OUTSERVICE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

IPCONN
The name of the IPCONN definition; that is, the name of the remote system.

ISCB_TOKEN
ISCB token for this connection.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISIC gate, DISCARD_IPCONN function

Remove an IPCONN from the system, if it is in an appropriate state.

Input Parameters

IPCONN
The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
IN_USE
NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISIC gate, ENDBROWSE_IPCONN function

End an IPCONN browse.

Input Parameters

BROWSE_TOKEN

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
TOKEN_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISIC gate, GETNEXT_IPCONN function

Get the next IPCONN for browse.

Input Parameters

BROWSE_TOKEN
Dispatcher domain browse token.

CERTIFICATE
Optional Parameter

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
INVALID_BROWSE_TOKEN

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

IPCONN
The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

APPLID
Optional Parameter

Application identifier.

AUTOCONNECT
Optional Parameter

Autoconnect.

Values for the parameter are:
AUTOCONNECT_NO
AUTOCONNECT_YES

CIPHER_COUNT
Optional Parameter

CIPHER_SUITES
Optional Parameter

CONNSTATUS
Optional Parameter

Values for the parameter are:
ACQUIRED
FREEING
OBTAINING
RELEASED

HOST
Optional Parameter

Host name.

MAXQTIME
Optional Parameter

NETWORKID
Optional Parameter

Network identifier.

PENDSTATUS
Optional Parameter

Indicates whether work is pending.

Values for the parameter are:
NOTPENDING
PENDING

PORTNUMBER
Optional Parameter

Port number.

QUEUELIMIT
Optional Parameter

Queue limit.

RECEIVECOUNT
Optional Parameter

Number of receives.

RECOVSTATUS
Optional Parameter

Recovery status.

Values for the parameter are:
NORECOVDATA
NRS
RECOVDATA

SECURITYNAME
Optional Parameter

SENDCOUNT
Optional Parameter

Number of sends.

SERVSTATUS
Optional Parameter

Service status.

Values for the parameter are:
INSERV
OUTSERV

SSLTYPE
Optional Parameter

SSL type.

Values for the parameter are:
SSL_NO
SSL_YES

TCPIPSERVICE
Optional Parameter

TCPIPSERVICE name.

USERAUTH
Optional Parameter

User authentication method.

Values for the parameter are:
CERTIFICAUTH
IDENTIFY
LOCAL
USERAUTH_NO
VERIFY

ISIC gate, INQUIRE_IPCONN function

Get information about an IPCONN.

Input Parameters

IPCONN
The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

CERTIFICATE
Optional Parameter

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

APPLID
Optional Parameter

The application identifier (applid) of the remote system. (If the remote system is a CICS region, its applid is specified on the APPLID parameter of its system initialization table.)

AUTOCONNECT
Optional Parameter

Values for the parameter are:
AUTOCONNECT_NO
AUTOCONNECT_YES

CIPHER_COUNT
Optional Parameter

The number of cipher suites that are available to negotiate with clients during the SSL handshake.

CIPHER_SUITES
Optional Parameter

The list of cipher suites that is used to negotiate with clients during the SSL handshake.

CONNSTATUS
Optional Parameter

The current status of the connection.

Values for the parameter are:
ACQUIRED
FREEING
OBTAINING
RELEASED

HOST
Optional Parameter

The host name of the remote system (for example, `abc.example.com`), or its dotted decimal IP address (for example, `9.20.181.3`).

MAXQTIME
Optional Parameter

The maximum time, in seconds, for which allocate requests may be queued on this connection.

NETWORKID
Optional Parameter

The network ID of the remote system.

PENDSTATUS
Optional Parameter

Indicates whether there are any pending units of work for this connection.

Values for the parameter are:
NOTPENDING
PENDING

PORTNUMBER
Optional Parameter

The port number used for outbound requests on this connection; that is, the number of the port on which the remote system is listening.

QUEUELIMIT
Optional Parameter

The maximum number of allocate requests that can be queued for this connection.

RECEIVECOUNT
Optional Parameter

The number of receive sessions defined for this connection.

RECOVSTATUS
Optional Parameter

Recovery status of the remote connection.

Values for the parameter are:
NORECOVDATA
NRS
RECOVDATA

SECURITYNAME
Optional Parameter

Link userid used for this connection.

SENDCOUNT
Optional Parameter

The number of send sessions defined for this connection.

SERVSTATUS
Optional Parameter

Service status.

Values for the parameter are:
INSERV
OUTSERV

SSLTYPE
Optional Parameter

Indicates whether the Secure Sockets Layer (SSL) is being used to secure communications for this transaction.

Values for the parameter are:
SSL_NO
SSL_YES

TCPIPSERVICE
Optional Parameter

The name of the PROTOCOL(IPIC) TCPIPSERVICE definition that defines the attributes of the inbound processing for this connection.

USERAUTH
Optional Parameter

The level of attach-time user security used for the connection.

Values for the parameter are:
CERTIFICAUTH
IDENTIFY
LOCAL
USERAUTH_NO
VERIFY

ISIC gate, INQUIRE_IPCONN_BY_APPLID function

Get information about an IPCONN with the given APPLID.

Input Parameters

APPLID
The application identifier (applid) of the remote system. If the remote system is a CICS region, its applid is specified on the APPLID parameter of its system initialization table.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

AUTOCONNECT

Optional Parameter

Values for the parameter are:

```
AUTOCONNECT_NO
AUTOCONNECT_YES
```

CONNSTATUS

Optional Parameter

The current status of the connection.

Values for the parameter are:

```
ACQUIRED
FREEING
OBTAINING
RELEASED
```

HOST

Optional Parameter

The host name of the remote system (for example, `abc.example.com`), or its dotted decimal IP address (for example, `9.20.181.3`).

IPCONN

The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

MAXQTIME

Optional Parameter

The maximum time, in seconds, for which allocate requests may be queued on this connection.

NETWORKID

Optional Parameter

The network ID of the remote system.

PENDSTATUS

Optional Parameter

Indicates whether there are any pending units of work for this connection.

Values for the parameter are:

```
NOTPENDING
PENDING
```

PORTNUMBER
Optional Parameter

The port number used for outbound requests on this connection; that is, the number of the port on which the remote system is listening.

QUEUELIMIT
Optional Parameter

The maximum number of allocate requests that can be queued for this connection.

RECEIVECOUNT
Optional Parameter

The number of receive sessions defined for this connection.

RECOVSTATUS
Optional Parameter

Recovery status of the remote connection.

Values for the parameter are:
```
NORECOVDATA
NRS
RECOVDATA
```

SECURITYNAME
Optional Parameter

Link userid used for this connection.

SENDCOUNT
Optional Parameter

The number of send sessions defined for this connection.

SERVSTATUS
Optional Parameter

Service status.

Values for the parameter are:
```
INSERV
OUTSERV
```

TCPIPSERVICE
Optional Parameter

The name of the PROTOCOL(IPIC) TCPIPSERVICE definition that defines the attributes of the inbound processing for this connection.

ISIC gate, SET_IPCONN function

Change the attributes of an IPCONN or cancel outstanding AIDs.

Input Parameters

IPCONN
Name of the IPCONN.

CONNSTATUS
Optional Parameter

Connection status.

Values for the parameter are:
```
ACQUIRED
RELEASED
```

PENDSTATUS
Optional Parameter

Indicates whether work is pending on this connection.

Values for the parameter are:
PENDING

PURGETYPE
Optional Parameter

Specifies the conditions for CICS to purge the task.

Values for the parameter are:
CANCEL
FORCECANCEL
FORCEPURGE
KILL
PURGE

RECOVSTATUS
Optional Parameter

Recovery status for this conection.

Values for the parameter are:
NORECOVDATA

SERVSTATUS
Optional Parameter

Service status for this connection.

Values for the parameter are:
INSERV
OUTSERV

UOWACTION
Optional Parameter

Normal resynchronization process is to be partially overridden: decisions are taken for any units of work that are indoubt because of a failure of the IPCONN; but the decisions are recorded and any data inconsistencies are reported when the connection is next acquired.

Values for the parameter are:
BACKOUT
COMMIT
FORCEUOW
RESYNC

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ACQUIRED_ONE_WAY
ACQUIRED_WHEN_FREEING
NOT_FOUND
NOTPENDING_ERROR
RECOVSTATUS_INVALID
SERVSTATUS_ERROR
UNSUCCESSFUL_BACKOUT

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ALLOCATES_CANCELLED
Optional Parameter

Indicates whether allocates are cancelled.

Values for the parameter are:
CANCELLED_NO
CANCELLED_YES

ISIC gate, STARTBROWSE_IPCONN function

Start a browse operation on IPCONN resources.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

BROWSE_TOKEN
The browse token for the browse operation.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISIF gate, GET_IPFACILITY_LIST function

Input Parameters

TASK_NUMBER

IP_FACILITY_LIST
Optional Parameter

Name of list to get.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BUFFER_NOT_BIG_ENOUGH
NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

LIST_SIZE
Size of retrieved list.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

ISIF gate, INQUIRE_IPFACILITY function

Retrieve information about an IPCONN facility.

Input Parameters

FACILITY_TOKEN
IPCONN facility token.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
NOT_FOUND
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

IPCONN
Optional Parameter

The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

IPFACILITY_TYPE
Optional Parameter

Values for the parameter are:
```
ALTERNATE
PRINCIPAL
```

ISIS gate, ALLOCATE_SEND function

Allocate a session on the named IPIC connection.

Input Parameters

IPCONN
The name of the IPCONN resource that is used to route the transaction to the remote CICS region.

QUEUE
Flag indicating whether to queue if no sessions are immediately available.

Values for the parameter are:
YES
NO

FUNCTION_AREA
Indicates the CICS functional areas for which the session is being used. This parameter is passed to the user exit.

Value for the parameter is:
transaction_routing

TRAN_REMOTENAME
Name of the transaction to be routed. This parameter is passed to the user exit (for UEPTRANR).

Output Parameters

SESSION
Pointer to the ISSB for the link to the remote CICS region.

REASON
The following values are returned when RESPONSE is EXCEPTION:
ISIS_NOT_FOUND
ISIS_CAPABILITIES_UNKNOWN
ISIS_NOT_IN_SERVICE
ISIS_ALLOCATE_REJECTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:
OK
EXCEPTION
INVALID

ISIS gate, BIND_RECEIVER function

Sets the IPCONN to be the BIND receiver

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISIS gate, CONVERSE function

Send a request to a partner system using an IPCONN.

Input Parameters

EISQBUSY
Indicates whether QBUSY HANDLE is set for a transient data request.

Values for the parameter are:
NO
YES

EISNOSPA
Indicates whether HANDLE NOSPACE is set or NOSUSPEND was specified for a temporary storage request.

Values for the parameter are:
NO
YES

EISTDIA_ADDR
The address of the transient data input area (TDIA) for a transient data request.

EISTSIOA_ADDR
The address of the temporary storage input output area (TSIOA) for a temporary storage request.

EXEC_ARGS
Specifies the argument string being passed.

IPCONN
The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

QUEUE
Indicates whether the request is queued.

Values for the parameter are:
NO
YES

NETNAME
The applid of the owning terminal for a START request.

NETNAME_SET
Indicates whether a netname has been specified for a START request.

Values for the parameter are:
NO
YES

REMOTE_TRAN_ACD
The application context for a remote transaction which is defined as an application entry point.

ROUTABLE_START
Indicates whether the START request is routable for a START request.

Values for the parameter are:
NO
YES

TDLEN
The length of the data to read/write for a transient data request.

TDNAME
The name of the remote TD queue for a transient data request.

TERM_NETNAME
The terminal netname for a START request.

TERM_NETNAME_SET
Indicates whether a terminal netname has been specified for a START request.

Values for the parameter are:
NO
YES

TRANSID
The name of the transaction to be run by a task started as the result of a START request.

XFSTG
Transform storage area.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ALLOCATE_REJECTED
CONVERSATION_FAILURE
QUEUE_LIMIT_REACHED
QUEUE_PURGED
NO_SESSION
NOT_FOUND
NOT_IN_SERVICE
PROGRAM_ABEND
RESOURCE_UNAVAILABLE
SECURITY_VIOLATION
TPN_NOT_RECOGNIZED
UNSUPPORTED_REQUEST
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_DATA_LENGTH
INVALID_FORMAT
INVALID_FUNCTION
INVALID_HRS
INVALID_MINS
INVALID_SECS
INVALID_SYNCONRETURN
INVALID_TRANSID
```

ABEND_CODE

EXEC abend code.

CHANNEL_BYTES

The size of the containers shipped, in bytes, for a START CHANNEL request. This total includes the length of any headers to the data.

EIBRCODE

EIB reason code.

EISCSET1

The length of the data returned on a READQ transient data or temporary storage request.

REQID

The request identifier for a START request.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

WLMRCODE

Workload manager response code.

ISIS gate, INITIALIZE_RECEIVER function

Check that the inbound message is consistent with the IPCONN USERAUTH attribute and return an error response if it is inconsistent.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

MESSAGE_MISMATCH_IDENTIFY
MESSAGE_MISMATCH_LOCAL
MESSAGE_MISMATCH_VERIFY
SECURITY_INACTIVE
SECURITY_VIOLATION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

SET_USER_TOKEN

Indicactes whether a user token is be used to identify the inbound message sender.

Values for the parameter are:

NO
YES

USER_TOKEN

User token associated with the inbound message sender.

ISIS gate, INQUIRE_FACILITY function

Expose web session token. A web session is provided to the IS domain, by the WB domain, when a socket is opened or closed for an IPIC connection.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND

The following values are returned when RESPONSE is EXCEPTION:

ALLOCATE_REJECTED
CONVERSATION_FAILURE
FACILITY_NOT_ISSESSION
MESSAGE_MISMATCH_IDENTIFY
MESSAGE_MISMATCH_LOCAL
MESSAGE_MISMATCH_VERIFY
NO_DATA
NO_SESSION
NOT_FOUND
NOT_IN_SERVICE
PROGRAM_ABEND
RESOURCE_UNAVAILABLE
SECURITY_INACTIVE
SECURITY_VIOLATION
UNSUPPORTED_REQUEST

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

INVALID_SYNCONRETURN
INVALID_TRANSID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

IPCONN
Optional Parameter

The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

WB_SESSION
Optional Parameter

Web session identifier.

ISIS gate, RECEIVE_BUFFER function

Receive the next buffer on the specified session. This function is used when the channel being transmitted does not fit into the first buffer.

Input Parameters

BUFFER_TYPE
Specifies whether this buffer is for a request or a response.

Values for the parameter are:
REQ
RESP

SESSION
Session name.

Output Parameters

LAST_IN_CHAIN
Indicates whether the buffer is last in chain.

Values for the parameter are:
LIC
NOT_LIC

DATA_BUFFER
Optional parameter.

Address and length of the data.

CONTINUE
Flag indicating whether the conversation ends after a request has been processed.

Values for the parameter are:
YES
NO

CONDITION
Indicates the action if CONTINUE is set to NO.

Values for the parameter are:
NORMAL

END
SYNCPOINT
ROLLBACK
ABENDED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is EXCEPTION:
ALLOCATE_REJECTED
CONVERSATION_FAILURE
FACILITY_NOT_ISSESSION
MESSAGE_MISMATCH_IDENTIFY
MESSAGE_MISMATCH_LOCAL
MESSAGE_MISMATCH_VERIFY
NO_DATA
NO_SESSION
NOT_FOUND
NOT_IN_SERVICE
PROGRAM_ABEND
RESOURCE_UNAVAILABLE
SECURITY_INACTIVE
SECURITY_VIOLATION
UNSUPPORTED_REQUEST

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SYNCONRETURN
INVALID_TRANSID

ISIS gate, RECEIVE_REQUEST function

Receive a complete request from the request stream domain.

Input Parameters

EXEC_ARGS
Argument string

XFSTG
Transform.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is EXCEPTION:
ALLOCATE_REJECTED

CONVERSATION_FAILURE
FACILITY_NOT_ISSESSION
MESSAGE_MISMATCH_IDENTIFY
MESSAGE_MISMATCH_LOCAL
MESSAGE_MISMATCH_VERIFY
NO_DATA
NO_SESSION
NOT_FOUND
NOT_IN_SERVICE
PROGRAM_ABEND
RESOURCE_UNAVAILABLE
SECURITY_INACTIVE
SECURITY_VIOLATION
UNSUPPORTED_REQUEST

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SYNCONRETURN
INVALID_TRANSID

INVOKING_PROGRAM
Name of the program that invoked this function.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

TRANSID
Transaction identifier.

CONDITION
Optional Parameter

Values for the parameter are:
ABENDED
NORMAL
ROLLBACK
SYNCPOINT

CONTINUE
Optional Parameter

Indicates whether this function should listen for the next request .

Values for the parameter are:
NO
YES

ISIS gate, RETRIEVE_ICRX function

If CICS receives an ICRX on a start request over an IPIC session, it is saved. This function is called when the ICE for the started task is being built.

Input Parameters

FACILITY
The principal facility token, which will be the IPIC session. Specifically, the ISSB address.

Output Parameters

OUT_ICRX
The ICRX representing the user.

Reason
NOT_FOUND

Response
Indicates whether the domain call was successful.

ISIS gate, ROUTING_CONVERSE function

Send data that is already transformed to the remote CICS region and receive the response data.

Input Parameters

SESSION_TOKEN
Pointer to the ISSB for the link to the remote CICS region, as returned by the ISIS gate, ALLOCATE_SEND function.

BUFFER_TYPE
Specifies whether this buffer is for a request or a response.

Values for the parameter are:
```
REQ
RESP
```

CHAINING
Specifies whether this request or response is one of a chain of requests or responses from the remote CICS region.

Values for the parameter are:
```
CHAIN
NOT_CHAIN
```

Output Parameters

LAST_IN_CHAIN
Flag indicating whether more data is to be transferred from the CICS remote region. If this flag is set to YES, the ISIS gate, RECEIVE_BUFFER, is used to retrieve the remaining data.

Values for the parameter are:
```
YES
NO
```

CONTINUE
Flag indicating whether the conversation ends after a request has been processed.

Values for the parameter are:
```
YES
NO
```

CONDITION
Indicates the action if CONTINUE is set to NO.

Values for the parameter are:
```
NORMAL
END
SYNCPOINT
ROLLBACK
```

ABENDED

REASON
The following values are returned when RESPONSE is EXCEPTION:
ISIS_CONVERSATION_FAILURE
ISIS_TPN_NOT_RECOGNISED
ISIS_NOT_FOUND
ISIS_TRANSACTION_DISABLED
ISIS_REMOTE_SYSTEM_QUIESCING

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:
OK
EXCEPTION
INVALID
PURGED

ISIS gate, SEND_BUFFER function

Send the current buffer of the specified session. This function is used when the channel being transmitted is too large for the first buffer.

Input Parameters

BUFFER_TYPE
Specifies whether this buffer is for a request or a response.

Values for the parameter are:
REQ
RESP

DATA_BUFFER
Address and size of the buffer.

CHAINING
Specifies whether the buffer is chained.

Values for the parameter are:
CHAIN
NOT_CHAIN

SESSION
Session name

LAST_IN_CHAIN
Optional parameter.

Specifies whether the buffer is last in chain.

Values for the parameter are:
LIC
NOT_LIC

LAST
Flag indicating whether this message is the last for this transaction.

Values for the parameter are:
YES
NO

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is EXCEPTION:

ALLOCATE_REJECTED
CONVERSATION_FAILURE
FACILITY_NOT_ISSESSION
MESSAGE_MISMATCH_IDENTIFY
MESSAGE_MISMATCH_LOCAL
MESSAGE_MISMATCH_VERIFY
NO_DATA
NO_SESSION
NOT_FOUND
NOT_IN_SERVICE
PROGRAM_ABEND
RESOURCE_UNAVAILABLE
SECURITY_INACTIVE
SECURITY_VIOLATION
UNSUPPORTED_REQUEST

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_SYNCONRETURN
INVALID_TRANSID

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISIS gate, SEND_ERROR function

Issue a CICS message based on the sense code and, if the session is in the incorrect state, send an IS7 error message back to the client.

Input Parameters

SENSE

Sense code.

ABEND_CODE

Optional Parameter

Abend code.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND

The following values are returned when RESPONSE is EXCEPTION:

ALLOCATE_REJECTED
CONVERSATION_FAILURE
FACILITY_NOT_ISSESSION
MESSAGE_MISMATCH_IDENTIFY
MESSAGE_MISMATCH_LOCAL
MESSAGE_MISMATCH_VERIFY
NO_DATA

NO_SESSION
NOT_FOUND
NOT_IN_SERVICE
PROGRAM_ABEND
RESOURCE_UNAVAILABLE
SECURITY_INACTIVE
SECURITY_VIOLATION
UNSUPPORTED_REQUEST

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SYNCONRETURN
INVALID_TRANSID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISIS gate, SEND_RESPONSE function
Sends the response data back to the caller.

Input Parameters
EIBRCODE
EIB reason code.
EXEC_ARGS
Argument string.
XFSTG
Transform storage area.

Output Parameters
REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is EXCEPTION:
ALLOCATE_REJECTED
CONVERSATION_FAILURE
FACILITY_NOT_ISSESSION
MESSAGE_MISMATCH_IDENTIFY
MESSAGE_MISMATCH_LOCAL
MESSAGE_MISMATCH_VERIFY
NO_DATA
NO_SESSION
NOT_FOUND
NOT_IN_SERVICE
PROGRAM_ABEND
RESOURCE_UNAVAILABLE
SECURITY_INACTIVE
SECURITY_VIOLATION
UNSUPPORTED_REQUEST

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_SYNCONRETURN
INVALID_TRANSID

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

WLMRCODE

Workload Manager response code.

ISIS gate, SET_PARAMETERS function

Modify parameters for the IS domain obtained by Parameter Manager .

Input Parameters

CONFDATA

Optional Parameter

Specifies whether CICS is to suppress (hide) user data that might otherwise appear in CICS trace entries or in dumps.

Values for the parameter are:

HIDETC
SHOW

NETWORKID

Optional Parameter

Network identifier.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND

The following values are returned when RESPONSE is EXCEPTION:

ALLOCATE_REJECTED
CONVERSATION_FAILURE
FACILITY_NOT_ISSESSION
MESSAGE_MISMATCH_IDENTIFY
MESSAGE_MISMATCH_LOCAL
MESSAGE_MISMATCH_VERIFY
NO_DATA
NO_SESSION
NOT_FOUND
NOT_IN_SERVICE
PROGRAM_ABEND
RESOURCE_UNAVAILABLE
SECURITY_INACTIVE
SECURITY_VIOLATION
UNSUPPORTED_REQUEST

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SYNCONRETURN
INVALID_TRANSID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISRE gate, CICS_RESYNC function

Respond to messages from a partner CICS region that is attempting to resynchronize work after a connection is reestablished over IPCONNs.

When communication is reestablished between a pair of CICS regions over IPCONNs, one region assumes responsibility for a resync attempt, while the other calls the CICS_RESYNC function and waits for instructions from its partner. The CICS_RESYNC function responds to any messages that the partner sends it until the resync attempt is completed.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISRE gate, FORCE_LINKS function

Help force UOWs following an Exchange Log Name (XLN) failure during Acquire.

This function is called under the following circumstances to force indoubt and shunted UOWs associated with an IPCONN to complete heuristically:

- Following an Exchange Log Name (XLN) failure during Acquire, when the IPCONN is defined with XLNACTION(FORCE).
- In response to SET IPCONN() NOTPENDING, when the connection is acquired service and has pending work.
- In response to SET IPCONN() NORECOVDATA, when the connection is released and has outstanding work associated with it.

Input Parameters

IPCONN_NAME
The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

ISRE gate, KEEP_LINKS function

Looks for any outstanding UOWs that are either indoubt and shunted, or committed and awaiting forget, following an Exchange Log Name (XLN) failure.

This function is called when the connection is being acquired and an XLN failure is detected, and the local IPCONN is configured with XLNACTION(KEEP). If any outstanding UOWs are found, then a message is issued for each one indicating that a resync attempt could not be carried out because of the XLN failure, and the PENDING condition is raised for the IPCONN.

Input Parameters

IPCONN_NAME

The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

ISRE gate, RESYNC_LINKS function

Attempt to resynchronize links following reestablishment of an IPCONN.

When communication is reestablished between a pair of regions over IPCONNs, one region assumes responsibility for an attempt to resynchronize links, and calls this function to initiate it.

The function looks for units of work on the local system associated with the IPCONN resource that are either indoubt and shunted, or committed and awaiting forget, and attempts to drive them to completion. When it has processed its own work, the function passes control to the partner region to carry out the same activity there.

When the function has completed, both regions know the outcome of the resync attempt, and can either put their end of the connection into service, or mark it to show that there is still further resync work to be carried out.

Input Parameters

IPCONN_NAME

The name of the IPCONN definition; that is, the name by which CICS knows the remote system.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

COMBINED_FAILURE
LOCAL_FAILURE
REMOTE_FAILURE

RESPONSE

Indicates whether the domain call was successful. For more information, see reference.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISRE gate, XA_RESYNC function

Resynchronize XA links in response to a request from an XA client.

An XA client can make one of two types of resync requests into CICS:

1. A request for a list of XIDs to be returned to the client, for all outstanding units of work that are associated with a connection that are indoubt and shunted.
2. A request to schedule a resync attempt for a specific unit of work based upon its associated XID.

CICS uses the XA_RESYNC function to respond to either of these requests.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISRR gate, NOTIFY function

Notify the system of an event on an IPCONN.

Input Parameters

ACTION

Event being performed.

Values for the parameter are:

DATA
ERROR
SERVICE_CLOSING

SERVICE_OPENED
SESSION_CANCELLED
SESSION_CLOSED
TIMEOUT

SESSION_TOKEN
IPCONN Sesstion Token.

USER_TOKEN
User token associated with the session token.

Output Parameters

REASON
The values for the parameter are:
INVALID_ACTION
INVALID_FORMAT
INVALID_FUNCTION
INVALID_USER_TOKEN
UNEXPECTED_EXCEPTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISRR gate, NOTIFY_SERVICE function

Notifiy the system of an event relating to an IPIC TCPIPSERVICE.

Input Parameters

ACTION
The event being performed by the TCPIPSERVICE.

Values for the parameter are:
DATA
ERROR
SERVICE_CLOSING
SERVICE_OPENED
SESSION_CANCELLED
SESSION_CLOSED
TIMEOUT

TCPIPSERVICE
Optional Parameter

The name of the PROTOCOL(IPIC) TCPIPSERVICE definition that defines the attributes of the inbound processing for this connection.

If no TCPIPSERVICE name is supplied, the action relates to all connections in the system with TCPIPSERVICE(IPIC).

Output Parameters

REASON
The values for the parameter are:
ATTACH_FAILED
INVALID_ACTION

INVALID_FORMAT
INVALID_FUNCTION
UNEXPECTED_EXCEPTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISRR gate, PROCESS_ERROR_QUEUE function

Handle errors that require error processing, message processing, or both.

Output Parameters

REASON

The values for the parameter are:

INVALID_FORMAT
INVALID_FUNCTION
SHUTDOWN
UNEXPECTED_EXCEPTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ISRR gate, PROCESS_INPUT_QUEUE function

Handle inbound requests and responses for all IPCONNs.

Output Parameters

REASON

The values for the parameter are:

BAD_INPUT_QUEUE
INVALID_FORMAT
INVALID_FUNCTION
SHUTDOWN
UNEXPECTED_EXCEPTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR

PURGED

ISRR gate, TERMINATE_INPUT function

Terminate the handling of the request/response input queue at CICS termination.

Output Parameters

REASON

The values for the parameter are:

INVALID_FORMAT
INVALID_FUNCTION
UNEXPECTED_EXCEPTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

IS domain modules

Module	Function
DFHISAIP	Autoinstall user program to allow tailoring of autoinstalled IPCONN resources. Assembler user replaceable module. Default configuration.
DFHISCIP	Autoinstall user program to allow tailoring of autoinstalled IPCONN resources. COBOL version of DFHISAIP.
DFHISDIP	Autoinstall user program to allow tailoring of autoinstalled IPCONN resources. C version of DFHISAIP.
DFHISPIP	Autoinstall user program to allow tailoring of autoinstalled IPCONN resources. PL/I version of DFHISAIP.
DFH0ISAI	Sample COBOL program to illustrate a technique of customizing the autoinstall of an IPCONN resource. The IPCONN name and APPLID are generated according to a template IPCONN.
DFH$ISAI	Sample Assembler program to illustrate a technique of customizing the autoinstall of an IPCONN resource. The IPCONN name and APPLID are generated according to a template IPCONN.
DFHISAL	IPCONN resource session management.
DFHISBU	Returns the entry points of the ISCU, and ISJU gates, which process the calls issued to RMCs during sync point.
DFHISCO	Basic connectivity functions for IPCONN resources.
DFHISCOP	The initial program for the IS domain connectivity transactions.
DFHISCU	Performs the processing for CICS to CICS communication using IPIC, and for JCA to CICS (respectively) during UOW sync point.

Module	Function
DFHISDM	IS initialization and termination.
DFHISDUF	IS Domain dump formatting.
DFHISEM	IPIC errors and messages.
DFHISFS	IS FS Transformers.
DFHISIC	IPCONN resource management.
DFHISIF	IS Inquire IP Facilities data gate.
DFHISIS	IPIC main functions.
DFHISJU	Entry points for ISCU and ISJU.
DFHISRE	ISRE gate module.
DFHISREX	IPCONN resource resync recovery for XA.
DFHISRE1	IPCONN resource resync recovery for CICS.
DFHISRR	IPIC inbound request and response.
DFHISRRP	IPIC receiver.
DFHISSR	IPIC inbound request and response.
DFHISTRI	IS Domain Trace Interpretation.
DFHISUE	IS Domain User Exit Control.
DFHISXF	IS Request Transformers.
DFHISXFT	IS Transformers.
DFHISXM	IS XM Attach client.
DFHISZA	IS Domain Request Logic.

Kernel domain (KE)

The kernel domain provides a consistent linkage and recovery environment for CICS.

Kernel domain's specific gates

The specific gates provide access for other domains to functions that are provided by the KE domain.

KEAR gate, DEREGISTER function

The DEREGISTER function of the KEAR gate is used when performing a normal shutdown (and optionally at an immediate shutdown) to deregister CICS(R) from the MVS(TM) automatic restart manager.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEAR gate, READY function

The READY function of the KEAR gate is used at the end of CICS initialization to indicate to the MVS automatic restart manager. that this CICS region is ready for work.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEAR gate, REGISTER function

The REGISTER function of the KEAR gate is used very early in CICS initialization to register CICS with the MVS automatic restart manager.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEAR gate, WAITPRED function

The WAITPRED function of the KEAR gate is used to wait on predecessors in the restart policy for this CICS region, to ensure that prerequisite subsystems are available to CICS.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, ADD_DOMAIN function

The ADD_DOMAIN function of the KEDD gate is used to add a new domain to the domain table.

Input Parameters

DOMAIN_NAME

is the 8-character domain name for the new domain to be added.

DOMAIN_TOKEN

is the 31-bit constant that uniquely identifies the domain, for example, DFHSM_DOMAIN for storage manager domain.

ENTRY_POINT

is the 31-bit address of the entry point for that domain, for example, A(X'80000000' + DFHSMDM) for storage manager domain.

DOMAIN_AFFINITY

Optional Parameter

is the TCB that the domain has affinity with for TERMINATE_DOMAIN.

Values for the parameter are:

```
CO
FO
QR
RO
STEP
```

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
DUPLICATE_DOMAIN_NAME
DUPLICATE_DOMAIN_TOKEN
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_DOMAIN_TOKEN
INVALID_ENTRY_POINT
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, ADD_GATE function

The ADD_GATE function of the KEDD gate is used to update the domain table to add a new gate to the calling domain's gate table.

Input Parameters

ENTRY_POINT
is the 31-bit address of the entry point for that domain, for example, A(X'80000000' + DFHSMDM) for storage manager domain.

GATE_INDEX
is the 31-bit constant that uniquely identifies the gate in the domain's gate table.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
DUPLICATE_GATE_INDEX
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_DOMAIN_TOKEN
INVALID_ENTRY_POINT
INVALID_GATE_INDEX
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, DELETE_GATE function

The DELETE_GATE function of the KEDD gate is used to delete an existing gate from the calling domain's gate table.

Input Parameters

GATE_INDEX
is the 31-bit constant that uniquely identifies the gate in the domain's gate table.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:

```
INVALID_DOMAIN_TOKEN
INVALID_GATE_INDEX
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, INQUIRE_ANCHOR function

The INQUIRE_ANCHOR function of the KEDD gate is used to return the global storage pointer of the specified domain to the caller. If the domain token is omitted, the calling domain is assumed.

Input Parameters

DOMAIN_TOKEN
Optional Parameter

The 31-bit constant that uniquely identifies the domain; for example, DFHSM_DOMAIN for the storage manager domain.

Output Parameters

ANCHOR | ANCHOR_64

ANCHOR
The 31-bit address of the global storage of the domain.

ANCHOR_64
The 64-bit address of the global storage of the domain.

REASON
The following values are returned when RESPONSE is EXCEPTION:
`DOMAIN_TOKEN_NOT_FOUND`

The following values are returned when RESPONSE is INVALID:
`INVALID_DOMAIN_TOKEN`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, INQUIRE_DOMAIN_BY_NAME function

The INQUIRE_DOMAIN_BY_NAME function of the KEDD gate is used to return the domain token for a given domain name.

Input Parameters

DOMAIN_NAME
is the 8-character domain name for the new domain to be added.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`DOMAIN_NAME_NOT_FOUND`

DOMAIN_TOKEN
is the 31-bit constant that uniquely identifies the domain.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, INQUIRE_DOMAIN_BY_TOKEN function

The INQUIRE_DOMAIN_BY_TOKEN function of the KEDD gate is used to return the domain name for a specified domain token.

Input Parameters

DOMAIN_TOKEN
is the 31-bit constant that uniquely identifies the domain, for example, DFHSM_DOMAIN for storage manager domain.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`DOMAIN_TOKEN_NOT_FOUND`

The following values are returned when RESPONSE is INVALID:
`INVALID_DOMAIN_TOKEN`

DOMAIN_NAME
is the 8-character domain name for the new domain to be added.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, INQUIRE_DOMAIN_TRACE function

The INQUIRE_DOMAIN_TRACE function of the KEDD gate is used to return the value of the specified domain's trace flags to the caller. If the domain token is omitted, the calling domain is assumed.

Input Parameters

DOMAIN_TOKEN
Optional Parameter

is the 31-bit constant that uniquely identifies the domain, for example, DFHSM_DOMAIN for storage manager domain.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`DOMAIN_TOKEN_NOT_FOUND`

The following values are returned when RESPONSE is INVALID:
`INVALID_DOMAIN_TOKEN`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SPECIAL_TRACE_FLAGS
Optional Parameter

is the set of 32 bits which determines selectivity of tracing within the domain for special tasks.

STANDARD_TRACE_FLAGS
Optional Parameter

is the set of 32 bits which determines selectivity of tracing within the domain for standard tasks.

KEDD gate, INQUIRE_GLOBAL_TRACE function

The INQUIRE_GLOBAL_TRACE function of the KEDD gate is used to return the value of the global trace flags to the caller.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
`ABEND`
`LOOP`

The following values are returned when RESPONSE is EXCEPTION:
`DOMAIN_NAME_NOT_FOUND`
`DOMAIN_TOKEN_NOT_FOUND`
`DUPLICATE_DOMAIN_NAME`
`DUPLICATE_DOMAIN_TOKEN`
`DUPLICATE_GATE_INDEX`

The following values are returned when RESPONSE is INVALID:
`INVALID_DOMAIN_TOKEN`
`INVALID_ENTRY_POINT`
`INVALID_FUNCTION`
`INVALID_GATE_INDEX`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MASTER_TRACE_FLAG
Optional Parameter

determines whether tracing, for any of the trace destinations, is active.

Values for the parameter are:
OFF
ON

SYSTEM_TRACE_FLAG
Optional Parameter

determines whether tracing is allowed for tasks for which standard tracing is in effect.

Values for the parameter are:
OFF
ON

KEDD gate, INQUIRE_TASK_TRACE function

The INQUIRE_TASK_TRACE function of the KEDD gate is used to return the value of the calling task's trace flag to the caller.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
DOMAIN_NAME_NOT_FOUND
DOMAIN_TOKEN_NOT_FOUND
DUPLICATE_DOMAIN_NAME
DUPLICATE_DOMAIN_TOKEN
DUPLICATE_GATE_INDEX

The following values are returned when RESPONSE is INVALID:
INVALID_DOMAIN_TOKEN
INVALID_ENTRY_POINT
INVALID_FUNCTION
INVALID_GATE_INDEX

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRACE_TYPE
Optional Parameter

determines whether standard, special, or no tracing is required for this task.

Values for the parameter are:
SPECIAL
STANDARD
SUPPRESSED

KEDD gate, PERFORM_SYSTEM_ACTION function

The PERFORM_SYSTEM_ACTION function of the KEDD gate is used in exceptional circumstances either to terminate CICS (with or without a dump) or to take an MVS SDUMP.

Input Parameters

DUMP_SYSTEM

Optional Parameter

Specifies whether an MVS SDUMP is to be taken or not.

Values for the parameter are:

NO
YES

NORMAL_TERMINATION

Optional Parameter

Specifies whether CICS is being terminated normally. Normal termination includes controlled and immediate shutdowns.

Values for the parameter are:

NO
YES

TERMINATE_SYSTEM

Optional Parameter

Specifies whether CICS is to be terminated or not.

Values for the parameter are:

NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

DOMAIN_NAME_NOT_FOUND
DOMAIN_TOKEN_NOT_FOUND
DUPLICATE_DOMAIN_NAME
DUPLICATE_DOMAIN_TOKEN
DUPLICATE_GATE_INDEX

The following values are returned when RESPONSE is INVALID:

INVALID_DOMAIN_TOKEN
INVALID_ENTRY_POINT
INVALID_FUNCTION
INVALID_GATE_INDEX

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, SET_ANCHOR function

The SET_ANCHOR function of the KEDD gate is used to establish the global storage pointer of the calling domain.

Input Parameters

ANCHOR | ANCHOR_64

ANCHOR

The 31-bit address of the global storage of the domain.

ANCHOR_64

The 64-bit address of the global storage of the domain.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

`INVALID_DOMAIN_TOKEN`

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, SET_DEFAULT_RECOVERY function

The SET_DEFAULT_RECOVERY function of the KEDD gate is used to establish the default recovery routine of the calling domain. The Application domain uses this function to identify DFHSRP as its default recovery routine.

Input Parameters

ENTRY_POINT_64

The 64-bit address of the default recovery routine for the domain.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

`INVALID_DOMAIN_TOKEN`

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, SET_DOMAIN_TRACE function

The SET_DOMAIN_TRACE function of the KEDD gate is used to store the value of the specified domain's trace flags in the kernel. If the domain token is omitted, the calling domain is assumed.

Input Parameters

DOMAIN_TOKEN

Optional Parameter

is the 31-bit constant that uniquely identifies the domain, for example, DFHSM_DOMAIN for storage manager domain.

SPECIAL_TRACE_FLAGS

Optional Parameter

is the set of 32 bits which determines selectivity of tracing within the domain for special tasks.

STANDARD_TRACE_FLAGS

Optional Parameter

is the set of 32 bits which determines selectivity of tracing within the domain for standard tasks.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

`DOMAIN_TOKEN_NOT_FOUND`

The following values are returned when RESPONSE is INVALID:

`INVALID_DOMAIN_TOKEN`

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, SET_GLOBAL_TRACE function

The SET_GLOBAL_TRACE function of the KEDD gate is used to store the value of the global trace flags within the kernel.

Input Parameters

MASTER_TRACE_FLAG

Optional Parameter

determines whether tracing, for any of the trace destinations, is active.

Values for the parameter are:

OFF
ON

SYSTEM_TRACE_FLAG

Optional Parameter

determines whether tracing is allowed for tasks for which standard tracing is in effect.

Values for the parameter are:

OFF
ON

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

DOMAIN_NAME_NOT_FOUND
DOMAIN_TOKEN_NOT_FOUND
DUPLICATE_DOMAIN_NAME
DUPLICATE_DOMAIN_TOKEN
DUPLICATE_GATE_INDEX

The following values are returned when RESPONSE is INVALID:

INVALID_DOMAIN_TOKEN
INVALID_ENTRY_POINT
INVALID_FUNCTION
INVALID_GATE_INDEX

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, SET_TASK_TRACE function

The SET_TASK_TRACE function of the KEDD gate is used to store the value of the task trace flag in the current task's task table entry. A task table is a logical block of tasks, allocated together by the Kernel domain, and used to simplify the process of dynamically adding new tasks. Task tables are chained together, and vary in number.

Input Parameters

TRACE_TYPE

determines whether standard, special, or no tracing is required for this task.

Values for the parameter are:

SPECIAL
STANDARD

SUPPRESSED

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

DOMAIN_NAME_NOT_FOUND
DOMAIN_TOKEN_NOT_FOUND
DUPLICATE_DOMAIN_NAME
DUPLICATE_DOMAIN_TOKEN
DUPLICATE_GATE_INDEX

The following values are returned when RESPONSE is INVALID:

INVALID_DOMAIN_TOKEN
INVALID_ENTRY_POINT
INVALID_FUNCTION
INVALID_GATE_INDEX

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, SET_TRAP_OFF function

The SET_TRAP_OFF function of the KEDD gate is used to reset the kernel global trap point.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

DOMAIN_NAME_NOT_FOUND
DOMAIN_TOKEN_NOT_FOUND
DUPLICATE_DOMAIN_NAME
DUPLICATE_DOMAIN_TOKEN
DUPLICATE_GATE_INDEX

The following values are returned when RESPONSE is INVALID:

INVALID_DOMAIN_TOKEN
INVALID_ENTRY_POINT
INVALID_FUNCTION
INVALID_GATE_INDEX

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDD gate, SET_TRAP_ON function

The SET_TRAP_ON function of the KEDD gate is used to set a kernel global trap point.

Input Parameters

ENTRY_POINT

is the 31-bit address of the entry point for that domain, for example, A(X'80000000' + DFHSMDM) for storage manager domain.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

INVALID_ENTRY_POINT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, ABNORMALLY_TERMINATE_TASK function

The ABNORMALLY_TERMINATE_TASK function of the KEDS gate identifies the task which is to be abnormally terminated.

Input Parameters

DUMP

A binary value indicating whether CICS should take a memory dump when the task terminates.

Values for the parameter are:

DUMP_NO
DUMP_YES

RETRY

A binary value indicating whether the task should be retried.

Values for the parameter are:

RETRY_NO
RETRY_YES

TASK_TOKEN

identifies the task which is to be abnormally terminated.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

TERMINATE_FAILED

The following values are returned when RESPONSE is INVALID:

INVALID_TASK_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, ADD_CRITICAL_MODULE function

Adds the module address to the vector of modules in which a runaway condition will be deferred.

Input Parameters

MODULE_ADDR

The address of the module to be added to the vector.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

VECTOR_FULL

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, ADD_CRITICAL_WINDOW function

Adds the window address to the vector of windows in modules in which the Runaway condition will be deferred. Within such windows Runaway will not be deferred.

Input Parameters

WINDOW_END
The end address of the window.

WINDOW_START
The start address of the window.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
VECTOR_FULL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, CREATE_TASK function

The CREATE_TASK function of the KEDS gate is used to allocate a new executable task from the task table. A task table is a logical block of tasks, allocated together by the Kernel domain, and used to simplify the process of dynamically adding new tasks. Task tables are chained together, and vary in number.

Input Parameters

ALLOCATION
indicates whether or not the returned task should be allocated from those tasks pre-allocated for MXT.

Values for the parameter are:
DYNAMIC
STATIC

ATTACH_TOKEN
is the 31-bit token that uniquely identifies the request. This token is returned on the corresponding TASK_REPLY to identify the request.

TRANSACTION_BUILD
Optional Parameter

Values for the parameter are:
BUILD
TERMINATE

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
INQUIRE_ERROR
TRANSACTION_ABEND
TRANSACTION_ERROR_MSG

The following values are returned when RESPONSE is EXCEPTION:
ADD_TASK_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TASK_TOKEN
is the 31-bit token that uniquely identifies the newly created task.

KEDS gate, CREATE_TCB function

The CREATE_TCB function of the KEDS gate creates the default task for a new MVS TCB, and MVS posts the TCB to start execution. The default task is the task, associated with the TCB, that executes the dispatcher loop which chooses the next CICS task (system or non-system) to be dispatched, or if no CICS task is to be dispatched, issues an MVS WAIT.

Input Parameters

ATTACH_TOKEN
is the 31-bit token that uniquely identifies the request. This token is returned on the corresponding TASK_REPLY to identify the request.

ESSENTIAL_TCB
indicates whether CICS is to be terminated if a TCB in this mode has its ESTAE exit driven for a non recoverable error.

Values for the parameter are:
ESSENTIAL_NO
ESSENTIAL_YES

EXEC_CAPABLE
indicates whether support should be provided under the new TCB for CICS API commands.

Values for the parameter are:
EXEC_NO
EXEC_YES

INHERIT_SUBSPACE
indicates whether TCBs in this mode are to inherit the subspace of the attaching TCB.

Values for the parameter are:
INHERIT_NO
INHERIT_YES

LE_ENVIRONMENT
indicates whether CICS should tell Language Environment that it is running in a CICS environment under this TCB. If LE_CICS is specified, Language Environment will issue CICS API commands.

Values for the parameter are:
LE_CICS
LE_MVS

MODENAME
specifies the mode of the new TCB.

PARENT_MODENAME
identifies the mode of the TCB that is to ATTACH the new TCB.

PRTY_RELATIVE_TO_QR
gives the priority of this TCB relative to QR.

TCB_KEY
specifies the key to be specified on the ATTACH of TCBs in this mode. The value ends up in TCBPKF.

Values for the parameter are:
KEY8
KEY9

DEPENDENT_ON
Optional Parameter

specifies that the TCB is dependent on the named parent TCB mode.

PTHREAD
Optional Parameter

A binary value that indicates if a pthread is to be created.

Values for the parameter are:

NO
YES

SZERO

Optional Parameter

gives the value (YES or NO) of the SZERO parameter for the ATTACH request. If TCB_KEY(USERKEY) is specified, SZERO(NO) is assumed.

Values for the parameter are:

SZERO_NO
SZERO_YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

INQUIRE_ERROR
VECTOR_FULL

The following values are returned when RESPONSE is EXCEPTION:

ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
INVALID_CALLING_MODE

MVS_TCB_ADDRESS

The address of the newly created MVS TCB.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TASK_TOKEN

is the 31-bit token that uniquely identifies the newly created task.

KEDS gate, DETACH_TERMINATED_OWN_TCBS function

The DETACH_TERMINATED_OWN_TCBS function of the KEDS gate detaches any terminated TCBs which were attached by the TCB on which this function is invoked.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
VECTOR_FULL

The following values are returned when RESPONSE is EXCEPTION:

ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
INVALID_CALLING_MODE
NOT_FOUND
TCB_NOT_WAITING
TERMINATE_FAILED

The following values are returned when RESPONSE is INVALID:

```
INVALID_INPUT_COMB
INVALID_TASK_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, END_TASK function

The END_TASK function of the KEDS gate is used to free any resources that have been acquired by the kernel domain during the lifetime of the current task and need freeing before the end of the task.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
VECTOR_FULL
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
INVALID_CALLING_MODE
NOT_FOUND
TCB_NOT_WAITING
TERMINATE_FAILED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_INPUT_COMB
INVALID_TASK_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, FREE_TCBS function

The FREE_TCBS function of the KEDS gate conditionally frees control blocks, in collaboration with the Dispatcher for re-use, associated with any detached TCBs.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
VECTOR_FULL
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
```

FREE_TCBS_LOCK_HELD
INVALID_CALLING_MODE
NOT_FOUND
TASK_NOT_TERMINATED
TCB_NOT_WAITING
TERMINATE_FAILED

The following values are returned when RESPONSE is INVALID:

INVALID_INPUT_COMB
INVALID_TASK_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, FREE_TCBS_NOTIFY_IMMED function

The FREE_TCBS_NOTIFY_IMMED function of the KEDS gate immediately notifies the Kernel system task that it needs to call function FREE_TCBS.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, INQUIRE_MVSTCB function

Retrieve information about an MVS TCB.

Input Parameters

MVS_TCB_ADDRESS

The address of the TCB.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TCA_TASK_NUMBER

The task number.

TCB_ID

The TCB identifier.

KEDS gate, INQUIRE_TCB function

Retrieve the kernel task token for the current TCB.

Input Parameters

DEFAULT_TASK_TOKEN

The retrieved task token.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

CANNOT_ACCESS_TCB
TCB_NOT_WAITING

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, POP_TASK function

Given a TCB executing the current CICS task, the POP_TASK function of the KEDS gate is used to make it execute its default task instead.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
VECTOR_FULL
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
INVALID_CALLING_MODE
NOT_FOUND
TCB_NOT_WAITING
TERMINATE_FAILED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_INPUT_COMB
INVALID_TASK_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, PROCESS_KETA_ERROR function

The PROCESS_KETA_ERROR function of the KEDS gate is used to handle any errors for the DFHKETA module. (The DFHKETA module handles the performance sensitive KEDS functions, and calls the DFHKEDS module when its recovery routine is invoked.)

Input Parameters

ERROR_DATA

address of the error data that describes the error that has occurred in the DFHKETA module.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, PUSH_TASK function

Given a TCB executing its default task, the PUSH_TASK function of the KEDS gate is used to make it execute a CICS task instead.

Input Parameters

TASK_TOKEN

Identifies the task which is to be abnormally terminated.

Output Parameters

TASK_CPU_ON_CP_INT
Optional Parameter

The CPU time on standard CP used by the task.

TASK_CPU_OFFLD_ON_CP_INT
Optional Parameter

The CPU time on standard CP that was offload eligible used by the task.

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
VECTOR_FULL
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
INVALID_CALLING_MODE
NOT_FOUND
TCB_NOT_WAITING
TERMINATE_FAILED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_INPUT_COMB
INVALID_TASK_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TASK_CPU_INTERVAL
Optional Parameter

The CPU time used by the task.

KEDS gate, RESTORE_STIMER function

The RESTORE_STIMER function of the KEDS gate is used to restore the kernel's STIMER exit after MVS requests that use the MVS STIMER macro internally.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
VECTOR_FULL
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
INVALID_CALLING_MODE
```

NOT_FOUND
TCB_NOT_WAITING
TERMINATE_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_INPUT_COMB
INVALID_TASK_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, SEND_DEFERRED_ABEND function

The SEND_DEFERRED_ABEND function of the KEDS gate is used by the transaction manager to implement the deferred purge function. If a purge request is made against a task that is not in a suitable state to be purged, this function defers the abend of that task until the task is no longer protected against purge.

Input Parameters

ERROR_CODE
The abend code that CICS issues when the task is eventually purged.

DS_TASK_TOKEN
Optional Parameter

is the 31-bit dispatcher token that identifies the CICS task to be abended. If not supplied, DS_TASK_TOKEN defaults to the current task.

FORCE
Optional Parameter

indicates whether or not the deferred abend is to be forced.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
VECTOR_FULL

The following values are returned when RESPONSE is EXCEPTION:
ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
INVALID_CALLING_MODE
NOT_FOUND
TCB_NOT_WAITING
TERMINATE_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_INPUT_COMB
INVALID_TASK_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, START_FORCE_PURGE_PROTECT function

The START_PURGE_PROTECTION function of the KEDS gate is used to inhibit force-purge for the current task.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, START_PURGE_PROTECTION function

The START_PURGE_PROTECTION function of the KEDS gate is used to inhibit purge, but not force-purge, for the current task.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, START_RUNAWAY_TIMER function

The START_RUNAWAY_TIMER function of the KEDS gate is used to resume runaway timing for the current task. This reduces the stop runaway count by one. The timer is resumed only when all outstanding STOP_RUNAWAY_TIMER requests have been canceled.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
VECTOR_FULL
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
INVALID_CALLING_MODE
NOT_FOUND
TCB_NOT_WAITING
TERMINATE_FAILED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_INPUT_COMB
INVALID_TASK_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, STOP_FORCE_PURGE_PROTECT function

The STOP_FORCE_PURGE_PROTECTION function of the KEDS gate is used to enable again force purge for the current task after force purge has been suspended by a previous START_FORCE_ PURGE_PROTECTION function call.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, STOP_PURGE_PROTECTION function

The STOP_PURGE_PROTECTION function of the KEDS gate is used to enable again purge for the current task after purge has been suspended by a previous START_PURGE_PROTECTION function call.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, STOP_RUNAWAY_TIMER function

The STOP_RUNAWAY_TIMER function of the KEDS gate is used to inhibit runaway detection for the current task. The remaining runaway interval is preserved until a START_RUNAWAY_TIMER request is issued. The stop runaway count is incremented by one; this allows STOP_RUNAWAY_TIMER requests to be nested.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
VECTOR_FULL
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
INVALID_CALLING_MODE
NOT_FOUND
TCB_NOT_WAITING
TERMINATE_FAILED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_INPUT_COMB
INVALID_TASK_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEGD gate, INQUIRE_KERNEL function

The INQUIRE_KERNEL function of the KEGD gate is used to obtain the global data maintained by the kernel.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ALTERNATE_XRF_IDS

Optional Parameter

is the 8-character name of the recoverable service table used if the CICS region is running with XRF and DBCTL.

CICS_SVC_NUMBER

Optional Parameter

is the 8-bit CICS service SVC number.

DUMP_RETRY_TIME

Optional Parameter

specifies the total time that CICS is to continue trying to obtain a system dump using the SDUMP macro.

GENERIC_APPLID

Optional Parameter

is the 8-character generic applid that identifies the active and alternate CICS systems to z/OS Communications Server in an XRF environment.

HPO

Optional Parameter

specifies whether CICS is to use the z/OS Communications Server high performance option.

Values for the parameter are:

```
NO
YES
```

ISC

Optional Parameter

specifies whether ISC support is included in this CICS region.

Values for the parameter are:

```
NO
YES
```

OP_REL

Optional Parameter

is the 2-byte operating system release and modification level.

OP_SYS

Optional Parameter

is the 1-character operating system identifier, for example, 'B' = MVS.

OP_VER

Optional Parameter

is the 1-byte operating system version.

OS_PARMS

Optional Parameter

is the 8-byte block containing the 31-bit address and 31-bit length of the MVS parameters.

SIT_NAME
Optional Parameter

is the 8-character SIT name.

SPECIFIC_APPLID
Optional Parameter

is the 8-character specific applid that identifies the CICS system in the z/OS Communications Server network.

SYSID
Optional Parameter

is the 4-character ZCP system entry name.

SYSTEM_RUNAWAY_LIMIT
Optional Parameter

the ICVR time to be used by all tasks that have been defined to have the default runaway limit in the system.

USS_PROCESS
Optional Parameter

specifies whether the kernel successfully issued a Unix System Services **SET_DUB_DEFAULT DUBPROCESS** command during CICS initialization.

Values for the parameter are:
NO
YES

XRF
Optional Parameter

specifies whether ISC support is included in this CICS region.

Values for the parameter are:
NO
YES

XRF_COMMAND_LIST
Optional Parameter

is the 8-character name of the command list table used by the XRF alternate CICS region.

KEGD gate, SET_KERNEL function

The SET_KERNEL function of the KEGD gate is used to change the global data maintained by the kernel.

Input Parameters

ALTERNATE_XRF_IDS
Optional Parameter

is the 8-character name of the recoverable service table used if the CICS region is running with XRF and DBCTL.

CICS_SVC_NUMBER
Optional Parameter

is the 8-bit CICS service SVC number.

DUMP_RETRY_TIME
Optional Parameter

specifies the total time that CICS is to continue trying to obtain a system dump using the SDUMP macro.

GENERIC_APPLID
Optional Parameter

is the 8-character generic applid that identifies the active and alternate CICS systems to z/OS Communications Server in an XRF environment.

HPO
Optional Parameter

specifies whether CICS is to use the z/OS Communications Server high performance option.

Values for the parameter are:
NO
YES

ISC
Optional Parameter

specifies whether ISC support is included in this CICS region.

Values for the parameter are:
NO
YES

SIT_NAME
Optional Parameter

is the 8-character name of the system initialization table.

SPECIFIC_APPLID
Optional Parameter

is the 8-character specific applid that identifies the CICS system in the z/OS Communications Server network.

SYSID
Optional Parameter

is the 4-character ZCP system entry name.

SYSTEM_RUNAWAY_LIMIT
Optional Parameter

the ICVR time to be used by all tasks that have been defined to have the default runaway limit in the system.

TERMINATE_FO
Optional Parameter

specifies whether the FO TCB can be normally terminated on an immediate shutdown.

Values for the parameter are:
NO
YES

XRF
Optional Parameter

specifies whether XRF support is included in the CICS region.

Values for the parameter are:
NO
YES

XRF_COMMAND_LIST
Optional Parameter

is the 8-character name of the command list table used by the XRF alternate CICS region.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

WRONG_SVC_NUMBER

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KETI gate, ADJUST_STCK_TO_LOCAL function

Perform local time adjustment on a STCK value

Input Parameters

GMT_STCK

The STCK value to be adjusted.

Output Parameters

LOCAL_STCK

The adjusted STCK value.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KETI gate, CONVERT_TO_DECIMAL_TIME function

The CONVERT_TO_DECIMAL_TIME function of the KETI gate is used to convert dates and times in the internal store clock (STCK) format to decimal format.

Input Parameters

STCK_TIME

is a doubleword containing a date and time in STCK format.

LOCAL_ADJUST

Optional Parameter

Specifies whether to adjust the STCK value to local time.

Values for the parameter are:

NO
YES

Output Parameters

DECIMAL_DATE

is an 8-character date in the format determined by FULL_DATE_FORMAT.

DECIMAL_MICROSECONDS

is the 6-character microseconds portion of DECIMAL_TIME.

DECIMAL_TIME

is the current local decimal time in the format HHMMSS.

FULL_DATE_FORMAT

is the current full date format determined by the default date format of the timer domain.

Values for the parameter are:

DDMMYYYY
MMDDYYYY
YYYYMMDD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KETI gate, CONVERT_TO_STCK_FORMAT function

The CONVERT_TO_STCK_FORMAT function of the KETI gate is used to convert times and dates to STCK format.

Input Parameters

DECIMAL_TIME
is the current local decimal time in the format HHMMSS.

DECIMAL_DATE
Optional Parameter

is an optional 8-character date in the format determined either by FULL_DATE_FORMAT or by the default for the timer domain if FULL_DATE_FORMAT is omitted.

FULL_DATE_FORMAT
Optional Parameter

is the current full date format.

Values for the parameter are:
```
DDMMYYYY
MMDDYYYY
YYYYMMDD
```

INSTANCE
Optional Parameter

is required only if DECIMAL_DATE is omitted.

Values for the parameter are:
```
LAST
NEXT
TODAY
```

LOCAL_ADJUST
Optional Parameter

Specifies whether to apply a local time adjustment.

Values for the parameter are:
```
NO
YES
```

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STCK_TIME
is a doubleword containing the STCK value corresponding to the input local time.

KETI gate, INQ_LOCAL_DATETIME_DECIMAL function

The INQ_LOCAL_DATETIME_DECIMAL function of the KETI gate is used to return the local date, and the local time in decimal format.

Output Parameters

DECIMAL_DATE
is an 8-character date in the format determined by FULL_DATE_FORMAT.

DECIMAL_MICROSECONDS
is the 6-character microseconds portion of DECIMAL_TIME.

DECIMAL_TIME
is the current local decimal time in the format HHMMSS.

FULL_DATE_FORMAT
is the current full date format determined by the default date format of the timer domain.

Values for the parameter are:
DDMMYYYY
MMDDYYYY
YYYYMMDD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KETI gate, INQUIRE_DATE_FORMAT function

The INQUIRE_DATE_FORMAT function of the KETI gate is used to return the current date format.

Output Parameters

DATE_FORMAT
is the current default date format for the timer domain.

Values for the parameter are:
DDMMYY
MMDDYY
YYMMDD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KETI gate, REQUEST_NOTIFY_OF_A_RESET function

The REQUEST_NOTIFY_OF_A_RESET function of the KETI gate requests a shoulder tap from KETI whenever the local time is reset.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KETI gate, RESET_LOCAL_TIME function

The RESET_LOCAL_TIME function of the KETI gate is used by the AP domain to inform KETI that a local time reset has occurred.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KETI gate, SET_DATE_FORMAT function

The SET_DATE_FORMAT function of the KETI gate is used to set the date format for the timer domain.

Input Parameters

DATE_FORMAT
is the format to be set as the default for the timer domain.

Values for the parameter are:
DDMMYY
MMDDYY
YYMMDD

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEXM gate, TRANSACTION_INITIALISATION function

The TRANSACTION_INITIALISATION function of the KEXM gate performs kernel initialization during XM task-reply.

Input Parameters

TRANSACTION_TOKEN
A token identifying the transaction for which kernel initialization is to be performed.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Kernel domain's generic formats

Table 25 describes the generic formats owned by the domain and shows the functions performed on the calls.

Table 25. Kernel domain's generic formats

Format	Calling module	Function
KEDS	DFHKETA	TASK_REPLY
	DFHKETCB	TCB_REPLY
KETI	DFHKETI	NOTIFY_RESET

Note: In the descriptions of the formats, the input parameters are input not to the Kernel domain, but to the domain being called by the Kernel domain. Similarly, the output parameters are output by the domain that was called by the Kernel domain, in response to the call.

KEDS gate, TASK_REPLY function

The TASK_REPLY function of the KEDS format is issued by the kernel to the issuer of CREATE_TASK, under the new task.

Input Parameters

ATTACH_TOKEN
is the 31-bit token that uniquely identifies the corresponding CREATE_TASK request.

TASK_TOKEN
is the 31-bit token that uniquely identifies the new task.
TRANSACTION_BUILD
Optional Parameter

Values for the parameter are:
BUILD
TERMINATE

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
TRANSACTION_ABEND
TRANSACTION_ERROR_MSG
VECTOR_FULL

The following values are returned when RESPONSE is EXCEPTION:
ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
INVALID_CALLING_MODE
NOT_FOUND
TCB_NOT_WAITING
TERMINATE_FAILED

The following values are returned when RESPONSE is INVALID:
INVALID_INPUT_COMB
INVALID_TASK_TOKEN
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEDS gate, TCB_REPLY function

The TCB_REPLY function of the KEDS format is issued by the kernel to the issuer of CREATE_TCB, under the new TCB's default task.

Input Parameters

ATTACH_TOKEN
is the 31-bit token that uniquely identifies the corresponding CREATE_TCB request.
TASK_TOKEN
is the 31-bit token that uniquely identifies the new TCB's task.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
INQUIRE_ERROR
INVALID_FUNCTION
LOOP
VECTOR_FULL

The following values are returned when RESPONSE is EXCEPTION:

```
ADD_KTCB_ERROR
ADD_TASK_ERROR
ATTACH_KTCB_ERROR
CANNOT_ACCESS_TCB
DEFERRED_ABEND_NOT_SENT
INVALID_CALLING_MODE
NOT_FOUND
TCB_NOT_WAITING
TERMINATE_FAILED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_INPUT_COMB
INVALID_TASK_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KETI gate, NOTIFY_RESET function

The NOTIFY_RESET function of the KETI format is used by KETI itself to inform domains that a RESET has occurred.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHKEAR	Implements KEAR service requests.
DFHKEDCL	Implements domain call requests.
DFHKEDD	Services KEDD-format requests.
DFHKEDRT	Implements domain return requests.
DFHKEDS	Services KEDS-format requests.
DFHKEDUF	Offline dump formatting routine to format the kernel domain control blocks.
DFHKEEDA	Handles deferred abends
DFHKEGD	Services KEGD-format requests.
DFHKEIN	Implements kernel domain initialization.
DFHKELCL	Implements LIFO Push.
DFHKELOC	Offline dump formatting routine to locate the kernel domain anchor blocks.
DFHKELRT	Implements LIFO Pop.
DFHKERCD	Constructs the kernel domain error data for error handling routines.
DFHKERER	Updates the kernel domain error table for error handling routines.
DFHKERET	Implements RESET_ADDRESS requests.
DFHKERKE	Handles KERNERROR responses for domain call requests which cannot handle them.
DFHKERPC	Implements recovery percolation both from RECOVERY_PERCOLATE requests and also other recovery events that, because of the existing environment, must be percolated.
DFHKERRI	Responsible for passing control to a recovery routine.
DFHKERRQ	Implements RECOVERY_REQUEST requests.

Module	Function
DFHKERRU	Implements runaway task error handling.
DFHKERRX	Implements RECOVERY_EXIT requests.
DFHKESCL	Implements subroutine call requests.
DFHKESFM	Handles freeing of stack segments.
DFHKESGM	Handles allocation of new stack segments.
DFHKESIP	Receives control from and returns control to MVS.
DFHKESRT	Implements subroutine return requests.
DFHKESTX	The CICS ESTAE exit which passes control to the appropriate level of recovery routine.
DFHKESVC	Provides authorized services for kernel domain functions.
DFHKETA	Implements KEDS CREATE_TASK requests.
DFHKETCB	Receives control from MVS for a kernel domain TCB.
DFHKETI	Provides service time functions at the KETI gate.
DFHKETIX	Performs task CPU monitoring functions and task runaway detection.
DFHKETRI	Offline trace formatting routine for kernel domain trace entries.
DFHKETXR	Allows an attaching TCB to detemine that a TCB (but not a specific TCB) which it attached, has terminated. This allows for the possibility of initiating a more timely detach of TCBs which have terminated normally, and to detect TCBs which have prematurely terminated.
DFHKEXM	Implements KEXM_FORMAT requests.

Loader domain (LD)

The loader domain is used to obtain access to storage-resident copies of nucleus and application programs, maps, and tables. The loader domain uses the operating system interfaces to load programs into the CICS dynamic storage areas (DSAs), and to scan the link pack area (LPA).

Loader domain's specific gates

The specific gates provide access for other domains to functions that are provided by the LD domain.

LDLB gate, ADD_REPLACE_LIBRARY function

The ADD_REPLACE_LIBRARY function of the LDLB gate is used to install a new LIBRARY resource into the CICS system, or to replace an installed disabled LIBRARY resource of the same name.

Input Parameters

APPLICATION_ID
Optional parameter

The 64-character name of the application.

APPL_MAJOR_VER
Optional parameter

The fullword binary value that is the major version of the application.

APPL_MICRO_VER
Optional parameter

The fullword binary value that is the micro version of the application.

APPL_MINOR_VER
Optional parameter

The fullword binary value that is the minor version of the application.

CRITICAL
Optional parameter

Whether the LIBRARY is to be installed as critical (must be available at CICS startup) or non-critical (does not have to be available at CICS startup).

Values for the parameter are:
```
CRITICAL_YES
CRITICAL_NO
```

DSNAME01
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME02
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME03
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME04
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME05
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME06
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME07
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME08
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME09
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME10
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME11
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME12
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME13
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME14
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME15
Optional Parameter

The name of a data set in the LIBRARY concatenation.

DSNAME16
Optional Parameter

The name of a data set in the LIBRARY concatenation.

ENABLE_STATUS
Optional Parameter

Whether the LIBRARY is to be installed as enabled (participates in the search order) or disabled (does not participate in the search order).

Values for the parameter are:

```
DISABLED
ENABLED
```

LIBRARY_NAME
The name of the LIBRARY to be installed or replaced.

PLATFORM_ID
Optional parameter

The 64-character name of the platform.

RANKING
Optional Parameter

The ranking value to be assigned to this LIBRARY, which is used to determine its position within the search order.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
CATALOG_WRITE_FAILED
CATALOG_DELETE_FAILED
LIBRARY_LOCK_ERROR
LIBRARY_NAME_ERROR
LIBRARY_CHAIN_ERROR
LOOP
DSNAME_ARRAY_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
INSUFFICIENT_STORAGE
ALLOCATE_FAILED_ENABLE
CONCATENATE_FAILED_ENABLE
OPEN_FAILED_ENABLE
NOT_DISABLED
SECURITY_FAILURE
USERID_NOTAUTHED
MVS_ABEND_CONDITION
SERIOUS MVS ABEND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETERS
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLB gate, DISCARD_LIBRARY function

The DISCARD_LIBRARY function of the LDLB gate is used to remove a LIBRARY resource from the CICS system.

Input Parameters

APPLICATION_ID
Optional parameter

The 64-character name of the application.

APPL_MAJOR_VER
Optional parameter

The fullword binary value that is the major version of the application.

APPL_MICRO_VER
Optional parameter

The fullword binary value that is the micro version of the application.

APPL_MINOR_VER
Optional parameter

The fullword binary value that is the minor version of the application.

LIBRARY_NAME
The name of the LIBRARY to be discarded.

PLATFORM_ID
Optional parameter

The 64-character name of the platform.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
CATALOG_DELETE_FAILED
LIBRARY_LOCK_ERROR
LIBRARY_NAME_ERROR
LIBRARY_CHAIN_ERROR
LOOP
DSNAME_ARRAY_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
LIBRARY_NOT_FOUND
NOT_DISABLED
CLOSE_FAILED
DECONCATENATE_FAILED
UNALLOCATE_FAILED
LIBRARY_DELETE_ERROR
MVS_ABEND_CONDITION
SERIOUS MVS ABEND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLB gate, END_BROWSE_LIBRARY function

The END_BROWSE_LIBRARY function of the LDLB gate is used to end a browse session of the LIBRARY resources installed in the CICS system.

Input Parameters

BROWSE_TOKEN
is a token which identifies this browse session of LIBRARY resources.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLB gate, GET_NEXT_APPLICATION function

The GET_NEXT_APPLICATION function of the LDLB gate is used to get information about the next application. The browse session is managed by the CURRENT_TOKEN and NEW_TOKEN parameters.

Input Parameters

APPLICATION_ID
Optional parameter

Specifies buffer storage in which the application name is to be returned.

CURRENT_TOKEN
A token which identifies the current position of the browse through the applications. This value should be zero for a new browse and replaced by the returned NEW_TOKEN parameter value on subsequent requests.

PLATFORM_ID
Optional parameter

Specifies buffer storage in which the platform name is to be returned.

Output Parameters

APPL_MAJOR_VER
Optional parameter

Specifies the major version number for the application.

APPL_MICRO_VER
Optional parameter

Specifies the micro version number for the application.

APPL_MINOR_VER
Optional parameter

Specifies the minor version number for the application.

APPL_TOKEN
Optional parameter

Specifies the application token.

NEW_TOKEN
A token which identifies the next position of the browse through the applications. This value should be used as the value for the CURRENT_TOKEN input parameter on subsequent requests.

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
IAE_LOCK_ERROR
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:
```
NO_MORE_DATA_AVAILABLE
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLB gate, GET_NEXT_LIBRARY function

The GET_NEXT_LIBRARY function of the LDLB gate is used to get information about the next LIBRARY in the current browse session of LIBRARY resources currently installed in the CICS system. The browse is in ranking order, and in install-time order within ranking.

Input Parameters

BROWSE_TOKEN
is a token which identifies this browse of LIBRARY resources.

LIBRARY_DSNAMES
Optional parameter

specifies buffer storage in which the list of all data sets within the LIBRARY is to be returned.

Output Parameters

LIBRARY_NAME
returns the name of the next LIBRARY in the browse of LIBRARY resources.

CRITICAL
Optional parameter

specifies whether the LIBRARY is to be installed as critical (must be available at CICS startup) or non-critical (does not have to be available at CICS startup).

Values for the parameter are:
```
CRITICAL_YES
CRITICAL_NO
```

DSNAME01
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME02
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME03
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME04
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME05
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME06
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME07
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME08
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME09
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME10
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME11
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME12
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME13
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME14
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME15
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME16
Optional Parameter

returns the name of a data set in the LIBRARY concatenation. This name can be blank.

ENABLE_STATUS
Optional Parameter

returns a value which indicates whether the LIBRARY is currently enabled (participates in the search order) or disabled (does not participate in the search order)

Values for the parameter are:
DISABLED
ENABLED

RANKING
Optional Parameter

returns the ranking value currently assigned to this LIBRARY, which is used to determine its position within the search order.

SEARCH_POSITION
Optional Parameter

returns the actual current position of this LIBRARY in the overall LIBRARY search order (zero if the LIBRARY is disabled).

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
LIBRARY_NOT_FOUND
NO_MORE_DATA_AVAILABLE
BUFFER_TOO_SMALL

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLB gate, INQUIRE_LIBRARY function

The INQUIRE_LIBRARY function of the LDLB gate is used to get information about the specified LIBRARY.

Input Parameters

APPLICATION_ID
Optional parameter

The 64-character name of the application.

APPL_MAJOR_VER
Optional parameter

The fullword binary value that is the major version of the application.

APPL_MICRO_VER
Optional parameter

The fullword binary value that is the micro version of the application.

APPL_MINOR_VER
Optional parameter

The fullword binary value that is the minor version of the application.

LIBRARY_NAME
The name of the required LIBRARY.

LIBRARY_DSNAMES
Optional parameter

Buffer storage in which the list of all data sets in the LIBRARY is to be returned.

PLATFORM_ID
Optional parameter

The 64-character name of the platform.

Output Parameters

CRITICAL
Optional parameter

Whether the LIBRARY is defined as critical (must be available at CICS startup) or non-critical (does not have to be available at CICS startup).

Values for the parameter are:

```
CRITICAL_YES
CRITICAL_NO
```

DSNAME01
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME02
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME03
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME04
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME05
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME06
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME07
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME08
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME09
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME10
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME11
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME12
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME13
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME14
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME15
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

DSNAME16
Optional Parameter

The name of a data set in the LIBRARY concatenation. This name can be blank.

ENABLE_STATUS
Optional Parameter

Indicates whether the LIBRARY is currently enabled (participates in the search order) or disabled (does not participate in the search order).

Values for the parameter are:
```
DISABLED
ENABLED
```

RANKING
Optional Parameter

The ranking value currently assigned to this LIBRARY, which is used to determine its position within the search order.

RESOURCE_SIGNATURE
Optional Parameter

A buffer to receive the LIBRARY resource signature.

SEARCH_POSITION
Optional Parameter

The actual current position of this LIBRARY in the overall LIBRARY search order (zero if the LIBRARY is disabled).

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:
```
LIBRARY_NOT_FOUND
BUFFER_TOO_SMALL
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLB gate, LOG_LIBRARY_ORDER function

The LOG_LIBRARY_ORDER function of the LDLB gate is used to log the current configuration of installed enabled LIBRARY resources in the CICS system as part of an audit trail.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLB gate, SET_LIBRARY function

The SET_LIBRARY function of the LDLB gate is used to set attributes of the specified LIBRARY. The specified LIBRARY must be installed in the CICS system.

Input Parameters

APPLICATION_ID
Optional parameter

The 64-character name of the application.

APPL_MAJOR_VER
Optional parameter

The fullword binary value that is the major version of the application.

APPL_MICRO_VER
Optional parameter

The fullword binary value that is the micro version of the application.

APPL_MINOR_VER
Optional parameter

The fullword binary value that is the minor version of the application.

CRITICAL
Optional parameter

Whether the LIBRARY is defined as critical (must be available at CICS startup) or non-critical (does not have to be available at CICS startup).

Values for the parameter are:

```
CRITICAL_YES
CRITICAL_NO
```

ENABLE_STATUS
Optional Parameter

Whether the LIBRARY is to be enabled (participates in the search order) or disabled (does not participate in the search order).

Values for the parameter are:

```
DISABLED
ENABLED
```

LIBRARY_NAME
The name of the LIBRARY to be updated.

PLATFORM_ID
Optional parameter

The 64-character name of the platform.

RANKING
Optional Parameter

The ranking value to be assigned to this LIBRARY, which is used to determine its position within the search order.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
CATALOG_WRITE_FAILED
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
LIBRARY_NOT_FOUND
ALLOCATE_FAILED_ENABLE
CONCATENATE_FAILED_ENABLE
OPEN_FAILED_ENABLE
CLOSE_FAILED
DECONCATENATE_FAILED
UNALLOCATE_FAILED
MVS_ABEND_CONDITION
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLB gate, START_BROWSE_LIBRARY function

The START_BROWSE_LIBRARY function of the LDLB gate is used to start a browse session through the LIBRARY resources currently installed in the CICS system. It is used to obtain a browse token for use with a subsequent GET_NEXT_LIBRARY or END_BROWSE_LIBRARY call.

Output Parameters

BROWSE_TOKEN
returns a token used to refer to this browse session on subsequent LIBRARY browse requests

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLD gate, ACQUIRE_PROGRAM function

The ACQUIRE_PROGRAM function of the LDLD gate is used to obtain the entry point and load point addresses and the length of a usable copy of the named program. The program must previously have been identified to the system in a DEFINE request, either during this session or in a previous session, if the catalog is in use.

Input Parameters

APPL_CONTEXT_TOKEN
Optional parameter

A token representing the application context of **PLATFORM_ID**, **APPLICATION_ID**, **APPL_MAJOR_VER**, **APPL_MINOR_VER**, and **APPL_MICRO_VER**.

APPLICATION_ID
Optional parameter

The 64-character name of the application associated with the task.

APPL_MAJOR_VER
Optional parameter

The fullword binary value that is the major version of the application associated with the task.

APPL_MINOR_VER
Optional parameter

The fullword binary value that is the minor version of the application associated with the task.

APPL_MICRO_VER
Optional parameter

The fullword binary value that is the micro version of the application associated with the task.

PLATFORM_ID
Optional parameter

The 64-character name of the platform associated with the task.

PROGRAM_NAME
The name of the required program.

PROGRAM_TOKEN
A valid program-identifying token as returned by a previous DEFINE or ACQUIRE request for the same program name.

SUSPEND
Optional Parameter

Whether the caller expects to receive control with an exception response if the loader encounters a shortage of virtual storage, or other transient error conditions. If there is insufficient storage to satisfy the request, SUSPEND(YES) causes the caller to be suspended until the request can be satisfied, and SUSPEND(NO) causes an exception response (reason NO_STORAGE) to be returned to the caller.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LIBRARY_IO_ERROR
```

LOOP
OS_STORAGE_SHORTAGE

The following values are returned when RESPONSE is EXCEPTION:
NO_STORAGE

The following values are returned when RESPONSE is EXCEPTION:
PROGRAM_NOT_DEFINED
PROGRAM_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_PROGRAM_TOKEN

RESPONSE
Whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

COPY_STATUS
Optional Parameter

Whether this request resulted in a physical load of the program into storage, and is used by the program manager domain to recognize that a COBOL program requires initialization.

Values for the parameter are:
NEW_COPY
OLD_COPY

ENTRY_POINT
The address of the entry point of the program instance.

FETCH_TIME
Optional Parameter

The time taken to load the program from the DFHRPL or dynamic LIBRARY concatenation. This is represented as the middle 4 bytes of a doubleword stored clock (STCK) value. If the acquired program resides in the MVS link pack area (LPA) or has already been loaded into one of the CICS dynamic storage areas (DSAs), the returned value is zero.

LOAD_POINT
Optional Parameter

The address of the load point of the program instance.

LOCATION
Optional Parameter

Determines where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

Values for the parameter are:
CDSA
ECDSA
ELPA
ERDSA
ESDSA
LPA
NONE
RDSA
SDSA

NEW_APPL_CONTEXT_TOKEN
Optional parameter

A token to be used on subsequent calls, representing the application context of **PLATFORM_ID**, **APPLICATION_ID**, **APPL_MAJOR_VER**, **APPL_MINOR_VER**, and **APPL_MICRO_VER**.

NEW_PROGRAM_TOKEN
Optional Parameter

The identifying token that may be used on subsequent ACQUIRE or RELEASE calls for this program name.

PROGRAM_ATTRIBUTE
Optional Parameter

Reflects the program attribute from the program definition, and is used by the program manager domain to recognize RELOAD programs.

Values for the parameter are:
RELOAD
RESIDENT
REUSABLE
TEST
TRANSIENT

PROGRAM_LENGTH
Optional Parameter

The length of the program instance in bytes.

LDLD gate, CATALOG_PROGRAMS function

The CATALOG_PROGRAMS function of the LDLD gate is used at the end of CICS initialization to request the loader domain to catalog all the program definitions that need cataloging. The call is issued by the DFHSIJ1 module.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
CATALOG_ERROR
CATALOG_NOT_OPERATIONAL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLD gate, CONVERT_NAME function

Obtain the primary member name for a long alias name from the cache if known, otherwise from the DFHRPL or dynamic LIBRARY concatenation.

Input Parameters

LONG_NAME
Optional Parameter

the alias name to be converted.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
LIBRARY_IO_ERROR
NO_STORAGE
OS_STORAGE_SHORTAGE

PROGRAM_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PROGRAM_NAME
Optional Parameter

The primary member name corresponding to the alias name.

LDLD gate, DEFINE_PROGRAM function

The DEFINE_PROGRAM function of the LDLD gate is used to introduce a new program to the CICS system or to update the details of an existing program.

Input Parameters

APPL_CONTEXT_TOKEN
Optional parameter

A token representing the application context of **PLATFORM_ID**, **APPLICATION_ID**, **APPL_MAJOR_VER**, **APPL_MINOR_VER**, and **APPL_MICRO_VER**.

APPLICATION_ID
Optional parameter

The 64-character name of the application associated with the task.

APPL_MAJOR_VER
Optional parameter

The fullword binary value that is the major version of the application associated with the task.

APPL_MINOR_VER
Optional parameter

The fullword binary value that is the minor version of the application associated with the task.

APPL_MICRO_VER
Optional parameter

The fullword binary value that is the micro version of the application associated with the task.

CATALOG_MODULE
Optional Parameter

indicates whether the program definition should be written to one of the catalogs.

Values for the parameter are:
NO
YES

EXECUTION_KEY
Optional Parameter

is the execution key for the program. This is used to determine which DSA the program instance resides in.

Values for the parameter are:
CICS
USER

PLATFORM_ID
Optional parameter

The 64-character name of the platform associated with the task.

PROGRAM_NAME
specifies the name of the required program.

PROGRAM_ATTRIBUTE
Optional Parameter

is a residency attribute to be associated with the program.

Values for the parameter are:
RELOAD
RESIDENT
REUSABLE
TEST
TRANSIENT

PROGRAM_TYPE
Optional Parameter

is the type of program copy to be used.

Values for the parameter are:
PRIVATE
SHARED
TYPE_ANY

PROGRAM_USAGE
Optional Parameter

defines whether the program is part of the CICS nucleus, or is an application program defined by the user. This determines whether the program definition is written to the local catalog or to the global catalog.

Values for the parameter are:
APPLICATION
NUCLEUS

REQUIRED_AMODE
Optional Parameter

is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded.

Values for the parameter are:
AMODE_ANY
24
31
64

REQUIRED_RMODE
Optional Parameter

is the residency mode required by CICS for the program. A program that does not have the required mode requirements is not loaded.

Values for the parameter are:
RMODE_ANY
24

UPDATE
Optional Parameter

indicates whether the loader domain should update the program definition if the loader domain already has a program definition for the program. If UPDATE(NO) is specified, and the loader domain already has a program definition for the specified program, PROGRAM_ALREADY_DEFINED is returned.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
CATALOG_ERROR
CATALOG_NOT_OPERATIONAL
INVALID_PROGRAM_NAME
PROGRAM_ALREADY_DEFINED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_MODE_COMBINATION
INVALID_TYPE_ATTRIB_COMBIN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NEW_APPL_CONTEXT_TOKEN

Optional parameter

A token to be used on subsequent calls, representing the application context of **PLATFORM_ID**, **APPLICATION_ID**, **APPL_MAJOR_VER**, **APPL_MINOR_VER**, and **APPL_MICRO_VER**.

NEW_PROGRAM_TOKEN

Optional Parameter

is the identifying token that may be used on subsequent ACQUIRE or RELEASE calls for this program name.

LDLD gate, DELETE_PROGRAM function

The DELETE_PROGRAM function of the LDLD gate is used to remove a program from the CICS system. All subsequent ACQUIRE requests for the named program fail with a reason of PROGRAM_NOT_DEFINED. Any instance of the program in use at the time the DELETE is received continue to exist until a RELEASE request reduces the use count to zero, at which time the instance is removed from memory.

Input Parameters

APPL_CONTEXT_TOKEN

Optional parameter

A token representing the application context of **PLATFORM_ID**, **APPLICATION_ID**, **APPL_MAJOR_VER**, **APPL_MINOR_VER**, and **APPL_MICRO_VER**.

APPLICATION_ID

Optional parameter

The 64-character name of the application associated with the task.

APPL_MAJOR_VER

Optional parameter

The fullword binary value that is the major version of the application associated with the task.

APPL_MINOR_VER

Optional parameter

The fullword binary value that is the minor version of the application associated with the task.

APPL_MICRO_VER

Optional parameter

The fullword binary value that is the micro version of the application associated with the task.

PLATFORM_ID
Optional parameter

The 64-character name of the platform associated with the task.

PROGRAM_NAME
specifies the name of the required program.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
PROGRAM_NOT_DEFINED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NEW_APPL_CONTEXT_TOKEN
Optional parameter

A token to be used on subsequent calls, representing the application context of **PLATFORM_ID**, **APPLICATION_ID**, **APPL_MAJOR_VER**, **APPL_MINOR_VER**, and **APPL_MICRO_VER**.

LDLD gate, END_BROWSE function

The END_BROWSE function of the LDLD gate is used to end a browse session.

Input Parameters

BROWSE_TOKEN
is a valid browse token as returned by the preceding START_BROWSE request.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BROWSE_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLD gate, GET_NEXT_INSTANCE function

The GET_NEXT_INSTANCE function of the LDLD gate is used to browse the current program instances in ascending load point address sequence.

Input Parameters

BROWSE_TOKEN
is a valid browse token as returned by the preceding START_BROWSE request.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
```

LOOP

The following values are returned when RESPONSE is EXCEPTION:
END_LIST

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESS
Optional Parameter

is the type of storage that the program resides in.

Values for the parameter are:
CICS
NONE
READ_ONLY
USER

ENTRY_POINT
Optional Parameter

is the address of the entry point of the program instance.

EXECUTION_KEY
Optional Parameter

is the execution key for the program.

Values for the parameter are:
CICS
USER

INSTANCE_USE_COUNT
Optional Parameter

is the current number of users of this instance.

LOAD_POINT
Optional Parameter

is the address of the load point of the program instance.

LOCATION
Optional Parameter

determines where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

Values for the parameter are:
CDSA
ECDSA
ELPA
ERDSA
ESDSA
LPA
NONE
RDSA
SDSA

PROGRAM_ATTRIBUTE
Optional Parameter

reflects the program attribute from the program definition, and is used by the program manager domain to recognize RELOAD programs.

Values for the parameter are:
RELOAD
RESIDENT
REUSABLE
TEST
TRANSIENT

PROGRAM_LENGTH
Optional Parameter

is the length of the program instance in bytes.

PROGRAM_NAME
Optional Parameter

is the name of the program whose attributes have been returned.

PROGRAM_TYPE
Optional Parameter

is the current program copy type.

Values for the parameter are:
PRIVATE
SHARED
TYPE_ANY

PROGRAM_USAGE
Optional Parameter

is the current usage definition.

Values for the parameter are:
APPLICATION
NUCLEUS

SPECIFIED_AMODE
Optional Parameter

is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

Values for the parameter are:
AMODE_ANY
AMODE_NOT_SPECIFIED
24
31
64

SPECIFIED_RMODE
Optional Parameter

is the residency mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.

Values for the parameter are:
RMODE_ANY
RMODE_NOT_SPECIFIED
24

LDLD gate, GET_NEXT_PROGRAM function

The GET_NEXT_PROGRAM function of the LDLD gate is used to perform an INQUIRE function for the next program in the alphabetic sequence of programs in the current browse session.

Input Parameters

BROWSE_TOKEN
is a valid browse token as returned by the preceding START_BROWSE request.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
END_LIST

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESS
Optional Parameter

is the type of storage that the program resides in.

Values for the parameter are:
CICS
NONE
READ_ONLY
USER

ENTRY_POINT
Optional Parameter

is the address of the entry point of the program instance.

EXECUTION_KEY
Optional Parameter

is the execution key for the program.

Values for the parameter are:
CICS
USER

LIBRARY
Optional parameter

is the name of the LIBRARY concatenation from which the program was loaded.

LIBRARYDSN
Optional parameter

is the name of the data set within the LIBRARY concatenation from which the program was loaded.

LOAD_POINT
Optional Parameter

is the address of the load point of the program instance.

LOCATION
Optional Parameter

determines where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

Values for the parameter are:

CDSA
ECDSA
ELPA
ERDSA
ESDSA
LPA
NONE
RDSA
SDSA

PROGRAM_ATTRIBUTE
Optional Parameter

reflects the program attribute from the program definition, and is used by the program manager domain to recognize RELOAD programs.

Values for the parameter are:

RELOAD
RESIDENT
REUSABLE
TEST
TRANSIENT

PROGRAM_LENGTH
Optional Parameter

is the length of the program instance in bytes.

PROGRAM_NAME
Optional Parameter

is the name of the program whose attributes have been returned.

PROGRAM_TYPE
Optional Parameter

is the current program copy type.

Values for the parameter are:

PRIVATE
SHARED
TYPE_ANY

PROGRAM_USAGE
Optional Parameter

is the current usage definition.

Values for the parameter are:

APPLICATION
NUCLEUS

PROGRAM_USE_COUNT
Optional Parameter

is the cumulative use count of the program.

PROGRAM_USER_COUNT
Optional Parameter

is the current number of users of the program.

SPECIFIED_AMODE
Optional Parameter

is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

Values for the parameter are:
AMODE_ANY
AMODE_NOT_SPECIFIED
24
31
64

SPECIFIED_RMODE
Optional Parameter

is the residency mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.

Values for the parameter are:
RMODE_ANY
RMODE_NOT_SPECIFIED
24

LDLD gate, IDENTIFY_PROGRAM function

The IDENTIFY_PROGRAM function of the LDLD gate is used to locate the program instance which contains the specified address.

Input Parameters

ADDRESS
is a storage address.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
INSTANCE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESS
Optional Parameter

is the type of storage that the program resides in.

Values for the parameter are:
CICS
NONE
READ_ONLY
USER

CSECT_NAME
Optional Parameter

is the name of the CSECT within the module which contains the address. If no CSECT is available, the module name is returned.

ENTRY_POINT
Optional Parameter

is the address of the entry point of the program instance.

EXECUTION_KEY
Optional Parameter

is the execution key for the program.

Values for the parameter are:
CICS
USER

INSTANCE_USE_COUNT
Optional Parameter

is the current number of users of this instance.

LOAD_POINT
Optional Parameter

is the address of the load point of the program instance.

LOCATION
Optional Parameter

determines where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

Values for the parameter are:
CDSA
ECDSA
ELPA
ERDSA
ESDSA
LPA
NONE
RDSA
SDSA

OFFSET_INTO_CSECT
Optional Parameter

is the offset of the address within the CSECT. If no CSECT is available, the module name is returned.

PROGRAM_ATTRIBUTE
Optional Parameter

reflects the program attribute from the program definition, and is used by the program manager domain to recognize RELOAD programs.

Values for the parameter are:
RELOAD
RESIDENT
REUSABLE
TEST
TRANSIENT

PROGRAM_LENGTH
Optional Parameter

is the length of the program instance in bytes.

PROGRAM_NAME
Optional Parameter

is the name of the program whose attributes have been returned.

PROGRAM_TYPE
Optional Parameter

is the current program copy type.

Values for the parameter are:
PRIVATE
SHARED
TYPE_ANY

PROGRAM_USAGE
Optional Parameter

is the current usage definition.

Values for the parameter are:
APPLICATION
NUCLEUS

SPECIFIED_AMODE
Optional Parameter

is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

Values for the parameter are:
AMODE_ANY
AMODE_NOT_SPECIFIED
24
31
64

SPECIFIED_RMODE
Optional Parameter

is the residency mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.

Values for the parameter are:
RMODE_ANY
RMODE_NOT_SPECIFIED
24

LDLD gate, INQUIRE_OPTIONS function

The INQUIRE_OPTIONS function of the LDLD gate is used to return loader global options.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHARED_PROGRAMS
Optional Parameter

indicates whether the loader is utilizing LPA-resident programs to satisfy ACQUIRE requests.

Values for the parameter are:
NO
YES

STORAGE_FACTOR
Optional Parameter

indicates the percentage of system free storage that may be occupied by program instances that have a zero use count.

LDLD gate, INQUIRE_PROGRAM function

The INQUIRE_PROGRAM function of the LDLD gate is used to return the details of a specific program.

Input Parameters

PROGRAM_NAME
specifies the name of the required program.

PROGRAM_TOKEN
is a valid program-identifying token as returned by a previous DEFINE or ACQUIRE request for the same program name.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
PROGRAM_NOT_DEFINED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_PROGRAM_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESS
Optional Parameter

is the type of storage that the program resides in.

Values for the parameter are:

```
CICS
NONE
READ_ONLY
USER
```

ENTRY_POINT
Optional Parameter

is the address of the entry point of the program instance.

EXECUTION_KEY
Optional Parameter

is the execution key for the program.

Values for the parameter are:

```
CICS
USER
```

LIBRARY
Optional parameter

is the name of the LIBRARY concatenation from which the program was loaded.

LIBRARYDSN
Optional parameter

is the name of the data set within the LIBRARY concatenation from which the program was loaded.

LOAD_POINT
Optional Parameter

is the address of the load point of the program instance.

LOCATION
Optional Parameter

determines where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

Values for the parameter are:
- CDSA
- ECDSA
- ELPA
- ERDSA
- ESDSA
- LPA
- NONE
- RDSA
- SDSA

NEW_PROGRAM_TOKEN
Optional Parameter

is the identifying token that may be used on subsequent ACQUIRE or RELEASE calls for this program name.

PROGRAM_ATTRIBUTE
Optional Parameter

reflects the program attribute from the program definition, and is used by the program manager domain to recognize RELOAD programs.

Values for the parameter are:
- RELOAD
- RESIDENT
- REUSABLE
- TEST
- TRANSIENT

PROGRAM_LENGTH
Optional Parameter

is the length of the program instance in bytes.

PROGRAM_TYPE
Optional Parameter

is the current program copy type.

Values for the parameter are:
- PRIVATE
- SHARED
- TYPE_ANY

PROGRAM_USAGE
Optional Parameter

is the current usage definition.

Values for the parameter are:
- APPLICATION
- NUCLEUS

PROGRAM_USE_COUNT
Optional Parameter

is the cumulative use count of the program.

PROGRAM_USER_COUNT
Optional Parameter

is the current number of users of the program.

SPECIFIED_AMODE
Optional Parameter

is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

Values for the parameter are:
```
AMODE_ANY
AMODE_NOT_SPECIFIED
24
31
64
```

SPECIFIED_RMODE
Optional Parameter

is the residency mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.

Values for the parameter are:
```
RMODE_ANY
RMODE_NOT_SPECIFIED
24
```

LDLD gate, REFRESH_PROGRAM function

The REFRESH_PROGRAM function of the LDLD gate is used to inform the loader domain that a new version of the program has been cataloged, and that this version of the named program should be used for all future ACQUIRE requests.

Input Parameters

PROGRAM_NAME
specifies the name of the required program.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LIBRARY_IO_ERROR
LOOP
OS_STORAGE_SHORTAGE
```

The following values are returned when RESPONSE is EXCEPTION:
```
PROGRAM_NOT_DEFINED
PROGRAM_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NEW_VERSION_FOUND
Optional Parameter

indicates whether a new version of the program has been found.

Values for the parameter are:

NO
YES

LDLD gate, RELEASE_PROGRAM function

The RELEASE_PROGRAM function of the LDLD gate is used to inform the loader domain that use of a copy of the named program is no longer required. The use count of the specified program instance is decremented; if the use count reaches zero, and the program is eligible to be removed from memory, it is removed from memory.

Input Parameters

APPL_CONTEXT_TOKEN
Optional parameter

A token representing the application context of **PLATFORM_ID**, **APPLICATION_ID**, **APPL_MAJOR_VER**, **APPL_MINOR_VER**, and **APPL_MICRO_VER**.

APPLICATION_ID
Optional parameter

The 64-character name of the application associated with the task.

APPL_MAJOR_VER
Optional parameter

The fullword binary value that is the major version of the application associated with the task.

APPL_MINOR_VER
Optional parameter

The fullword binary value that is the minor version of the application associated with the task.

APPL_MICRO_VER
Optional parameter

The fullword binary value that is the micro version of the application associated with the task.

ENTRY_POINT
specifies the address of the entry point of the module.

PLATFORM_ID
Optional parameter

The 64-character name of the platform associated with the task.

PROGRAM_NAME
specifies the name of the required program.

PROGRAM_TOKEN
is a valid program-identifying token as returned by a previous DEFINE or ACQUIRE request for the same program name.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

PROGRAM_NOT_DEFINED
PROGRAM_NOT_IN_USE

The following values are returned when RESPONSE is INVALID:

INVALID_ENTRY_POINT
INVALID_PROGRAM_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LOAD_POINT
Optional Parameter

is the address of the load point of the program instance.

LOCATION
Optional Parameter

determines where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

Values for the parameter are:
CDSA
ECDSA
ELPA
ERDSA
ESDSA
LPA
NONE
RDSA
SDSA

NEW_APPL_CONTEXT_TOKEN
Optional parameter

A token to be used on subsequent calls, representing the application context of **PLATFORM_ID**, **APPLICATION_ID**, **APPL_MAJOR_VER**, **APPL_MINOR_VER**, and **APPL_MICRO_VER**.

PROGRAM_LENGTH
Optional Parameter

is the length of the program instance in bytes.

LDLD gate, SET_OPTIONS function

The SET_OPTIONS function of the LDLD gate is used to set loader global options.

Input Parameters

LLACOPY
Optional Parameter

indicates whether the loader is to use the MVS macro LLACOPY or BLDL to locate programs.

Values for the parameter are:
NEWCOPY
NO
YES

PRVMOD
Optional Parameter

is a list of the names of modules that are not to be used from the MVS link pack area (LPA), but instead are to be loaded as private copies from the DFHRPL or dynamic program LIBRARY.

SHARED_PROGRAMS
Optional Parameter

indicates whether the loader is to use LPA-resident programs to satisfy ACQUIRE requests.

Values for the parameter are:
NO
YES

STORAGE_FACTOR
Optional Parameter

indicates the percentage of system free storage that may be occupied by program instances that have a zero use count.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
CATALOG_ERROR
CATALOG_NOT_OPERATIONAL

The following values are returned when RESPONSE is INVALID:
INVALID_STORAGE_FACTOR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LDLD gate, START_BROWSE function

The START_BROWSE function of the LDLD gate is used to start a browse session.

Input Parameters

ENTRY_POINT
Optional Parameter

specifies the address of the entry point of the module.

PROGRAM_NAME
Optional Parameter

specifies the name of the required program.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

BROWSE_TOKEN
is a token used to refer to this browse session on subsequent browse requests.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Loader domain's generic gates

Table 26 on page 644 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 26. Loader domain's generic gates

Gate	Trace	Functions	Format
DMDM	LD 6001 LD 6002	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
SMNT	LD 4001 LD 4002	STORAGE_NOTIFY	SMNT
STST	LD 5001 LD 5002	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
LDRL	LD 8401 LD 8402	CREATE DISCARD INQUIRE_BY_NAME SET	RLCB

Modules

Module	Function
DFHLDDM	Handles the following requests: PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHLDDMI	Reinstates any program resources and dynamic LIBRARY resources defined during previous runs of CICS. It is called by DFHLDDM.
DFHLDDUF	Formats the loader domain control blocks in a CICS system.
DFHLDLD	Directs the following requests to DFHLDLD1, DFHLDLD2, or DFHLDLD3, as appropriate: ACQUIRE_PROGRAM RELEASE_PROGRAM REFRESH_PROGRAM DEFINE_PROGRAM DELETE_PROGRAM INQUIRE_PROGRAM START_BROWSE GET_NEXT_PROGRAM GET_NEXT_INSTANCE END_BROWSE IDENTIFY_PROGRAM SET_OPTIONS INQUIRE_OPTIONS CATALOG_OPTIONS
DFHLDLD1	Handles the following requests: ACQUIRE_PROGRAM RELEASE_PROGRAM REFRESH_PROGRAM
DFHLDLD2	Handles the following requests: DEFINE_PROGRAM DELETE_PROGRAM

Module	Function
DFHLDLD3	Handles the following requests: INQUIRE_PROGRAM START_BROWSE GET_NEXT_PROGRAM GET_NEXT_INSTANCE END_BROWSE IDENTIFY_PROGRAM SET_OPTIONS INQUIRE_OPTIONS CATALOG_OPTIONS
DFHLDLB	Handles the following request: LOG_LIBRARY_ORDER and directs the following requests to DFHLDLB2 or DFHLDLB3 as appropriate: ADD_REPLACE_LIBRARY DISCARD_LIBRARY SET_LIBRARY INQUIRE_LIBRARY START_BROWSE_LIBRARY GET_NEXT_LIBRARY END_BROWSE_LIBRARY
DFHLDLB2	Handles the following requests: ADD_REPLACE_LIBRARY DISCARD_LIBRARY
DFHLDLB3	Handles the following requests: SET_LIBRARY INQUIRE_LIBRARY START_BROWSE_LIBRARY GET_NEXT_LIBRARY END_BROWSE_LIBRARY
DFHLDNT	Handles the following request: STORAGE_NOTIFY
DFHLDST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHLDSVC	Provides authorized services for loader domain functions that involve MVS load facilities.
DFHLDTRI	Provides a loader domain trace interpretation routine for CICS dumps and traces.

Log manager domain (LG)

The log manager domain (also sometimes known as "log manager" or "logger") provides facilities for Recovery Manager to write records to the CICS system log, read records from the CICS system log, and maintain the system log deleting obsolete records and shunting old, but still needed, records to a secondary system log.

The log manager also provides facilities to:

- Write user journal, forward recovery and auto journals records to MVS system logger logstreams or the MVS SMF log

- Install, discard and inquire for Journalmodel resource definitions
- Auto-install, discard, inquire and set for Journal definitions
- Connect, disconnect and define for MVS system logger logstreams
- Collect statistics for Journal and Logstream usage.

Log manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the LG domain.

LGBA gate, BROWSE_ALL_GET_NEXT function

Returns the next record in the browse all object.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USER_DATA

is the address of the record just read from the system log.

USER_DATA_LEN

is the length of the record just read from the system log.

USER_TOKEN

is a user token that was passed in by RESTORE_CHAIN_TOKEN.

LGBA gate, END_BROWSE_ALL function

Destroys the browse all object.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGBA gate, START_BROWSE_ALL function

Creates a browse all object for the CICS system log.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCB gate, CHAIN_BROWSE_GET_NEXT function

Creates a browse object for the chain denoted by CHAIN_TOKEN.

Input Parameters

CHAIN_TOKEN
is a chain token.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USER_DATA
is the address of the record just read from the system log.

USER_DATA_LEN
is the length of the record just read from the system log.

LGCB gate, END_CHAIN_BROWSE function

Destroys the chain browse object denoted by CHAIN_TOKEN.

Input Parameters

CHAIN_TOKEN
is a chain token.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCB gate, START_CHAIN_BROWSE function

Creates a browse object for the chain denoted by CHAIN_TOKEN.

Input Parameters

CHAIN_TOKEN
is a chain token.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, BROWSE_CHAINS_GET_NEXT function

Returns the next chain token and moves the browse cursor position to the next chain.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
AKP_KICK_OFF
```

BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

CHAIN_TOKEN
is a new chain token, which can be used as input to RELEASE_CHAIN_TOKEN, RESTORE_CHAIN_TOKEN, START_CHAIN_BROWSE, CHAIN_BROWSE_GET_NEXT, END_CHAIN_BROWSE, MOVE_CHAIN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USER_TOKEN
is a user token that was passed in by RESTORE_CHAIN_TOKEN.

LGCC gate, CREATE_CHAIN_TOKEN function

Creates a CHAIN TOKEN.

Output Parameters

REASON
The values for the parameter are:

ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

CHAIN_TOKEN
is a new chain token, which can be used as input to RELEASE_CHAIN_TOKEN, RESTORE_CHAIN_TOKEN, START_CHAIN_BROWSE, CHAIN_BROWSE_GET_NEXT, END_CHAIN_BROWSE, MOVE_CHAIN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, DELETE_ALL function

Deletes all of the data on both log streams of the CICS system log.

Output Parameters

REASON
The values for the parameter are:

ABEND
AKP_KICK_OFF

BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, DELETE_HISTORY function

Deletes all blocks of data, for both log streams of the CICS system log, that are older than the corresponding history point saved during a call of SET_HISTORY.

Output Parameters

REASON
The values for the parameter are:
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, END_BROWSE_CHAINS function

Destroys the browse chains object.

Output Parameters

REASON
The values for the parameter are:
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, INQUIRE_DEFER_INTERVAL function

Returns the number of millisecoonds for which a forced log write will be deferred.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE
```

DEFER_INTERVAL
is the number of millisecoonds for which a forced log write will be deferred.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, INQUIRE_KEYPOINT_FREQUENCY function

Returns the activity keypoint frequency value in KEYPOINT_FREQUENCY.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE
```

KEYPOINT_FREQUENCY
is the current keypoint frequency value.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, INQUIRE_KEYPOINT_STATS function

Return the number of keypoints that have occurred since the count was last reset.

Output Parameters

REASON

The values for the parameter are:

ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

KEYPOINT_COUNT

is the number of keypoints that have occurred since the count was last reset.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, RELEASE_CHAIN_TOKEN function

Destroys the chain token in CHAIN_TOKEN

Input Parameters

CHAIN_TOKEN

is a chain token.

Output Parameters

REASON

The values for the parameter are:

ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, RESET_KEYPOINT_STATS function

Reset the count of the number of keypoints.

Output Parameters

REASON

The values for the parameter are:

ABEND
AKP_KICK_OFF
BUFFER_FULL

CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, RESTORE_CHAIN_TOKEN function

Creates a chain token and adds the last record (viewed as a chain element) read from the system log during a BROWSE_ALL_GET_NEXT

Input Parameters

USER_TOKEN
is a user token that is returned by BROWSE_CHAINS_GET_NEXT and BROWSE_ALL_GET_NEXT.

Output Parameters

REASON
The values for the parameter are:
ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

CHAIN_TOKEN
is a new chain token, which can be used as input to RELEASE_CHAIN_TOKEN, RESTORE_CHAIN_TOKEN, START_CHAIN_BROWSE, CHAIN_BROWSE_GET_NEXT, END_CHAIN_BROWSE, MOVE_CHAIN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, SET_DEFER_INTERVAL function

Sets the log defer interval.

Input Parameters

DEFER_INTERVAL
is the number of milliseconds for which a forced log write will be deferred. The maximum value that may be specified is 65535 milliseconds.

Output Parameters

REASON

The values for the parameter are:

ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, SET_HISTORY function

Evaluates and saves the current history point for both log streams of the CICS system log. The history point of a log stream is the oldest block id that CICS knows of on the log stream.

Output Parameters

REASON

The values for the parameter are:

ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, SET_KEYPOINT_FREQUENCY function

Sets the activity frequency to KEYPOINT_FREQUENCY.

Input Parameters

KEYPOINT_FREQUENCY

How often, in terms of physical writes to the system log, activity keypoints should be initiated. A value of zero indicates that activity keypoints should not be initiated.

Non-zero values outside the range from 200 to 65535 inclusive are invalid and cause the OUT_OF_RANGE exception to be returned.

Output Parameters

REASON

The values for the parameter are:

ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, START_BROWSE_CHAINS function

Creates a chains browse object and initializes the browse cursor position.

Output Parameters

REASON

The values for the parameter are:

ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGCC gate, SYSINI function

Creates a primary and secondary log stream objects of type MVS(TM) that comprises the CICS system log.

Output Parameters

REASON

The values for the parameter are:

ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION

LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGGL gate, CLOSE function

Invalidates the LOG_TOKEN, on the last usage of a log stream disconnects from the log stream

Input Parameters

LOG_TOKEN
The token returned by OPEN

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
WRITE_ERROR

The following values are returned when RESPONSE is INVALID:
UNKNOWN_LOG_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGGL gate, FORCE function

Ensures that the previously written records have been flushed from the buffer and hardened on the chosen log stream

Input Parameters

LOG_TOKEN
The token returned by OPEN

FORCE_TOKEN
Optional Parameter

Token returned by WRITE to indicate a specific record to be written. If omitted all records are forced.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
WRITE_ERROR

The following values are returned when RESPONSE is INVALID:
UNKNOWN_LOG_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGGL gate, FORCE_JNL function

Ensures that the previously written records have been hardened on the chosen log.

Input Parameters

JNL_NAME
The 8-byte journal name to be opened

FORCE_TOKEN
Optional Parameter

Token returned by WRITE to indicate a specific record to be written. If omitted all records are forced.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
LOG_HAS_FAILED
LOG_IS_DISABLED
LOG_IS_NOT_ACTIVE
LOG_IS_SYSTEM_LOG
WRITE_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGGL gate, INITIALIZE function

Establish subpools, locks, and anchor control blocks

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGGL gate, OPEN function

Opens a general log and returns a log token. The log token is used by the WRITE, FORCE and CLOSE operations.

Input Parameters

COMPONENT
Identifies the component (e.g. FC) opening this stream

JNL_NAME
The 8-byte journal name to be opened

STREAM_NAME
The 26-byte log stream name to be opened

ERROR_GATE
Optional Parameter

The domain gate number that the logger should call using ERROR if an error occurs accessing the log stream.

USER_TOKEN
Optional Parameter

is a user token that is returned by BROWSE_CHAINS_GET_NEXT and BROWSE_ALL_GET_NEXT.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ERROR_OPENING_LOG
INVALID_JNL_NAME
LOG_HAS_FAILED
LOG_IS_DISABLED
LOG_IS_SYSTEM_LOG
LOG_NOT_DEFINED
```

The following values are returned when RESPONSE is INVALID:
INVALID_PARAMETERS

LOG_TOKEN
The token to be used on subsequent WRITE, FORCE, CLOSE requests.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

JNL_STREAM
Optional Parameter

The MVS logstream name associated with the journal being opened

LOG_TYPE
Optional Parameter

The associated log stream type.

Values for the parameter are:
DUMMY
MVS
SMF

LGGL gate, UOW_TIME function

Returns the oldest active transactions first log write time for use in calculating the recovery time for Backup while open.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

UOW_TIME_STAMP
The 8-byte STCK format time of the oldest active transaction that has written log records with the FORCE_AT_SYNC option, or current time if there are no active transactions.

LGGL gate, WRITE function

Write a record to a general log identified by a token from a previous OPEN.

Input Parameters

DATA
The address of a reusable Iliffe vector describing the items of data to be written to the log stream.

LOG_TOKEN
The token returned by OPEN

FORCE_AT_SYNC
Optional Parameter

Indicates that the caller wants the log stream to be forced when the associated transaction reaches Syncpoint. FORCE_AT_SYNC can be used in conjunction with FORCE_NOW. This is needed by File control for ESDS writes which have to be forced immediately but which also need the UOW structure to allow the calculation of Fuzzy backup recovery times.

Values for the parameter are:
NO
YES

FORCE_NOW
Optional Parameter

Indicates that the caller wants to wait until the data has been successfully written to the log stream.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
BUFFER_LENGTH_ERROR
WRITE_ERROR

The following values are returned when RESPONSE is INVALID:
UNKNOWN_LOG_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FORCE_TOKEN

Optional Parameter

A token to be used on a subsequent FORCE to ensure that a specific records and any prior records have been hardened

LGGL gate, WRITE_JNL function

Write a record to a general log identified by a journal name

Input Parameters

COMPONENT

Identifies the component (e.g. FC) opening this stream

DATA

The address of a reusable Iliffe vector describing the items of data to be written to the log stream.

JNL_NAME

The 8-byte journal name to be opened

FORCE_AT_SYNC

Optional Parameter

Indicates that the caller wants the log stream to be forced when the associated transaction reaches Syncpoint. FORCE_AT_SYNC can be used in conjunction with FORCE_NOW. This is needed by File control for ESDS writes which have to be forced immediately but which also need the UOW structure to allow the calculation of Fuzzy backup recovery times.

Values for the parameter are:
NO
YES

FORCE_NOW

Optional Parameter

Indicates that the caller wants to wait until the data has been successfully written to the log stream.

Values for the parameter are:
NO
YES

SUSPEND

Optional Parameter

Supported for compatibility with old EXEC interface. Causes BUFFER_FULL exception to be raised if there is no space rather than waiting for space. The task may still be suspended for many other reasons.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
BUFFER_FULL
BUFFER_LENGTH_ERROR
ERROR_OPENING_LOG
INVALID_JNL_NAME
LOG_HAS_FAILED
LOG_IS_DISABLED
LOG_IS_SYSTEM_LOG
LOG_NOT_DEFINED
WRITE_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FORCE_TOKEN

Optional Parameter

A token to be used on a subsequent FORCE to ensure that a specific records and any prior records have been hardened

LGJN gate, DISCARD function

Remove a journal from the set of known journals to clean up the catalog or to allow it to be reinstalled with a new set of attributes.

Input Parameters

JNL_NAME

The 8-byte journal name to be opened

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
LOG_IS_SYSTEM_LOG
UNKNOWN_JNL_NAME

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGJN gate, END_BROWSE function

Terminate browse and invalidate browse token

Input Parameters

BROWSE_TOKEN

Token returned by START_BROWSE

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGJN gate, EXPLICIT_OPEN function

Inquire on a journal and if the journal does not already exist in the set of known journals perform the autoinstall process to define it.

Input Parameters

JNL_NAME
The 8-byte journal name to be opened

SYSTEM_LOG
Whether or not this journal is to be used as a system log

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ERROR_OPENING_LOG
INVALID_JNL_NAME
JNL_HAS_FAILED
JNL_IS_DISABLED
SYSTEM_LOG_CONFLICT
UNABLE_TO_CREATE_JNL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STREAM_TOKEN
The log stream token if the journal is currently connected to an MVS log stream or the logbuf token for an SMF journal.

If specified the stream shared lock will be acquired and it its the callers responsibility to free the lock when they have finished with the stream token.

JNL_STATUS
Optional Parameter

The associated log stream status. Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use

Values for the parameter are:
CONNECTED
DISABLED
DISCONNECTED
FAILED
FLUSH

LOG_TOKEN
Optional Parameter

The token to be used on subsequent WRITE, FORCE, CLOSE requests.

LOG_TYPE
Optional Parameter

The associated log stream type.

Values for the parameter are:
DUMMY

MVS
SMF

STREAM_NAME
Optional Parameter

The associated MVS log stream name. Blank for SMF or DUMMY

STRUCTURE_NAME
Optional Parameter

is the 16 byte name of the coupling facility structure of the log stream.

LGJN gate, GET_NEXT function

Return information for next Journal.

Input Parameters

BROWSE_TOKEN
Token returned by START_BROWSE

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NO_MORE_DATA_AVAILABLE

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

JNL_NAME
The next 8-byte Journal name found

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

JNL_STATUS
Optional Parameter

The associated log stream status. Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use.

Values for the parameter are:
CONNECTED
DISABLED
DISCONNECTED
FAILED

LOG_TYPE
Optional Parameter

The associated log stream type.

Values for the parameter are:
DUMMY
MVS
SMF

STREAM_NAME
Optional Parameter

The associated MVS log stream name. Blank for SMF or DUMMY

SYSTEM_LOG
Optional Parameter

Whether or not the journal is a system log.

Values for the parameter are:

NO
YES

LGJN gate, IMPLICIT_OPEN function

Inquire on a journal and if the journal does not already exist in the set of known journals perform the autoinstall process to define it. If the associated log stream has not been opened then it is opened and the stream token returned.

Input Parameters

JNL_NAME
The 8-byte journal name to be opened

SYSTEM_LOG
Whether or not this journal is to be used as a system log.

Values for the parameter are:
NO
YES

FORCE
Optional Parameter

Indicates that a force of the data in the buffer has been requested. This is used to indicate when the stats field in the journal info, which records the number of flushes, needs incrementing.

Values for the parameter are:
NO
YES

WRITE_BYTES
Optional Parameter

The number of bytes of data being written, as a 64 bit value. This field is used to update the bytes counter in the stats information for a journal, and also indicates that the writes counter also needs incrementing.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ERROR_OPENING_LOG
INVALID_JNL_NAME
JNL_HAS_FAILED
JNL_IS_DISABLED
SYSTEM_LOG_CONFLICT
UNABLE_TO_CREATE_JNL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STREAM_TOKEN
The log stream token if the journal is currently connected to an MVS log stream or the logbuf token for an SMF journal. If specified the stream shared lock will be acquired and it its the caller's responsibility to free the lock when it has finished with the stream token.

JNL_STATUS
Optional Parameter

The associated log stream status. Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use.

Values for the parameter are:

CONNECTED
DISABLED
DISCONNECTED
FAILED
FLUSH

LOG_TYPE
Optional Parameter

The associated log stream type.

Values for the parameter are:
DUMMY
MVS
SMF

STREAM_NAME
Optional Parameter

The associated MVS log stream name. Blank for SMF or DUMMY

LGJN gate, INITIALIZE function

Establish subpools, locks, and anchor control blocks

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGJN gate, INQUIRE function

Returns information about the current state of a user journal

Input Parameters

JNL_NAME
The 8-byte journal name to be opened

FORCE
Optional Parameter

Indicates that a force of the data in the buffer has been requested. This is used to indicate when the stats field in the journal info, which records the number of flushes, needs incrementing.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
UNKNOWN_JNL_NAME

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

JNL_STATUS
Optional Parameter

The associated log stream status. Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use.

Values for the parameter are:
CONNECTED

DISABLED
DISCONNECTED
FAILED

LOG_TYPE
Optional Parameter

Values for the parameter are:
DUMMY
MVS
SMF

STREAM_NAME
Optional Parameter

The associated MVS log stream name. Blank for SMF or DUMMY

STREAM_TOKEN
Optional Parameter

The log stream token if the journal is currently connected to an MVS log stream or the logbuf token for an SMF journal. If specified the stream shared lock will be acquired and it its the callers responsibility to free the lock when they have finished with the stream token.

SYSTEM_LOG
Optional Parameter

Whether or not the journal is a system log.

Values for the parameter are:
NO
YES

LGJN gate, PROCESS_STATISTICS function

Deal with the various types of requests for journal statistics using the information in the STST parameter list.

Input Parameters

STATS_PARMS
The address of the STST parameter list.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NO_JOURNALS_DEFINED
UNKNOWN_JNL_NAME

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGJN gate, SET function

Update the status of the Journal.

Input Parameters

JNL_NAME
The 8-byte journal name to be opened

JNL_STATUS
The new status for the journal.

Values for the parameter are:
CONNECTED
DISABLED

DISCONNECTED
FAILED
FLUSH

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ERROR_OPENING_LOG
INVALID_JNL_NAME
JNL_ALREADY_IN_REQ_STATE
JNL_HAS_FAILED
JNL_IS_NOT_ACTIVE
LOG_IS_SYSTEM_LOG
SYSTEM_LOG_CONFLICT
UNABLE_TO_CREATE_JNL
UNKNOWN_JNL_NAME
WRITE_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGJN gate, START_BROWSE function

Initialize browse token for subsequent GET_NEXT requests

Output Parameters

BROWSE_TOKEN

Token for use on subsequent GET_NEXT requests

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGJN gate, STREAM_FAIL function

Marks all journals that have used the failing log stream as failed, issues a message, and closes the stream connection. This ensures that all subsequent activity for the log stream is rejected until either CICS is restarted or the operator explicitly reactivates the journal

Input Parameters

STREAM_NAME

The 26-byte log stream name to be opened

STREAM_TOKEN

The token of the log stream that has failed

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGLB gate, CONNECT function

Creates a log stream object and if of type MVS, a connection is made to the log stream, denoted by its name, through the MVS logger.

Input Parameters

JOURNAL_NAME

is the journal name associated with the log stream on this request.

LOG_TYPE
is the log stream type.

Values for the parameter are:
MVS
SMF

STREAM_NAME
The 26-byte log stream name to be opened

SYSTEM_LOG
Whether or not this journal is to be used as a system log.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
ABEND
AKP_KICK_OFF
BUFFER_FULL
BUFFER_LENGTH_ERROR
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE

LOGBUF_TOKEN
is the token denoting the connected log stream, which can be used as input to GL_WRITE, GL_FORCE and DISCONNECT.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STRUCTURE_NAME
Optional Parameter

is the 16 byte name of the coupling facility structure of the log stream.

LGLB gate, DISCONNECT function

Destroys the log stream object and if it is of type MVS, disconnects from the MVS logger.

Input Parameters

LOGBUF_TOKEN
is the token of the log stream created during a call of CONNECT.

Output Parameters

REASON
The values for the parameter are:
ABEND
AKP_KICK_OFF
BUFFER_FULL
BUFFER_LENGTH_ERROR
CONNECT_FAILURE
END_OF_CHAINS

END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGLB gate, DISCONNECT_ALL function

Ensures that any data in the output buffer has been written to the physical media before the stream connection is destroyed for all connected streams.

Output Parameters

REASON
The values for the parameter are:
ABEND
AKP_KICK_OFF
BUFFER_FULL
BUFFER_LENGTH_ERROR
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGLB gate, GL_FORCE function

Ensures that the output buffer denoted by FORCE_TOKEN for the log stream denoted by LOGBUF_TOKEN has been written to the physical media.

Input Parameters

FORCE_TOKEN
Token returned by WRITE to indicate a specific record to be written. If omitted all records are forced.

LOGBUF_TOKEN
is the token of the log stream created during a call of CONNECT.

Output Parameters

REASON
The values for the parameter are:
ABEND
AKP_KICK_OFF
BUFFER_FULL
BUFFER_LENGTH_ERROR
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION

LOG_NOT_DEFINED
LOOP
WRITE_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGLB gate, GL_WRITE function

Writes a record to a general log denoted by LOGBUF_TOKEN.

Input Parameters

COMPONENT
Identifies the component (e.g. FC) opening this stream

DATA
The address of a reusable Iliffe vector describing the items of data to be written to the log stream.

JOURNAL_NAME
is the journal name associated with the log stream on this request.

LOGBUF_TOKEN
is the token of the log stream created during a call of CONNECT.

SUSPEND
Supported for compatibility with old EXEC interface. Causes BUFFER_FULL exception to be raised if there is no space rather than waiting for space. The task may still be suspended for many other reasons.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
ABEND
AKP_KICK_OFF
BUFFER_FULL
BUFFER_LENGTH_ERROR
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE

FORCE_TOKEN
A token to be used on a subsequent FORCE to ensure that a specific records and any prior records have been hardened

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGLD gate, DISCARD function

Remove a JournalModel from the set of defined JournalModels

Input Parameters

JOURNALMODEL_NAME
The 8-byte JournalModel name to be inquired upon

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

UNKNOWN_JOURNALMODEL_NAME

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGLD gate, END_BROWSE function

Terminate browse and invalidate browse token

Input Parameters

BROWSE_TOKEN

Token returned by START_BROWSE

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

INVALID_BROWSE_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGLD gate, GET_NEXT function

Return information for next JournalModel entry

Input Parameters

BROWSE_TOKEN

Token returned by START_BROWSE

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

NO_MORE_DATA_AVAILABLE

The following values are returned when RESPONSE is INVALID:

INVALID_BROWSE_TOKEN

JOURNALMODEL_NAME

The next 8-byte JournalModel name

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

JNL_TEMPLATE

Optional Parameter

The associated journal name template

LOG_TYPE

Optional Parameter

The associated log stream type.

Values for the parameter are:

DUMMY
MVS
SMF

STREAM_PROTOTYPE

Optional Parameter

The associated MVS log stream name prototype

LGLD gate, INITIALIZE function

Establish subpools, locks, and anchor control blocks

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGLD gate, INQUIRE function

Returns information about the current state of a JournalModel

Input Parameters

JOURNALMODEL_NAME

The 8-byte JournalModel name to be inquired upon

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

UNKNOWN_JOURNALMODEL_NAME

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

JNL_TEMPLATE

Optional Parameter

The associated journal name template

LOG_TYPE

Optional Parameter

The associated log stream type.

Values for the parameter are:

DUMMY
MVS
SMF

STREAM_PROTOTYPE

Optional Parameter

The associated MVS log stream name prototype

LGLD gate, INSTALL function

Create/replace JournalModel entry

Input Parameters

JNL_TEMPLATE

The associated journal name template

JOURNALMODEL_NAME

The 8-byte JournalModel name to be inquired upon

LOG_TYPE

is the log stream type.

Values for the parameter are:

DUMMY
MVS
SMF

STREAM_PROTOTYPE

The associated MVS log stream name prototype

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_JNL_TEMPLATE
INVALID_STREAM_PROTOTYPE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGLD gate, MATCH function

Find JournalModel entry that best matches a journal name. Variables in the stream name prototype are resolved and the resultant stream name is returned.

Input Parameters

JNL_NAME

The 8-byte journal name to be opened

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_JNL_NAME
```

LOG_TYPE

The associated log stream type.

Values for the parameter are:

```
DUMMY
MVS
SMF
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STREAM_NAME

The associated MVS log stream name. Blank for SMF or DUMMY

LGLD gate, START_BROWSE function

Initialize browse token for subsequent GET_NEXT requests

Output Parameters

BROWSE_TOKEN

Token for use on subsequent GET_NEXT requests

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGMV gate, MOVE_CHAIN function

Destroys the chain browse object denoted by CHAIN_TOKEN.

Input Parameters

CHAIN_TOKEN

is a chain token.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
AKP_KICK_OFF
BUFFER_FULL
```

CONNECT_FAILURE
DUMMY_SECONDARY_LOG
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGPA gate, INQUIRE_PARAMETERS function

Inquire logger domain parameters.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
OUT_OF_RANGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DEFER_INTERVAL
Optional Parameter

is the number of millisecoonds for which a forced log write will be deferred.

KEYPOINT_FREQUENCY
Optional Parameter

is the current keypoint frequency value.

LGPA gate, SET_PARAMETERS function

Set logger domain parameters.

Input Parameters

DEFER_INTERVAL
Optional Parameter

is the number of milliseconds for which a forced log write will be deferred. The maximum value that may be specified is 65535 milliseconds.

KEYPOINT_FREQUENCY
Optional Parameter

How often, in terms of physical writes to the system log, activity keypoints should be initiated. A value of zero indicates that activity keypoints should not be initiated.
Non-zero values outside the range from 200 to 65535 inclusive are invalid and cause the OUT_OF_RANGE exception to be returned.

Output Parameters

REASON
The values for the parameter are:
OUT_OF_RANGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGSR gate, LOGSTREAM_STATS function

Collects, and resets if required, the log stream statistics of either the log stream denoted by LOGSTREAM_NAME or of all log streams known to the log manager.

Input Parameters

ALL

if specified then the request is for all log streams of type MVS known to the log manager.

Values for the parameter are:

NO
YES

DATA

The address of a reusable Iliffe vector describing the items of data to be written to the log stream.

Values for the parameter are:

NO
YES

LOGSTREAM_NAME

if specified then this is a log stream name, which must be of type MVS.

RESET

is a request qualifier.

Values for the parameter are:

NO
YES

STATS_BUFFER_ADDR

is the address of a buffer to put the log stream statistics record(s).

STATS_BUFFER_LENGTH

is the length of the buffer.

Output Parameters

REASON

The values for the parameter are:

ABEND
AKP_KICK_OFF
BUFFER_FULL
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
OUT_OF_RANGE
WRITE_FAILURE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGST gate, CONNECT function

Connect to an MVS log stream, or increment use count on subsequent call.

Input Parameters

STREAM_NAME

The 26-byte log stream name to be opened

SYSTEM_LOG
Whether or not this journal is to be used as a system log.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CONNECT_FAILURE
DEFINE_FAILURE
LOG_HAS_FAILED
SYSTEM_LOG_CONFLICT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STREAM_TOKEN
The log stream token if the journal is currently connected to an MVS log stream or the logbuf token for an SMF journal.

If specified the stream shared lock will be acquired and it its the callers responsibility to free the lock when they have finished with the stream token.

STRUCTURE_NAME
Optional Parameter

is the 16 byte name of the coupling facility structure of the log stream.

LGST gate, DISCONNECT function

Decrement the stream use count and disconnect from the MVS logger on last use

Input Parameters

STREAM_TOKEN
The token of the log stream that has failed

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGST gate, END_BROWSE function

Terminate browse and invalidate browse token

Input Parameters

BROWSE_TOKEN
Token returned by START_BROWSE

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGST gate, GET_NEXT function

Return information for next stream entry

Input Parameters

BROWSE_TOKEN
Token returned by START_BROWSE

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NO_MORE_DATA_AVAILABLE

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STREAM_NAME
The associated MVS log stream name. Blank for SMF or DUMMY

FAILED
Optional Parameter

The MVS log stream has failed

Values for the parameter are:
NO
YES

SYSTEM_LOG
Optional Parameter

Whether or not the journal is a system log.

Values for the parameter are:
NO
YES

USE_CT
Optional Parameter

The current number of users of the stream

LGST gate, INITIALIZE function

Establish subpools, locks, and anchor control blocks

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGST gate, INQUIRE function

Returns information about the current state of a stream name

Input Parameters

STREAM_NAME
The 26-byte log stream name to be opened

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
UNKNOWN_STREAM_NAME

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FAILED
Optional Parameter

The MVS log stream has failed

Values for the parameter are:
NO
YES

SYSTEM_LOG
Optional Parameter

Whether or not the journal is a system log.

Values for the parameter are:
NO
YES

USE_CT
Optional Parameter

The current number of users of the stream

LGST gate, START_BROWSE function

Initialize browse token for subsequent GET_NEXT requests

Output Parameters

BROWSE_TOKEN
Token for use on subsequent GET_NEXT requests

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGWF gate, FORCE_DATA function

Ensures that the output buffer denoted by FORCE_TOKEN has been written to the physical media.

Input Parameters

FORCE_TOKEN
Token returned by WRITE to indicate a specific record to be written. If omitted all records are forced.

Output Parameters

REASON
The values for the parameter are:
ABEND
AKP_KICK_OFF
BUFFER_FULL
BUFFER_LENGTH_ERROR
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP
WRITE_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LGWF gate, WRITE function

Writes a record to the CICS system log.

Input Parameters

CHAIN_TOKEN
is a chain token.

DATA
The address of a reusable Iliffe vector describing the items of data to be written to the log stream.

FORCE

Indicates that a force of the data in the buffer has been requested.

This is used to indicate when the statistics field in the journal information, which records the number of flushes, needs incrementing.

Values for the parameter are:
NO
YES

RAISE_LENGERR
is a request qualifier. RAISE_LENGERR(YES) indicates that if the data length is too large to fit into the output buffer then an EXCEPTION condition is returned to the caller.

Values for the parameter are:
NO
YES

SUSPEND
Supported for compatibility with old EXEC interface. Causes BUFFER_FULL exception to be raised if there is no space rather than waiting for space. The task may still be suspended for many other reasons.

Values for the parameter are:
NO
YES

MOVE_NEEDED
Optional Parameter

Binary value indicating whether existence records are to be moved to the shunt log on each activity keypoint.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
ABEND
AKP_KICK_OFF
BUFFER_FULL
BUFFER_LENGTH_ERROR
CONNECT_FAILURE
END_OF_CHAINS
END_OF_DATA
INVALID_FORMAT
INVALID_FUNCTION
LOG_NOT_DEFINED
LOOP

WRITE_FAILURE

FORCE_TOKEN
A token to be used on a subsequent FORCE to ensure that a specific records and any prior records have been hardened

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Logger manager domain's generic gates

Table 27 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 27. Log manager domain's generic gates

Gate	Trace	Function	Format
APUE	LG 0101 LG 0102	SET_EXIT_STATUS	APUE
DMDM	LG 0101 LG 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	LG 0101 LG 0102	COLLECT_STATISTICS COLLECT_RESOURCE_STATISTICS	STST

In Initialization processing, the log manager domain retrieves Journal and Journalmodel information from the catalog and initializes the system log except on a cold start when system log initialization occurs after group list install has completed.

In Quiesce processing, the log manager disconnects from MVS(TM) log streams after all transactions have completed.

Log manager domain's call-back gates

Table 28 summarizes the domain's call-back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 28. Log manager domain's call-back gates

Gate	Trace	Function	Format
RMRO	LG 0201 LG 0202	PERFORM_PREPARE PERFORM_COMMIT START_BACKOUT DELIVER_BACKOUT_DATA END_BACKOUT PERFORM_SHUNT PERFORM_UNSHUNT	RMRO

For PERFORM_PREPARE, PERFORM_COMMIT, END_BACKOUT the log manager forces any log buffers written using the FORCE_AT_SYNCH option of the LGGL WRITE gate to the MVS system logger. For the other RMRO gate functions the log manager does nothing.

Log manager domain's call-back formats

Table 29 describes the call-back formats owned by the domain and shows the functions performed on the calls.

Table 29. Log manager domain's call-back formats

Format	Calling module	Function
LGGL	DFHLGGL	ERROR

Note: In the descriptions of the formats, the input parameters are input not to the log manager domain, but to the domain being called by the log manager domain. Similarly, the output parameters are output by the domain that was called by the log manager domain, in response to the call.

LGGL gate, ERROR function

This is a back-to-front or outbound function. The logger will call the domain that issued OPEN, using the gate number specified in ERROR_GATE, when a long term error condition is detected on the opened log stream.

Input Parameters

COMPONENT
The 2-byte component id supplied on OPEN

ERROR_TYPE
Indicates the severity of the error.

Values for the parameter are:
```
LONG_TERM
RECOVERED
```

LOG_TOKEN
The token returned by OPEN

STREAM_NAME
The 26-byte name of the failing log stream name

USER_TOKEN
The 8-byte token supplied on OPEN, this allows the opening domain to determine what resource (eg DSNB) this open is associated with.

JNL_NAME
Optional Parameter

The 8-byte journal name if the open was by journal name

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHLGDM	Log manager domain initialization and termination. Also handles exit activation for XLGSTRM and XRSINDI. Handles the DMDM and APUE gate functions
DFHLGDUF	A routine to format system dump information
DFHLGGL	Handles the LGGL and RMRO gate functions
DFHLGHB	Assesses the availability of the MVS system logger

Module	Function
DFHLGICV	Log record conversion for SSI exit
DFHLGIGT	Log record get routine for SSI exit
DFHLGILA	Lexical analysis for SSI exit
DFHLGIMS	Message composer for SSI exit
DFHLGIPA	Parser for SSI exit
DFHLGIPI	Parse interface for SSI exit
DFHLGISM	Parse message exit for SSI exit
DFHLGJN	Handles the LGJN gate functions
DFHLGLD	Handles the LGLD gate functions
DFHLGPA	Handles the LGPA gate functions
DFHLGSC	Handles the STST gate functions
DFHLGSSI	Handles the batch QSAM access to CICS(R) logstreams via the DD SUBSYS=(LOGR...) SSI interface
DFHLGST	Handles the LGST gate functions
DFHLGTRI	A routine to format trace points
DFHL2DM	Initializes the 'L2' part of the Log Manager Domain
DFHL2TRI	A routine to format the 'L2' trace points
DFHL2LB	Handles the LGLB gate functions
DFHL2SR	Handles the LGSR gate functions
DFHL2WF	Handles the LGWF gate functions
DFHL2CC	Handles the LGCC gate functions
DFHL2CB	Handles the LGCB gate functions
DFHL2BA	Handles the LGBA gate functions
DFHL2MV	Handles the LGMV gate functions
DFHL2BL1	Initializes the Block class data
DFHL2BL2	Retrieves the current block on the CICS system log
DFHL2BS1	Initializes the BrowseableStream class data
DFHL2BS2	Creates a BrowseableStream class instance
DFHL2BS33	Destroys a BrowseableStream class instance
DFHL2BS4	Destroys all BrowseableStream class instance
DFHL2CH1	Initializes the Chain class data
DFHL2CH2	Creates a Chain class instance
DFHL2CH3	Handles start chain browse
DFHL2CH4	Handles chain browse get next
DFHL2CH5	Handles end chain browse
DFHL2CHA	Handles start browse all
DFHL2CHN	Handles browse all get next
DFHL2CHL	Handles end browse all
DFHL2CHH	Handles start browse chains
DFHL2CHG	Handles browse chains get next
DFHL2CHI	Handles end browse chains

Module	Function
DFHL2CHR	Handles chain restore
DFHL2CHS	handles set history point
DFHL2CHE	Handles delete at history point
DFHL2CHM	Handles move chain
DFHL2HS2	Handles the log stream connect request to the MVS logger
DFHL2HS3	Handles the log stream disconnect request to the MVS logger
DFHL2HS4	Handles the log stream delete all request to the MVS logger
DFHL2HS5	Handles the log stream delete history request to the MVS logger
DFHL2HS6	Handles the log stream start browse block request to the MVS logger
DFHL2HS7	Handles the log stream start browse cursor request to the MVS logger
DFHL2HS8	Handles the log stream read browse cursor request to the MVS logger
DFHL2HS9	Handles the log stream end browse cursor request to the MVS logger
DFHL2HSG	Handles the log stream read browse block request to the MVS logger
DFHL2HSJ	Handles the log stream end browse block request to the MVS logger
DFHL2OFI	Initializes the ObjectFactory instance data
DFHL2SL1	Initializes the SystemLog class data
DFHL2SLN	Handles system log log stream open request
DFHL2SLE	Handles system log log stream failure notification
DFHL2SR1	Initializes the Stream class data
DFHL2SR2	Creates a Stream class instance
DFHL2SR3	Destroys a Stream class instance
DFHL2SR4	Collect and resets Stream statistics
DFHL2SR5	Destroys all Stream class instances
DFHL2VPX	Initializes the VariablePool class data

Exits

Two global user exit points are provided in this domain, XLGSTRM and XRSINDI.

XLGSTRM
This exit is called before defining a new log stream to the MVS system logger

XRSINDI
This exit is called when a Journal or Journalmodel is installed or discarded. It is also called when CICS connects or disconnects an MVS system logger logstream.

Lock manager domain (LM)

The lock manager domain provides locking and associated queuing facilities for CICS resources. Before using these facilities, a resource must add a named lock for itself. This lock can then be requested as either exclusive or shared. If an exclusive lock is obtained, no other task may obtain the lock with that name; if a shared lock is obtained, multiple tasks may obtain that lock, and the exclusive lock with that name cannot be acquired.

Lock manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the LM domain.

LMLM gate, ADD_LOCK function

The ADD_LOCK function of the LMLM gate is used to add a named lock to LM's state.

Input Parameters

LOCK_NAME
is an 8-character name.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INSUFFICIENT_STORAGE
LOOP
```

LOCK_TOKEN
is the 8-character token that uniquely identifies the lock, returned to the caller on the this call.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LMLM gate, DELETE_LOCK function

The DELETE_LOCK function of the LMLM gate is used to delete the named lock from LM's state.

Input Parameters

LOCK_TOKEN
is the token returned to the caller on the ADD_LOCK call.

OWNER_TOKEN
Optional Parameter

defines the owner of the lock.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
LOCK_TOKEN_NOT_FOUND
NOT_LOCK_OWNER
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LMLM gate, LOCK function

The LOCK function of the LMLM gate is used to request the lock.

Input Parameters

LOCK_TOKEN
is the token returned to the caller on the ADD_LOCK call.

MODE
defines the type of lock.

Values for the parameter are:
EXCLUSIVE
SHARED

WAIT
Optional Parameter

indicates whether a task is suspended (CICS) or a LOCK_BUSY is to be returned as a reason output parameter (NO).

Values for the parameter are:
CICS
NO

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
INSUFFICIENT_STORAGE
LOOP

The following values are returned when RESPONSE is EXCEPTION:
DUPLICATE_LOCK_OWNER
LOCK_BUSY
LOCK_TOKEN_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LMLM gate, TEST_LOCK_OWNER function

The TEST_LOCK_OWNER function of the LMLM gate is used to test the owner of a lock for itself.

Input Parameters

LOCK_TOKEN
is the token returned to the caller on the ADD_LOCK call.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
LOCK_TOKEN_NOT_FOUND
NOT_LOCK_OWNER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LMLM gate, UNLOCK function

The UNLOCK function of the LMLM gate is used to release the lock.

Input Parameters

LOCK_TOKEN
is the token returned to the caller on the ADD_LOCK call.

MODE
defines the type of lock.

Values for the parameter are:
EXCLUSIVE
SHARED

OWNER_TOKEN
Optional Parameter

defines the owner of the lock.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
LOCK_TOKEN_NOT_FOUND
NOT_LOCK_OWNER
SHARED_LOCK_FREE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Lock manager domain's generic gates

Table 30 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 30. Lock manager domain's generic gates

Gate	Trace	Functions	Format
DMDM	LM 0001 LM 0002	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
DSNT	LM 0005 LM 0006	DISPATCHER_NOTIFY	DSNT

Modules

Module	Function
DFHLMDM	Handles the following requests: PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHLMDS	Handles transaction manager domain MXT_CHANGE_NOTIFY requests.
DFHLMDUF	Formats the LM domain control blocks
DFHLMLM	Handles the following requests: ADD_LOCK DELETE_LOCK LOCK TEST_LOCK_OWNER UNLOCK

Module	Function
DFHLMTRI	Interprets LM domain trace entries

Message domain (ME)

The message domain acts as a repository for CICS messages, and handles the sending of messages to transient data destinations or to the console. It also provides an interface for returning the text of a message to the caller.

Message domain specific gates

The specific gates provide access for other domains to functions that are provided by the ME domain.

MEBM gate, INQUIRE_MESSAGE_DEFINITION function

The INQUIRE_MESSAGE_DEFINITION function of the MEBM gate is used to return the action and severity codes of a message.

Input Parameters

MESSAGE_NUMBER
is the numeric message identifier.

MESSAGE_TABLE
is a table containing all the message definitions for the message domain.

COMPONENT_ID
Optional Parameter

is the component identifier for the message.

Output Parameters

REASON
The values for the parameter are:
MESSAGE_CANNOT_BE_FOUND

ACTION_CODE
is the action code for the message.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SEVERITY_CODE
is the severity of the message.

MEBM gate, INQUIRE_MESSAGE_LENGTH function

The INQUIRE_MESSAGE_LENGTH function of the MEBM gate is used to find the length of the message in order to obtain the appropriate sized buffer to retrieve the message.

Input Parameters

MESSAGE_NUMBER
is the numeric message identifier.

MESSAGE_TABLE
is a table containing all the message definitions for the message domain.

COMPONENT_ID
Optional Parameter

is the component identifier for the message.

INSERT*n*
Optional Parameter

A user-supplied insert, if required by the message definition.

Output Parameters

REASON
The values for the parameter are:
MESSAGE_CANNOT_BE_FOUND

MESSAGE_LENGTH
is the length of the message being inquired on.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MEBM gate, RETRIEVE_MESSAGE function

The RETRIEVE_MESSAGE function of the MEBM gate is used to retrieve the message text and build the message into a buffer.

Input Parameters

MESSAGE_BUFFER
is the buffer to receive the message text.

MESSAGE_NUMBER
is the numeric message identifier.

MESSAGE_TABLE
is a table containing all the message definitions for the message domain.

COMPONENT_ID
Optional Parameter

is the component identifier for the message.

INSERT*n*
Optional Parameter

A user-supplied insert, if required by the message definition.

MODULE_NAME
Optional Parameter

is the name of the module in error, supplied as data for the symptom string.

MODULE_PTF
Optional Parameter

is the PTF level of the module in error, supplied as data for the symptom string.

SUPPRESS_SRBUILD
Optional Parameter

indicates whether or not a symptom record build is suppressed.

Values for the parameter are:
NO
YES

SYMPTOM_BUFFER
Optional Parameter

is the buffer to receive a symptom string for the message.

UPPERCASE
Optional Parameter

determines whether or not messages should be converted to uppercase.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The values for the parameter are:

```
MESSAGE_CANNOT_BE_PRODUCED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MEME gate, CONVERSE function

The CONVERSE function of the MEME gate is used to send a message and receive a reply for messages that use data in 31-bit or 24-bit storage.

Input Parameters

MESSAGE_NUMBER

The numeric message identifier.

REPLY_FORMAT

The format of the reply.

Values for the parameter are:

```
TEXT
TEXT_OR_VALUE
VALUE
```

COMPONENT_ID

Optional Parameter

The component identifier for the message.

INSERT*n*

Optional Parameter

A user-supplied insert, if required by the message definition.

NETNAME

Optional Parameter

The network name to override the network name that the message domain obtains.

PRODUCT

Optional Parameter

An optional product identifier.

REPLY_BUFFER

Optional Parameter

The buffer into which the text reply is to be returned.

TERMID

Optional Parameter

The terminal identifier to override the terminal identifier that the message domain obtains.

TRANID

Optional Parameter

The transaction identifier to override the transaction identifier that the message domain obtains.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INSUFFICIENT_STORAGE
INVALID_MODULE_PTR
```

INVALID_TEMPLATE
MAX_REPLIES_EXCEEDED

The following values are returned when RESPONSE is EXCEPTION:
REPLY_BUFFER_TOO_SMALL

The following values are returned when RESPONSE is INVALID:
INVALID_COMPONENT_TYPE
INVALID_DESTINATION
INVALID_FUNCTION
INVALID_INSERT
INVALID_REPLY_BUFFER
MESSAGE_NOT_FOUND
MESSAGE_SET_NOT_FOUND
MISSING_INSERT
OPT_INSERT_NOT_FOUND
REPLY_BUFFER_REQUIRED
REPLY_INDEX_REQUIRED
RETRY_MSG_LOCATE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REPLY_INDEX
Optional Parameter

is the number of the template reply option that matches the user's reply text.

MEME gate, INQUIRE_MESSAGE function

The INQUIRE_MESSAGE function of the MEME gate is used to find the system default language as a one-character CICS language suffix and a three-character IBM standard national language code.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

DEFAULT_LANGUAGE_CODE
is the three-character code for the default language.

DEFAULT_LANGUAGE_SUFFIX
is the one-character suffix for the default language.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MEME gate, INQUIRE_MESSAGE_LENGTH function

The INQUIRE_MESSAGE_LENGTH function of the MEME gate is used to find the length of the message in order to obtain the appropriate size buffer to retrieve the message.

Input Parameters

MESSAGE_NUMBER
The numeric message identifier.

COMPONENT_ID
Optional Parameter

The component identifier for the message.

DESTINATIONS
Optional Parameter

One or more transient data (TD) queues that are the destination of the message.

INSERT*n*
Optional Parameter

A user-supplied insert, if required by the message definition.

LANGUAGE
Optional Parameter

An optional language code.

MSGTABLE
Optional Parameter

Indicates that the feature message table is to be used.

NETNAME
Optional Parameter

The network name to override the network name that the message domain obtains.

PRODUCT
Optional Parameter

An optional product identifier.

TERMID
Optional Parameter

The terminal identifier to override the terminal identifier that the message domain obtains.

TRANID
Optional Parameter

The transaction identifier to override the transaction identifier that the message domain obtains.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INSUFFICIENT_STORAGE
INVALID_MODULE_PTR
INVALID_TEMPLATE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_COMPONENT_TYPE
INVALID_FUNCTION
INVALID_INSERT
MESSAGE_NOT_FOUND
MESSAGE_SET_NOT_FOUND
MISSING_INSERT
OPT_INSERT_NOT_FOUND
RETRY_MSG_LOCATE
```

MESSAGE_LENGTH
The length of the message being inquired on.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MEME gate, RETRIEVE_MESSAGE function

The RETRIEVE_MESSAGE function of the MEME gate is used to retrieve the text of a message that uses data in 31-bit or 24-bit storage.

Input Parameters

MESSAGE_BUFFER
The buffer to receive the message text.

MESSAGE_NUMBER
The numeric message identifier.

APPLID
Optional Parameter

The application identifier of the system that originated the message.

COMPONENT_ID
Optional Parameter

The component identifier for the message.

DATE
Optional Parameter

The date of the message.

DESTINATIONS
Optional Parameter

One or more transient data (TD) queues that are the destination of the message.

INSERT*n*
Optional Parameter

A user-supplied insert, if required by the message definition.

LANGUAGE
Optional Parameter

An optional language code.

MSGTABLE
Optional Parameter

Indicates that the feature message table is to be used.

NETNAME
Optional Parameter

The network name to override the network name that the message domain obtains.

PRODUCT
Optional Parameter

An optional product identifier.

SUPPRESS_DUMP
Optional Parameter

Indicates whether dumps have been suppressed.

Values for the parameter are:

```
NO
YES
```

TERMID
Optional Parameter

The terminal identifier to override the terminal identifier that the message domain obtains.

TIME
Optional Parameter

The time of the message.

TRANID

Optional Parameter

The transaction identifier to override the transaction identifier that the message domain obtains.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INSUFFICIENT_STORAGE
INVALID_MODULE_PTR
INVALID_TEMPLATE
```

The following values are returned when RESPONSE is EXCEPTION:

```
MSG_BUFFER_TOO_SMALL
REPLY_BUFFER_TOO_SMALL
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_COMPONENT_TYPE
INVALID_FUNCTION
INVALID_INSERT
INVALID_MESSAGE_BUFFER
MESSAGE_NOT_FOUND
MESSAGE_SET_NOT_FOUND
MISSING_INSERT
OPT_INSERT_NOT_FOUND
RETRY_MSG_LOCATE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MEME gate, SEND_MESSAGE function

The SEND_MESSAGE function of the MEME gate is used to send a message that uses data in 31-bit or 24-bit storage to one or more destinations.

Input Parameters

MESSAGE_NUMBER

The numeric message identifier.

COMPONENT_ID

Optional Parameter

The component identifier for the message.

IGNORE_EXCEPTIONS

Optional Parameter

Specifies whether the caller requests that a failure sending a message to a transient data destination is to be ignored.

INSERT*n*

Optional Parameter

A user-supplied insert, if required by the message definition.

MSGTABLE

Optional Parameter

Indicates that the feature message table is to be used.

NETNAME

Optional Parameter

The network name to override the network name that the message domain obtains.

NOREROUTE

Optional Parameter

Indicates that the message cannot be rerouted.

PRODUCT

Optional Parameter

An optional product identifier.

RESTART_CICS

Optional Parameter

Specifies whether the caller requests CICS to be restarted.

Values for the parameter are:

NO
YES

SYSTEM_DUMPCODE

Optional Parameter

The dump code to be used when the message domain requests a dump on behalf of its caller.

TDQUEUES

Optional Parameter

A block containing the names of the message destinations.

TERMID

Optional Parameter

The terminal identifier to override the terminal identifier that the message domain obtains.

TERMINATE_CICS

Optional Parameter

Specifies whether the caller requests CICS to be terminated.

Values for the parameter are:

NO
YES

TRANID

Optional Parameter

The transaction identifier to override the transaction identifier that the message domain obtains.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
INSUFFICIENT_STORAGE
INVALID_MODULE_PTR
INVALID_TEMPLATE
NO_STORAGE_FOR_WTO

The following values are returned when RESPONSE is INVALID:

INVALID_COMPONENT_TYPE
INVALID_DBCS_FORMAT
INVALID_DESTINATION
INVALID_FUNCTION
INVALID_INSERT

INVALID_MEFO_RESPONSE
MESSAGE_NOT_FOUND
MESSAGE_SET_NOT_FOUND
MISSING_INSERT
OPT_INSERT_NOT_FOUND
RETRY_MSG_LOCATE

The following values are returned when RESPONSE is PURGED:
TDQ_PURGED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RESP2
Optional Parameter

Second response code.

SEVERITY
Optional Parameter

The message severity.

MEME gate, VALIDATE_LANGUAGE_CODE function

The VALIDATE_LANGUAGE_CODE function of the MEME gate is used to determine whether a specific three-letter IBM® standard national language code is valid. If it is valid, this function returns the equivalent one-character CICS language suffix.

Input Parameters

LANGUAGE_CODE
The three-character national language code to be validated. See "Languages and their codes" for the IBM standard three-character codes, and their corresponding one-character CICS language suffixes.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is EXCEPTION:
LANGUAGE_CODE_INVALID
LANGUAGE_NOT_SUPPORTED

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. See RESPONSE parameter on domain interfaces. for more information.

LANGUAGE_SUFFIX
Optional Parameter

The one-character CICS language suffix that corresponds to the input LANGUAGE_CODE.

Languages and their codes

NATLANG code	Globalization code	Language
A	ENG	Alternative English
Q	ARA	Arabic

NATLANG code	Globalization code	Language
1	BEL	Byelorussian
L	BGR	Bulgarian
B	PTB	Brazilian Portuguese
T DBCS	CHT	Traditional Chinese
C DBCS	CHS	Simplified Chinese
2	CSY	Czech
D	DAN	Danish
G	DEU	German
O	ELL	Greek
S	ESP	Spanish
W	FIN	Finnish
F	FRA	French
X	HEB	Hebrew
3	HRV	Croatian
4	HUN	Hungarian
J	ISL	Icelandic
I	ITA	Italian
H DBCS	KOR	Korean
M	MKD	Macedonian
9	NLD	Dutch
N	NOR	Norwegian
5	PLK	Polish
P	PTG	Portuguese
6	ROM	Romanian
R	RUS	Russian
Y	SHC	Serbo-Croatian (Cyrillic)
7	SHL	Serbo-Croatian (Latin)
V	SVE	Swedish
Z	THA	Thai
8	TRK	Turkish
U	UKR	Ukrainian

Notes:

1. **DBCS** denotes Double-Byte Character Set languages.
2. Code letter A means *alternative English* to distinguish edited English message tables from the default US English message tables supplied by CICS. The default US English tables are designated by the language code letter E.
3. The NATLANG code for the selected language is used as the suffix of message data sets.

MEME gate, VALIDATE_LANGUAGE_SUFFIX function

The VALIDATE_LANGUAGE_SUFFIX function of the MEME gate is used to determine whether a specific one-character CICS language suffix is valid. If it is valid, this function returns the equivalent three-character IBM standard national language code.

Input Parameters

LANGUAGE_SUFFIX

is the one-character CICS language code to be validated. The IBM standard three-character codes, and their corresponding one-character CICS language suffices, are listed in "Languages and their codes" on page 694.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
```

The following values are returned when RESPONSE is EXCEPTION:

```
LANGUAGE_NOT_SUPPORTED
LANGUAGE_SUFFIX_INVALID
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LANGUAGE_CODE

Optional Parameter

is the three-character CICS language suffix that corresponds to the input LANGUAGE_SUFFIX.

MEM4 gate, CONVERSE function

The CONVERSE function of the MEM4 gate is used to send a message and receive a reply for messages that use data in 64-bit storage.

Input Parameters

MESSAGE_NUMBER

The numeric message identifier.

REPLY_FORMAT

The format of the reply. The parameter can take the following values:

```
TEXT
TEXT_OR_VALUE
VALUE
```

COMPONENT_ID

Optional Parameter

The component identifier for the message.

INSERT*n*

Optional Parameter

A user-supplied 64-bit insert, if required by the message definition.

NETNAME

Optional Parameter

A new network name that overrides the network name that the message domain obtains.

PRODUCT

Optional Parameter

An optional product identifier.

REPLY_BUFFER

Optional Parameter

The 64-bit buffer into which the text reply is to be returned.

TERMID

Optional Parameter

A new terminal identifier that overrides the terminal identifier that the message domain obtains.

TRANID

Optional Parameter

A new transaction identifier that overrides the transaction identifier that the message domain obtains.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INSUFFICIENT_STORAGE
INVALID_MODULE_PTR
INVALID_TEMPLATE
MAX_REPLIES_EXCEEDED
```

The following values are returned when RESPONSE is EXCEPTION:

```
REPLY_BUFFER_TOO_SMALL
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_COMPONENT_TYPE
INVALID_DESTINATION
INVALID_FUNCTION
INVALID_INSERT
INVALID_REPLY_BUFFER
MESSAGE_NOT_FOUND
MESSAGE_SET_NOT_FOUND
MISSING_INSERT
OPT_INSERT_NOT_FOUND
REPLY_BUFFER_REQUIRED
REPLY_INDEX_REQUIRED
RETRY_MSG_LOCATE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REPLY_INDEX

Optional Parameter

The number of the template reply option that matches the reply text of the user.

MEM4 gate, RETRIEVE_MESSAGE function

The RETRIEVE_MESSAGE function of the MEM4 gate is used to retrieve the text of a message that uses data in 64-bit storage.

Input Parameters

MESSAGE_BUFFER

The buffer to receive the message text.

MESSAGE_NUMBER

The numeric message identifier.

APPLID

Optional Parameter

The application identifier of the system that originated the message.

COMPONENT_ID

Optional Parameter

The component identifier for the message.

DATE

Optional Parameter

The date of the message.

DESTINATIONS
Optional Parameter

One or more transient data (TD) queues that are the destination of the message.

INSERT*n*
Optional Parameter

A user-supplied 64-bit insert, if required by the message definition.

LANGUAGE
Optional Parameter

An optional language code.

MSGTABLE
Optional Parameter

Indicates that the feature message table is to be used.

NETNAME
Optional Parameter

A new network name that overrides the network name that the message domain obtains.

PRODUCT
Optional Parameter

An optional product identifier.

SUPPRESS_DUMP
Optional Parameter

Indicates whether dumps were suppressed. The parameter can take the following values:

```
NO
YES
```

TERMID
Optional Parameter

A new terminal identifier that overrides the terminal identifier that the message domain obtains.

TIME
Optional Parameter

The time of the message.

TRANID
Optional Parameter

A new transaction identifier that overrides the transaction identifier that the message domain obtains.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
EYU_TABLE_MISSING
INSUFFICIENT_STORAGE
INVALID_MODULE_PTR
INVALID_TEMPLATE
```

The following values are returned when RESPONSE is EXCEPTION:

```
MSG_BUFFER_TOO_SMALL
REPLY_BUFFER_TOO_SMALL
```

The following values are returned when RESPONSE is INVALID:

INVALID_COMPONENT_TYPE
INVALID_FUNCTION
INVALID_INSERT
INVALID_MESSAGE_BUFFER
MESSAGE_NOT_FOUND
MESSAGE_SET_NOT_FOUND
MISSING_INSERT
OPT_INSERT_NOT_FOUND
RETRY_MSG_LOCATE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MEM4 gate, SEND_MESSAGE function

The SEND_MESSAGE function of the MEM4 gate is used to send a message that uses data in 64-bit storage to one or more destinations.

Input Parameters

MESSAGE_NUMBER
The numeric message identifier.

COMPONENT_ID
Optional Parameter

The component identifier for the message.

IGNORE_EXCEPTIONS
Optional Parameter

Specifies whether the caller requests that a failure sending a message to a transient data destination is to be ignored.

INSERT*n*
Optional Parameter

A user-supplied 64-bit insert, if required by the message definition.

MSGTABLE
Optional Parameter

Indicates that the feature message table is to be used.

NETNAME
Optional Parameter

A new network name that overrides the network name that the message domain obtains.

NOREROUTE
Optional Parameter

Indicates that the message cannot be rerouted.

PRODUCT
Optional Parameter

An optional product identifier.

RESTART_CICS
Optional Parameter

Specifies whether the caller requests CICS to be restarted. The parameter can take the following values:
NO
YES

SYSTEM_DUMPCODE
Optional Parameter

The dump code to use when the message domain requests a dump on behalf of its caller.

TDQUEUES

Optional Parameter

A 64-bit block that contains the names of the message destinations.

TERMID

Optional Parameter

A new terminal identifier that overrides the terminal identifier that the message domain obtains.

TERMINATE_CICS

Optional Parameter

Specifies whether the caller requests CICS to be terminated. Values for the parameter are:

NO
YES

TRANID

Optional Parameter

A new transaction identifier that overrides the transaction identifier that the message domain obtains.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
EYU_TABLE_MISSING
INSUFFICIENT_STORAGE
INVALID_MODULE_PTR
INVALID_TEMPLATE
NO_STORAGE_FOR_WTO

The following values are returned when RESPONSE is INVALID:

INVALID_COMPONENT_TYPE
INVALID_DBCS_FORMAT
INVALID_DESTINATION
INVALID_FUNCTION
INVALID_INSERT
INVALID_MEFO_RESPONSE
MESSAGE_NOT_FOUND
MESSAGE_SET_NOT_FOUND
MISSING_INSERT
OPT_INSERT_NOT_FOUND
RETRY_MSG_LOCATE

The following values are returned when RESPONSE is PURGED:

TDQ_PURGED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RESP2

Optional Parameter

Second response code.

SEVERITY

Optional Parameter

The message severity.

MESR gate, SET_MESSAGE_OPTIONS function

The SET_MESSAGE_OPTIONS function of the MESR gate is used to set the various message options specified by the system initialization parameters MSGCASE, MSGLVL, and NATLANG.

Input Parameters

LANGUAGES_USED

Optional Parameter

is a list of the languages used in the system.

MESSAGE_CASE

Optional Parameter

is either MIXED for mixed-case messages, or UPPER for messages to be folded to uppercase.

Values for the parameter are:

```
MIXED
UPPER
```

MESSAGE_LEVEL

Optional Parameter

can be 0 or 1. 0 means that information messages do not appear (are suppressed) at the console.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Message domain generic gates

Table 31 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 31. Message domain's generic gates

Gate	Trace	Functions	Format
DMDM	ME 0101 ME 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

In preinitialization processing, the message domain sets the following message options:

- The national languages to be supported during this CICS run
- The message level for initialization messages
- The message case.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

The message domain does no quiesce processing or termination processing.

Modules

Module	Function
DFHCMAC	Displays messages and codes online for the CMAC transaction.
DFHMEBM	Runs in an offline environment, and is provided for batch utility programs to use.
DFHMEBU	Builds a message into a buffer, and also builds a symptom string when required.
DFHMEDM	Runs the necessary domain manager functions; that is, preinitialize, initialize, quiesce, and terminate for the message domain.
DFHMEDUF	An offline routine to format ME domain dump information.
DFHMEFO	Formats a long message into lines of specified length.
DFHMEIN	Provides all the data required to build a message.
DFHMEME	Handles the following functions: **SEND_MESSAGE** Sends a message that uses data in 31-bit or 24-bit storage to any individual or combination of MVS/MCS consoles, or CICS TD queues. **CONVERSE** Sends a message that uses data in 31-bit or 24-bit storage to any individual or combination of MVS/MCS consoles and receives a reply from one of them. **RETRIEVE_MESSAGE** Builds a message that uses data in 31-bit or 24-bit storage and places it in a buffer passed by the caller. **INQUIRE_MESSAGE_LENGTH** Returns the length of a terminal user message. **INQUIRE_MESSAGE** Returns the requested data that the ME domain holds (for example, Default_Language). **VALIDATE_LANGUAGE_CODE** Checks whether a 3-character language code is valid. **VALIDATE_LANGUAGE_SUFFIX** Checks whether a 1-character language suffix is valid.
DFHME64	Handles the following functions for messages that use data in 64-bit storage: **SEND_MESSAGE** Sends a message to any individual or combination of MVS/MCS consoles, or CICS TD queues. **CONVERSE** Sends a message to any individual or combination of MVS/MCS consoles and receives a reply from one of them. **RETRIEVE_MESSAGE** Builds a message and places it in a buffer passed by the caller.
DFHMESR	Collects the system initialization parameter overrides for a particular CICS start.
DFHMETRI	An offline routine to format ME domain trace points.
DFHMEWS	Writes a symptom record that contains a symptom string to SYS1.LOGREC by using the MVS SYMRBLD macro.
DFHMEWT	Provides support to run the MVS Write To Operator With Reply (WTOR) supervisor call (SVC).

Exits

There is one global user exit point in the message domain: XMEOUT.

Markup language domain (ML)

The Markup language domain (ML) processes markup languages.

Markup language domain's specific gates

The specific gates provide access for other domains to functions that are provided by the ML domain.

MLJS gate, DISCARD_JSONTRANSFRM function

Discard a JSONTRANSFRM resource.

Input parameters

JSONTRANSFRM
The 32 character name of the JSONTRANSFRM resource.

Output parameters

RESPONSE
Indicates whether the domain call was successful.

REASON
The following values are returned when RESPONSE is DISASTER:
- INTERNAL_ERROR
- SEVERE_ERROR
- ABEND

The following values are returned when RESPONSE is EXCEPTION:
- JSONTRANSFRM_NOT_FOUND
- INVALID_STATE

MLJS gate, INQUIRE_JSONTRANSFRM function

Query a JSONTRANSFRM resource.

Input parameters

JSONTRANSFRM
The 32 character name of the JSONTRANSFRM resource.

Output parameters

JSONTR
Pointer to the JSONTRANSFRM.

STATUS
The status of the JSONTRANSFRM.

CCSID
The encoding of the JSONTRANSFRM.

RESPONSE
Indicates whether the domain call was successful.

REASON
The following values are returned when RESPONSE is DISASTER:
- INTERNAL_ERROR
- SEVERE_ERROR
- ABEND
- LOOP

The following values are returned when RESPONSE is EXCEPTION:
- JSONTRANSFRM_NOT_FOUND

MLJS gate, INSTALL_JSONTRANSFRM function

Install a JSONTRANSFRM resource.

Input parameters

JSONTRANSFRM
The 32 character name of the JSONTRANSFRM resource.

BUNDLE
The 8 character name of the owning BUNDLE resource.

JSBIND_CONTENT
The contents of the associated JSBind file.

JSBIND_FILENAME
The name of the JSBind file.

RESOURCE_SIGNATURE
The resource signature for the JSONTRANSFRM.

BUNDLE_TOKEN
The bundle token for the owning BUNDLE resource.

RESOURCE_TOKEN
The resource token for the owning BUNDLE resource.

Output parameters

JSONTR
Pointer to the JSONTRANSFRM.

RESPONSE
Indicates whether the domain call was successful.

REASON
The following values are returned when RESPONSE is DISASTER:

- INTERNAL_ERROR
- SEVERE_ERROR
- ABEND

The following values are returned when RESPONSE is EXCEPTION:

- INSTALL_FAILED
- INSTALLED_DISABLED

MLJS gate, SET_JSONTRANSFRM function

Set a JSONTRANSFRM resource.

Input parameters

JSONTRANSFRM
The 32 character name of the JSONTRANSFRM resource.

STATUS
The status of the JSONTRANSFRM.

Output parameters

NEW_STATUS
The new status of the JSONTRANSFRM.

RESPONSE
Indicates whether the domain call was successful.

REASON
The following values are returned when RESPONSE is DISASTER:

- INTERNAL_ERROR
- SEVERE_ERROR
- ABEND
- LOOP

The following values are returned when RESPONSE is EXCEPTION:

- JSONTRANSFRM_NOT_FOUND

- INVALID_STATE

MLJS gate, TRANSFORM_JSON function

Perform a JSON data transformation.

Input parameters

JSONTRANSFRM
The 32 character name of the JSONTRANSFRM resource.

JVMSERVER
The JVMSERVER in which to perform the transformation.

Output parameters

RESPONSE
Indicates whether the domain call was successful.

REASON
The following values are returned when RESPONSE is DISASTER:
- INTERNAL_ERROR
- SEVERE_ERROR
- ABEND
- LOOP

The following values are returned when RESPONSE is EXCEPTION:
- JSONTRANSFRM_NOT_FOUND
- INVALID_STATE
- JVMSERVER_NOT_FOUND
- JVMSERVER_NOT_ENABLED
- JAVA_ERROR

MLPC gate, PARSE_CONTAINER function

Parse the contents of a container.

Input parameters

CCSID
The fullword binary CCSID value. This value is used for header value input and output parameters.

CHANNEL_TOKEN
A token referencing the channel.

CONTAINER_NAME
The 16-character container name.

Output parameters

REASON
The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
LOCK_FAILURE
PARSE_FAILED
```

The following values are returned when RESPONSE is PURGED:
```
TASK_CANCELLED
```

The following values are returned when RESPONSE is EXCEPTION:
```
NOT_WELL_FORMED
RESOURCE_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLTF gate, PARSE_XSDBIND_FILE function

Parse the XML binding file.

Input parameters

XSDBIND_BLOCK
A block that contains the XML binding.

XMLSCHEMA
Optional parameter

A buffer for the XML schema.

XMLTRANSFORM
The 32-byte name of the XMLTRANSFORM resource.

Output parameters

XSDBIND_TOKEN
A token that represents the XML binding, which contains the metadata for transforming the XML to and from application data.

CCSID
Optional parameter

The fullword binary CCSID value.

MAPPINGLEVEL
Optional parameter

The 8-byte character string of the mapping level that was used to generate the XML binding.

MAPPINGVNUM
Optional parameter

The fullword binary value of the version number for the mapping level that was used when generating the XML binding.

MAPPINGRNUM
The fullword binary value of the release number for the mapping level that was used when generating the XML binding.

MINRUNLEVEL
An 8-byte character string of the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

MINRUNVNUM
The fullword binary value of the version number for the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

MINRUNRNUM
The fullword binary value of the release number for the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INTERNAL_ERROR
LOOP
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
XSDBIND_BAD_RUN_LVL
XSDBIND_CONVERSION_ERROR
XSDBIND_INPUT_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLTF gate, QUERY_XML function

Query a fragment of XML.

Input parameters

CHANNEL_NAME
The 16-byte name of the current channel.

CHANNEL_TOKEN
A token that represents the current channel.

ELEMENT_NAME
Optional parameter

A buffer for the XML element name.

ELEMENT_NAMESPACE
Optional parameter

A buffer for the XML element namespace.

NAMESPACE_CONTAINER
Optional parameter

A list of XML namespace prefix declarations that are processed as in scope for the XML.

TYPE_NAME
Optional parameter

A buffer for the name of the XML global data type.

TYPE_NAMESPACE
Optional parameter

A buffer for the namespace of the XML global data type.

XML_CONTAINER
The 16-byte name of the container that has the XML that CICS will transform into application data.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INTERNAL_ERROR
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_NOT_FOUND
CONTAINER_NOT_FOUND_XML
CONTAINER_NOT_FOUND_NS
CONTAINER_NOT_TEXT_MODE
ELEMENT_NAME_BUFF_OVERFLOW
ELEMENT_NMSP_BUFF_OVERFLOW
EMPTY_XML_CONTAINER
EMPTY_XML_DATA
TYPE_NAME_BUFF_OVERVIEW
TYPE_NMSP_BUFF_OVERVIEW
XML_CONVERSION_ERROR
XML_INPUT_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLTF gate, RELEASE_XSDBIND function

Release the XML binding token after the XML transformation or query has completed.

Input parameters

XSDBIND_TOKEN
A token that represents the XML binding, which contains the metadata for transforming the XML to and from application data.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INTERNAL_ERROR
LOOP
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
XSDBIND_TOKEN_INVALID
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLTF gate, TRANSFORM_STRUCTURE_TO_XML function

Transform application data to XML.

Input parameters

CHANNEL_NAME
The 16-byte name of the current channel.

CHANNEL_TOKEN
A token that represents the current channel.

DATA_CONTAINER
The 16-byte name of the container in which CICS puts the application data.

ELEMENT_NAME
Optional parameter

A buffer for the XML element name.

ELEMENT_NAMESPACE
Optional parameter

A buffer for the XML element namespace.

TYPE_NAME
Optional parameter

A buffer for the name of the XML global data type.

TYPE_NAMESPACE
Optional parameter

A buffer for the namespace of the XML global data type.

VALIDATE
Optional parameter

The parameter is set to Yes or No depending on whether validation is required.

XML_CONTAINER
The 16-byte name of the container that has the XML that CICS will transform into application data.

XMLSCHEMA
Optional parameter

A buffer for the XML schema.

XMLTRANSFORM
The 32-byte name of the XMLTRANSFORM resource.

XSDBIND_TOKEN
A token that represents the XML binding, which contains the metadata for transforming the XML to and from application data.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INTERNAL_ERROR
SEVERE_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_NOT_FOUND
CONTAINER_DATATYPE_ERR
CONTAINER_NOT_BIT_MODE
CONTAINER_NOT_FOUND_DATA
CONTAINER_NOT_FOUND_OTHER
DATA_CONVERSION_ERROR
DATA_INTPUT_ERROR
ELEMENT_NOT_SUPPORTED
METADATA_NOT_FOUND
TYPE_NOT_SUPPORTED
VALIDATION_FAILURE
VENDOR_CONVERTER_FAILURE
XSDBIND_TOKEN_INVALID
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLTF gate, TRANSFORM_XML_TO_STRUCTURE function

Transform XML to application data.

Input parameters

CHANNEL_NAME
The 16-byte name of the current channel.

CHANNEL_TOKEN
A token that represents the current channel.

DATA_CONTAINER
The 16-byte name of the container in which CICS puts the application data.

ELEMENT_NAME
Optional parameter

A buffer for the XML element name.

ELEMENT_NAMESPACE
Optional parameter

A buffer for the XML element namespace.

TYPE_NAME
Optional parameter

A buffer for the name of the XML global data type.

TYPE_NAMESPACE
Optional parameter

A buffer for the namespace of the XML global data type.

TYPE_NAME_OVERRIDE
Optional parameter

A block that sets the xsi:type that is assumed when parsing the XML.

TYPE_NAMESPACE_OVERRIDE
Optional parameter

A block that sets the xsi:type that is assumed when parsing the XML.

VALIDATE
Optional parameter

The parameter is set to Yes or No depending on whether validation is required.

XML_CONTAINER
The 16-byte name of the container that has the XML that CICS will transform into application data.

XMLSCHEMA
Optional parameter

A buffer for the XML schema.

XMLTRANSFORM
The 32-byte name of the XMLTRANSFORM resource.

XSDBIND_TOKEN
A token that represents the XML binding, which contains the metadata for transforming the XML to and from application data.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INTERNAL_ERROR
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_NOT_FOUND
CONTAINER_DATATYPE_ERR
CONTAINER_NOT_FOUND_XML
CONTAINER_NOT_FOUND_NS
CONTAINER_NOT_TEXT_MODE
CONTAINER_DATATYPE_ERR
DATA_CONVERSION_ERROR
DATA_INTPUT_ERROR
ELEMENT_NAME_BUFF_OVERFLOW
ELEMENT_NMSP_BUFF_OVERFLOW
ELEMENT_NOT_SUPPORTED
EMPTY_XML_CONTAINER
EMPTY_XML_DATA
METADATA_NOT_FOUND
TYPE_NAME_BUFF_OVERVIEW
TYPE_NMSP_BUFF_OVERVIEW
TYPE_NOT_SUPPORTED
```

VALIDATION_FAILURE
VENDOR_CONVERTER_FAILURE
XML_CONVERSION_ERROR
XML_INPUT_ERROR
XSDBIND_TOKEN_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLXT gate, INSTALL_XMLTRANSFORM function

Install an XMLTRANSFORM resource.

Input parameters

XMLTRANSFORM
The 32-byte name of the XMLTRANSFORM resource.

XSDBIND_CONTENT
A block for the content of the XML binding.

XSDBIND_FILENAME
A buffer for the name of the XML binding file.

RESOURCE_SIGNATURE
The resource signature of the XMLTRANSFORM resource.

BUNDLE
The name of the BUNDLE resource that created the XMLTRANSFORM resource. Either this parameter or the **ATOMSERVICE** parameter is used.

ATOMSERVICE
The name of the ATOMSERVICE resource that created the XMLTRANSFORM resource. Either this parameter or the **BUNDLE** parameter is used.

Output parameters

CCSID
Optional parameter

The fullword binary CCSID value.

MAPPINGLEVEL
Optional parameter

The 8-byte character string of the mapping level that was used to generate the XML binding.

MAPPINGVNUM
Optional parameter

The fullword binary value of the version number for the mapping level that was used when generating the XML binding.

MAPPINGRNUM
The fullword binary value of the release number for the mapping level that was used when generating the XML binding.

MINRUNLEVEL
An 8-byte character string of the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

MINRUNVNUM
The fullword binary value of the version number for the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

MINRUNRNUM
The fullword binary value of the release number for the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

INTERNAL_ERROR
SEVERE_ERROR

The following values are returned when RESPONSE is EXCEPTION:
INSTALL_FAILED
INSTALLED_DISABLED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLXT gate, DISCARD_XMLTRANSFORM function

Discard an XMLTRANSFORM resource.

Input parameters

XMLTRANSFORM
The 32-byte name of the XMLTRANSFORM resource.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
INTERNAL_ERROR
SEVERE_ERROR

The following values are returned when RESPONSE is EXCEPTION:
XMLTRANSFORM_NOT_FOUND
INVALID_STATE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLXT gate, INQUIRE_XMLTRANSFORM function

Inquire about an XMLTRANSFORM resource.

Input parameters

XMLTRANSFORM
The 32-byte name of the XMLTRANSFORM resource.

RESOURCE_SIGNATURE
Optional parameter

The resource signature of the XMLTRANSFORM resource.

XMLSCHEMA_FILENAME
Optional parameter

A buffer for the name of the XML schema file.

XSDBIND_FILENAME
Optional parameter

A buffer for the name of the XML binding file.

Output parameters

ATOMSERVICE
Optional parameter

The name of the ATOMSERVICE resource that is associated with the XMLTRANSFORM resource.

BUNDLE
Optional parameter

The name of the BUNDLE resource that created the XMLTRANSFORM resource.

CCSID
Optional parameter

The fullword binary CCSID value.

MAPPINGLEVEL
Optional parameter

The 8-byte character string of the mapping level that was used to generate the XML binding.

MAPPINGVNUM
Optional parameter

The fullword binary value of the version number for the mapping level that was used when generating the XML binding.

MAPPINGRNUM
The fullword binary value of the release number for the mapping level that was used when generating the XML binding.

MINRUNLEVEL
An 8-byte character string of the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

MINRUNVNUM
The fullword binary value of the version number for the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

MINRUNRNUM
The fullword binary value of the release number for the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

STATUS
Optional parameter

The status of the XMLTRANSFORM resource.

TOTAL_USE_COUNT
The number of times the XMLTRANSFORM resource has been used by CICS.

VALIDATION
The status of validation for the XMLTRANSFORM resource.

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INTERNAL_ERROR
LOOP
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
XMLTRANSFORM_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLXT gate, SET_XMLTRANSFORM function

Set the attributes on the XMLTRANSFORM resource.

Input parameters

XMLTRANSFORM
The 32-byte name of the XMLTRANSFORM resource.

RESOURCE_SIGNATURE
Optional parameter

The resource signature of the XMLTRANSFORM resource.

STATUS
Optional parameter

The status of the XMLTRANSFORM resource, either ENABLED or DISABLED.

VALIDATION
Optional parameter

The status of validation for the XMLTRANSFORM resource.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INTERNAL_ERROR
LOOP
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_STATE
XMLTRANSFORM_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLXT gate, START_BROWSE_XMLTRANSFORM function

Start the browse session for XMLTRANSFORM resources.

Input parameters

There are no input parameters.

Output parameters

BROWSE_TOKEN
A token to browse XMLTRANSFORM resources.

MLXT gate, GET_NEXT_XMLTRANSFORM function

Get the next XMLTRANSFORM resource.

Input parameters

BROWSE_TOKEN
The browse token that was returned by the START_BROWSE_XMLTRANSFORM function.

RESET
Optional parameter

A parameter that indicates whether the statistics for the XMLTRANSFORM are to be reset.

RESOURCE_SIGNATURE
Optional parameter

The resource signature of the XMLTRANSFORM resource.

XMLSCHEMA_FILENAME
Optional parameter

A buffer for the name of the XML schema file.

XSDBIND_FILENAME
Optional parameter

A buffer for the name of the XML binding file.

Output parameters

XMLTRANSFORM
The 32-byte name of the XMLTRANSFORM resource.

ATOMSERVICE
Optional parameter

The name of the ATOMSERVICE resource that is associated with the XMLTRANSFORM resource.

BUNDLE
Optional parameter

The name of the BUNDLE resource that created the XMLTRANSFORM resource.

CCSID
Optional parameter

The fullword binary CCSID value.

MAPPINGLEVEL
Optional parameter

The 8-byte character string of the mapping level that was used to generate the XML binding.

MAPPINGVNUM
Optional parameter

The fullword binary value of the version number for the mapping level that was used when generating the XML binding.

MAPPINGRNUM
The fullword binary value of the release number for the mapping level that was used when generating the XML binding.

MINRUNLEVEL
An 8-byte character string of the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

MINRUNVNUM
The fullword binary value of the version number for the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

MINRUNRNUM
The fullword binary value of the release number for the minimum runtime level that is required to install the XMLTRANSFORM resource in CICS.

STATUS
Optional parameter

The status of the XMLTRANSFORM resource.

TOTAL_USE_COUNT
The number of times the XMLTRANSFORM resource has been used by CICS.

VALIDATION
The status of validation for the XMLTRANSFORM resource.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
INVALID_BROWSE_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MLXT gate, END_BROWSE_XMLTRANSFORM function

End the browse session for XMLTRANSFORM resources.

Input parameters

BROWSE_TOKEN
The browse token that was returned by the START_BROWSE_XMLTRANSFORM function.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`INVALID_BROWSE_TOKEN`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHMLDM	Domain initialization and termination program
DFHMLDUF	ML domain dump formatting program
DFHMLPC	ML domain parse container program
DFHMLTF	Transformation engine for XML
DFHMLTRI	ML domain trace formatting program
DFHMLXT	XMLTRANSFORM resource manager

Monitoring domain (MN)

The monitoring domain is responsible for all monitoring functions within CICS. These functions enable the user to measure the amount of CPU, storage, temporary-storage requests, and so on used per task, and hence charge customers for computing services and help review the performance of a CICS system.

Monitoring domain's specific gates

The specific gates provide access for other domains to functions that are provided by the MN domain.

MNAC gate, GET_ACD function

Get the initial or the current application context data for the task.

Input parameters

INITIAL_ACD_BUFFER
Optional parameter

The buffer for the initial application context data record.

CURRENT_ACD_BUFFER
Optional parameter

The buffer for the current application context data record.

Output parameters

INITIAL_CONTEXT_SET
Optional parameter

Indicator if initial context is set.

CURRENT_CONTEXT_SET
Optional parameter

Indicator if current context is set.

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAC gate, INQUIRE_CURRENT_ACD function

Get the current application context data for the task. Addresses are passed in for platform, application, and operation to minimize the amount of data passed and traced.

Input parameters

APPLICATION_NAME
Optional parameter

The buffer for the application name.

OPERATION_NAME
Optional parameter

The buffer for the application operation name.

PLATFORM_NAME
Optional parameter

The buffer for the platform name.

Output parameters

MAJOR_VERSION
The major version of the operation.

MINOR_VERSION
The minor version of the operation.

MICRO_VERSION
The micro version of the operation.

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAC gate, INQUIRE_CURRENT_ACD_TOKEN function

Get the current application context token for the task.

Input parameters

None

Output parameters

ACD_TOKEN
An etoken that represents the current application context

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAC gate, INQUIRE_INITIAL_ACD function

Get the initial application context data for the task. Addresses are passed in for platform, application, and operation to minimize the amount of data passed and traced.

Input parameters

APPLICATION_NAME
Optional parameter

The buffer for the application name.

OPERATION_NAME
Optional parameter

The buffer for the application operation name.

PLATFORM_NAME
Optional parameter

The buffer for the platform name.

Output parameters

MAJOR_VERSION
The major version of the operation.

MINOR_VERSION
The minor version of the operation.

MICRO_VERSION
The micro version of the operation.

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAC gate, IS_ACD_SET function

Get a YES/NO answer for whether there is an initial application context set on the task, and if there is a current application context that differs from the initial context.

Output parameters

INITIAL_CONTEXT_SET
Indicates if the initial context is set.

CURRENT_CONTEXT_SET
Indicates if the current context is set.

REASON
No exception reasons.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAC gate, POP_ACD function

Returns the application context data to the previous application context data that was used.

Output parameters

REASON

RESPONSE

The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAC gate, PUSH_ACD function

Updates the current application context.

Input parameters

APPLICATION_NAME

The name of the application.

PLATFORM_NAME

The name of the platform.

OPERATION_NAME

The name of the application operation.

MAJOR_VERSION

The application major version.

MINOR_VERSION

The application minor version.

MICRO_VERSION

The application micro version.

REPLACE_INITIAL_CONTEXT

Optional parameter

Indicates whether to replace any existing initial context.

ONLY_SET_INITIAL_CONTEXT

Optional parameter

Indicates whether to fail if there is already an initial context.

Output parameters

INITIAL_CONTEXT

Optional parameter

Indicates whether the initial application context is set.

REASON

RESPONSE

The following values are returned when RESPONSE is EXCEPTION:

```
CONTEXT_ALREADY_SET
```

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAC gate, TRANSACTION_BIND function

Makes any application context passed into DFHXMAT the current application context.

Input parameters

INITIAL_CONTEXT_DATA
The buffer containing initial application context.

CURRENT_CONTEXT_DATA
Optional parameter

The buffer containing the current application context.

Output parameters

MAJOR_VERSION
The major version of the operation.

MINOR_VERSION
The minor version of the operation.

MICRO_VERSION
The micro version of the operation.

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAC gate, TRANSACTION_TERMINATION function

Cleans up any remaining application context elements stored, and this can include the initial application context.

Output parameters

REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAD gate, GET_ASSOCIATED_DATA_LIST function

Retrieve the list of user tasks that are in the local region.

Input parameters

TASK_LIST
The list of tasks in the local region.

UC_DATA_BUFFER
Optional parameter

Buffer for the user correlation data.

Output parameters

LIST_SIZE
The fullword binary number of items in the task list.

REASON
The following values are returned when RESPONSE is EXCEPTION:

BUFFER_TOO_SMALL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAD gate, INQUIRE_ADCB function

Retrieve the association data control block (ADCB) for a task.

Input parameters

ADCB_BUFFER
Buffer for the association data control block (ADCB).

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

BUFFER_TOO_SMALL

NOT_FOUND

UCD_CLOSED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAD gate, INQUIRE_ASSOCIATED_DATA function

Inquire on association data for a task.

Input parameters

TRANNUM
The 4-byte number of the task.

Output parameters

All output parameters are optional.

APPLICATION_NAME
The 64-character name of the application.

C_APPLID
An 8-character APPLID of the CICS region in which this task is running.

C_CLIENT_IPADDR
The 39-character IP address of the TCP/IP client that requested this task to start.

C_CLIENT_PORT
A 31-bit binary value that represents the number of the port that the TCP/IP stack used to send the request that resulted in this task being attached.

C_CLNTIPADDR_FAMILY
The client IP address family can have the following values:
- IPV4
- IPV6
- UNKNOWN

C_FACILITY_NAME
The 8-character name of the facility associated with the initiation of this task.

C_FACILITY_TYPE
The type of facility that initiated this task. This parameter can have one of the following values:
- APPC_SESSION

- BRIDGE
- EVENT
- IIRR
- IP_ECI
- IS_SESSION
- LU61_SESSION
- MRO_SESSION
- NONE
- RRS_UR
- RZ_INSTORE_TRPORT
- SCHEDULER
- SOCKET
- START
- START_TERMINAL
- TERMINAL
- TRANDATA
- WEB
- XM_RUN_TRANSACTION

C_IPADDR_FAMILY
The IP address family can have the following values:
- IPV4
- IPV6
- UNKNOWN

C_IPCONN
An 8-character name of any IPIC connection that was used to receive a request that resulted in this task starting.

C_MVSIMAGE
The 8-character name of the MVS image associated with the TCPIPSERVICE resource used to receive a request that resulted in this task starting.

C_NETID
The 8-character network ID of the terminal from which this task was started.

C_NETNAME
The 8-character name of the terminal from which this task was started.

C_PROGRAM_NAME
The 8-character name of the first program called by a task running this transaction.

C_SERVER_IPADDR
The 39-character area that represents the IP address of the TCP/IP service that scheduled this task.

C_SERVER_PORT
A 31-bit binary value that represents the port number on which the TCPIPSERVICE resource that received the request that resulted in this task being attached is listening.

C_START_TIME
A 21-character representation of the time when this task was started.

C_TCPIP_APPLDATA
The 40-character value of the application data associated by CICS with the socket that received the request that started this task.

C_TCPIP_ZONENAME
The 8-character name of the TCP/IP network security zone associated with the IPIC connection that received the request that resulted in this task starting.

C_TCPIPJOB
The 8-character name of the TCP/IP job associated with the IPIC connection that received the request that resulted in this task starting.

C_TCPIPSERVICE
The 8-character name of the TCPIPSERVICE resource associated with the IPIC connection that received the request that resulted in this task starting.

C_TRANSACTION_GROUP_ID
The 28-character unique identifier that represents the transaction group ID of the originating transaction.

C_USERID1
The 8-character user ID associated with this task.

C_USERID2
The 8-character user ID of the initiating task.

C_SRVRIPADDR_FAMILY
The server IP address family can have the following values:
- IPV4
- IPV6
- UNKNOWN

CLIENT_LOCATION
The 32-character area that represents the SO_CLUSTERCONNTYPE socket option returned by z/OS Communications Server for the facility.

MAJOR_VERSION
The fullword binary value that is the major version of the application.

MICRO_VERSION
The fullword binary value that is the micro version of the application.

MINOR_VERSION
The fullword binary value that is the minor version of the application.

O_APPLID
The 8-character APPLID for the origin descriptor associated with the task.

O_NETWORKID
The 8-character network qualifier for the origin region APPLID on which the task ran.

O_START_TIME
The 21-character representation of the time when the originating task started.

O_TASK_NUMBER
The 4-byte packed decimal identifier of the originating task that is associated with this task.

O_1ST_TRANSID
The 4-character name of the transaction under which the originating task ran.

O_USERID2
The 8-character user ID under which the originating task ran.

O_FACILITY_TYPE
The type of facility that initiated the originating task that is associated with this task. This parameter can have one of the following values:
- BRIDGE
- NONE
- SCHEDULER
- SOCKET
- START
- START_TERMINAL
- TERMINAL
- TRANDATA
- WEB
- XM_RUN_TRANSACTION

O_FACILITY_NAME
An 8-character name of the facility. This parameter is populated when the facility associated with the initiation of the originating task is a transient data queue, a terminal, or a system.

O_NETID
The 8-character network ID of the terminal from which the originating task was started.

O_NETNAME
The 8-character network qualifier for the origin region APPLID on which the task ran.

O_IPADDR_FAMILY
The form of TCP/IP addressing used by the originating task. This parameter can have the following values:
- IPV4
- IPV6
- UNKNOWN

O_CLIENT_IPADDR
The 39-character IP address of the TCP/IP client that requested the originating task to start.

O_CLIENT_PORT
A 31-bit binary value that represents the port number that the TCP/IP stack used to send the request that resulted in attaching the originating task.

O_USER_CORRELATOR
A 64-character area that contains the user correlator data that was added to the associated data origin descriptor by an XAPADMGR global user exit program.

O_ADAPTER_ID
A 64-character area that contains the data that was added to the origin data by the adapter.

O_ADAPTER_DATA_1
A 64-character area that contains the data that was added to the origin data by the adapter.

O_ADAPTER_DATA_2
A 64-character area that contains the data that was added to the origin data by the adapter.

O_ADAPTER_DATA_3
A 64-character area that contains the data that was added to the origin data by the adapter.

OPERATION_NAME
The 64-character name of the application operation.

PLATFORM_NAME
The 64-character name of the platform.

REASON
The following values are returned when RESPONSE is EXCEPTION:
`NOT_FOUND`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAD gate, INQUIRE_PREVIOUS_HOP_DATA function

Retrieve the previous hop data for a task.

Input parameters

TRANNUM
The 4-byte number of the task.

Output parameters

All output parameters are optional.

H_APPLID
The 8-character APPLID from previous hop data.

H_HOP_COUNT
The fullword binary number of times there has been a request from one CICS region to another to initiate a task with which this task is associated.

H_NETWORK_ID
The 8-character network qualifier from previous hop data.

H_START_TIME
The 21-character representation of the task start time from previous hop data.

H_TASK_NUMBER
The 4-byte packed decimal identifier from previous hop data.

H_TRANSID
The 4-character name of the transaction from previous hop data.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAD gate, INQUIRE_PREVIOUS_TRAN_DATA function

Retrieve the previous tran data for a task.

Input parameters

TRANNUM
The 4-byte number of the task.

Output parameters

All output parameters are optional.

H_HOP_COUNT
The fullword binary number of times there has been a request from one CICS region to another to initiate a task with which this task is associated.

H_START_TIME
The 21-character representation of the task start time from previous hop data.

H_TASK_NUMBER
The 4-byte packed decimal identifier from previous hop data.

H_TRANSID
The 4-character name of the transaction from previous hop data.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNAD gate, SET_APP_CONTEXT function

Sets application context into the task's association data control block. Also sets the transaction monitoring area fields for monitoring if an optional property is specified to indicate that the application context being passed is the initial application context for the task.

Input parameters

APPLICATION_NAME
The name of the application.

PLATFORM_NAME
The name of the platform.

OPERATION_NAME
The name of the application operation.

MAJOR_VERSION
The major version of the application.

MINOR_VERSION
The minor version of the application.

MICRO_VERSION
The micro version of the application.

INITIAL_CONTEXT
Optional parameter

Indicates whether this is the initial application context for the task.

Output parameters

REASON

RESPONSE
The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNMN gate, ACCUMULATE_RMI_TIME function

The ACCUMULATE_RMI_TIME function of the MNMN gate is used to accumulate all of the appropriate performance class DFHRMI timing fields.

Input Parameters

TRUE_NAME
is the name of the CICS resource manager being used by your transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INVALID_MONITORING_TOKEN
LOOP
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNMN gate, EXCEPTION_DATA_PUT function

The EXCEPTION_DATA_PUT function of the MNMN gate is used to produce an exception record at the completion of an EXCEPTION condition.

Input Parameters

EXCEPTION_START
is the start time of the exception in stored clock (STCK) format.

EXCEPTION_STOP
is the stop time of the exception in STCK format.

EXCEPTION_TYPE
is the type of exception to be recorded.

Values for the parameter are:

```
BUFFER_WAIT
STRING_WAIT
WAIT
```

RESOURCE_ID
is the identifier of the resource for which the exception data is to be recorded.

RESOURCE_TYPE
is the type of resource for which the exception data is to be recorded.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INVALID_MONITORING_TOKEN
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_RESOURCE_ID_LENGTH
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNMN gate, INQUIRE_MONITORING_DATA function

The INQUIRE_MONITORING_DATA function of the MNMN gate is used to access a transaction's monitoring information.

Input Parameters

DATA_BUFFER
specifies the address and length of a buffer for the monitoring data.

CURRENT_DATA_BUFFER
Optional Parameter

specifies the address and length of a buffer for the current monitoring data.

TRANSACTION_NUMBER
Optional Parameter

is the transaction number for which monitoring data is required.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
LENGTH_ERROR
```

MONITOR_DATA_UNAVAILABLE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNMN gate, INQUIRE_RESOURCE_DATA function

The INQUIRE_RESOURCE_DATA function of the MNMN gate is used to access a transaction's resource data when transaction resource monitoring is active.

Input Parameters

RESOURCE_DATA_BUFFER
specifies the address and length of a buffer for the transaction resource data.

TRANSACTION_NUMBER
Optional Parameter

is the transaction number for which monitoring data is required.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

LENGTH_ERROR
MONITOR_DATA_UNAVAILABLE
RESOURCE_DATA_UNAVAILABLE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNMN gate, MONITOR function

The MONITOR function of the MNMN gate is called to process a user event-monitoring point (EMP).

Input Parameters

POINT
is a value in the range 0 through 255 corresponding to a monitoring point identifier defined in the monitoring control table (MCT).

DATA1
Optional Parameter

supplies 4 bytes of data to be used in the operations performed by this user's EMP.

DATA2
Optional Parameter

supplies 4 bytes of data to be used in the operations performed by this user's EMP.

ENTRYNAME
Optional Parameter

is an ID qualifier, 1 through 8 bytes, corresponding to an entry name specified in the MCT.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND

INVALID_MONITORING_TOKEN
LOOP

The following values are returned when RESPONSE is EXCEPTION:

DATA1_NOT_SPECIFIED
DATA2_NOT_SPECIFIED
INVALID_DATA1_VALUE
INVALID_DATA2_VALUE
POINT_NOT_DEFINED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNMN gate, PERFORMANCE_DATA_PUT function

The PERFORMANCE_DATA_PUT function of the MNMN gate is used to produce a performance record and reset task monitoring information for a conversational task or a syncpoint.

Input Parameters

RECORD_TYPE

is the reason for the record to be output.

Values for the parameter are:

CONVERSE
DELIVER
SYNCPOINT

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
INVALID_MONITORING_TOKEN
LOOP

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNOD gate, DECODE_ODR function

Decode the origin data record.

Input parameters

ODR_ENCODED_BUFFER

The encoded buffer for the origin data record.

ODR_BUFFER

Buffer for the origin data record.

Output parameters

C_TRANSACTION_GROUP_ID

The 28-character unique identifier that represents the transaction group ID of the originating transaction.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNOD gate, ENCODE_ODR function

Encode the origin data record.

Input parameters

C_TRANSACTION_GROUP_ID
The 28-character unique identifier that represents the transaction group ID of the originating transaction.

ODR_BUFFER
Buffer for the origin data record.

ODR_ENCODED_BUFFER
The encoded buffer for the origin data record.

VERSION_SUPPORTED
The release of CICS. This parameter can have the following values:

- PRE_V420
- V420

Output parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNOD gate, GET_ODR function

Retrieve the origin data for a task.

Input parameters

ODR_BUFFER
Buffer for the origin data record.

UPDATE_PHD
Optional parameter

Flag to indicate if the origin data is updated. This parameter can have the following values:

- NO
- YES

Output parameters

C_TRANSACTION GROUP_ID
Optional parameter

The 28-character unique identifier that represents the transaction group ID of the originating transaction.

H_HOP_COUNT
Optional parameter

The fullword binary value for the number of times there has been a request from one CICS region to another to initiate a task with which this task is associated.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNOD gate, INQUIRE_ODR function

Retrieve information about the origin data.

Input parameters

None.

Output parameters

ADAPTER_DATA_SET
Optional parameter

Returns if adapter data is set. This parameter has the following values:

- NO
- YES

REASON
The following values are returned when RESPONSE is EXCEPTION:

NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNOD gate, SET_UCD function

Set the user correlation data.

Input parameters

UC_DATA_BUFFER
Buffer for the user correlation data.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

NOT_FOUND

UCD_CLOSED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNSR gate, INQ_MONITORING function

The INQ_MONITORING function of the MNSR gate is used to enquire on the monitoring classes and the monitoring options.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND

LOOP

APPLICATION_NAMING
Indicates whether application naming support is enabled in the CICS region.

Values for the parameter are:
NO
YES

COMPRESSION
Iindicates whether monitoring record compression is active.

Values for the parameter are:
NO
YES

CONVERSE
Indicates if a transaction performance class record is to be produced for conversational tasks for each pair of terminal control I/O requests.

Values for the parameter are:
NO
YES

DPL_LIMIT
Specifies the maximum number of distributed program links for which you want CICS to perform transaction resource monitoring. It can have a value in the range 0 - 64.

EXCEPTION_STATUS
Indicates whether exception class monitoring is active.

Values for the parameter are:
OFF
ON

FILE_LIMIT
Specifies the maximum number of files for which you want CICS to perform transaction resource monitoring. It can have a value in the range 0 - 64.

FREQUENCY
Is the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction. Frequency times are 0, or in the range 000100 - 240000. The default frequency value is 0, which means that frequency monitoring is inactive.

MONITORING_STATUS
Indicates whether monitoring is active.

Values for the parameter are:
OFF
ON

PERFORMANCE_STATUS
Indicates whether performance class monitoring is active.

Values for the parameter are:
OFF
ON

RESOURCE_STATUS
Indicates whether transaction resource class monitoring is active.

Values for the parameter are:
OFF
ON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMI_STATUS
Indicates whether additional monitoring performance class data is required for the resource managers used by your transaction.

Values for the parameter are:

NO

YES

SYNCPOINT

Indicates if a transaction performance class record is to be produced when a transaction takes an explicit or implicit sync point (unit-of-work).

Values for the parameter are:

NO

YES

TIME

Indicates whether the monitoring time-stamp fields returned on the INQUIRE_MONITORING_DATA function are to be in GMT or local time.

Values for the parameter are:

GMT

LOCAL

TSQUEUE_LIMIT

Specifies the maximum number of temporary storage queues for which you want CICS to perform transaction resource monitoring. It can have a value in the range 0 - 64.

MNSR gate, SET_MCT_SUFFIX function

The SET_MCT_SUFFIX function of the MNSR gate is used to identify to the monitoring domain the suffix of the monitoring control table (MCT).

Input Parameters

SUFFIX

is the 2-character MCT suffix.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND

LOOP

The following values are returned when RESPONSE is EXCEPTION:

MCT_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNSR gate, SET_MONITORING function

The SET_MONITORING function of the MNSR gate is used to set the monitoring classes on or off and to change the monitoring options.

Input Parameters

COMPRESSION

Optional Parameter

Alters the monitoring record compression setting.

Values for the parameter are:

NO

YES

CONVERSE

Optional Parameter

Indicates if a transaction performance class record is to be produced for conversational tasks for each pair of terminal control I/O requests.

Values for the parameter are:

NO

YES

DPL_LIMIT

Optional Parameter

Indicates the number of distributed program links for which you want CICS to perform transaction resource monitoring. The value must be in the range 0 - 64.

EXCEPTION_STATUS

Optional Parameter

Indicates the exception class monitoring setting.

Values for the parameter are:

OFF

ON

FILE_LIMIT

Optional Parameter

Indicates the number of files for which you want CICS to perform transaction resource monitoring. The value must be in the range 0 - 64.

FREQUENCY

Optional Parameter

Is the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction. Frequency times are 0, or in the range 000100 - 240000. The default frequency value is 0, which means that frequency monitoring is inactive.

MONITORING_STATUS

Optional Parameter

Indicates the monitoring status setting.

Values for the parameter are:

OFF

ON

PERFORMANCE_STATUS

Optional Parameter

Indicates the performance class monitoring setting.

Values for the parameter are:

OFF

ON

RESOURCE_STATUS

Optional Parameter

Indicates the transaction resource class monitoring setting.

Values for the parameter are:

OFF

ON

SYNCPOINT

Optional Parameter

Indicates if a transaction performance class record is to be produced when a transaction takes an explicit or implicit sync point (unit-of-work).

Values for the parameter are:

NO
YES

TIME

Optional Parameter

Indicates whether the monitoring time-stamp fields returned on the INQUIRE_MONITORING_DATA function are to be in GMT or local time.

Values for the parameter are:

GMT
LOCAL

TSQUEUE_LIMIT

Optional Parameter

Indicates the maximum number of temporary storage queues for which you want CICS to perform transaction resource monitoring. The value must be in the range 0 - 64.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following value is returned when RESPONSE is EXCEPTION:

INVALID_FREQUENCY
FILE_LIMIT_OUT_OF_RANGE
TSQUEUE_LIMIT_OUT_OF_RANGE
DPL_LIMIT_OUT_OF_RANGE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNTD gate, SET_TRACKING_DATA function

The SET_TRACKING_DATA function of the MNTD gate updates the transaction tracking origin data tag in the transaction tracking origin data.

Input Parameters

TRACKING_TAG

Transaction tracking origin data tag.

Note: TRACKING_TAG is defined as an enumerated type and can have the values from 1 through to 128.

TRACKING_TAG_VALUE

Transaction tracking origin data tag value in the range 129 through to 255.

Note: TRACKING_TAG and TRACKING_TAG_VALUE are mutually exclusive. TRACKING_TAG_VALUE is designed primarily for use by customers using the XPI and can have the values 129 through to 255.

TRACKING_OVERRIDE

Optional Parameter.

Indicates whether the current transaction tracking origin data tag must be overridden.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

NO_ASSOCIATION_DATA

TRACKING_TAG_ALREADY_SET
INVALID_TRACKING_TAG
INVALID_TRACKING_TAG_VALUE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNTD gate, INQUIRE_TRACKING_DATA function

The INQUIRE_TRACKING_DATA function of the MNTD gate inquires on the transaction tracking origin data tag in the transaction tracking origin data for the current transaction.

Output Parameters

TRACKING_TAG
Transaction tracking origin data tag.

TRACKING_TAG_VALUE
Transaction tracking origin data tag value.

REASON
The following values are returned when RESPONSE is EXCEPTION:
NO_ASSOCIATION_DATA

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNXM gate, TRANSACTION_BIND function

The TRANSACTION_BIND function of the MNXM gate is used to inform the monitoring domain of a transaction attach request so that the monitoring domain can complete its control block setup and initialization for the transaction being attached.

Input parameters

ADAPTER_FIELDS
Optional parameter

is the origin data record adapter data for this transaction.

EWLM_CORRELATOR
Optional parameter

is the EWLM correlator for this transaction.

INITIAL_DISPATCH_TIME
is the time when this task was first dispatched after attach.

ODR_ACTION
Optional parameter

indicates the origin data override action for this transaction.

ORIGIN_DATA
Optional parameter

is the transaction origin data record for this transaction.

TASK_ATTACH_TIME
is the time when this task was attached.

WLM_SRC_TOKEN
Optional parameter

is the z/OS Workload Manager (WLM) service report class token.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNXM gate, TRANSACTION_INITIALISATION function

The TRANSACTION_INITIALISATION function of the MNXM gate is used to inform the monitoring domain of a transaction attach request so that the monitoring domain can allocate task monitoring storage.

Input Parameters

INITIAL_DISPATCH_TIME
is the time when this task was first dispatched after attach.

MXT_DELAY_TIME
is the time this task was delayed due to the maximum user task limit (MXT) being reached.

TASK_ATTACH_TIME
is the time when this task was attached.

TCLASS_DELAY_TIME
is the time this task was delayed due to the transaction class (if any) limit for this transaction being reached.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MNXM gate, TRANSACTION_TERMINATION function

The TRANSACTION_TERMINATION function of the MNXM gate is used to inform the monitoring domain of a transaction detach request, so that the monitoring domain can report on task monitoring information and then release the storage.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
INVALID_MONITORING_TOKEN
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Monitoring domain's generic gates

Table 32 on page 738 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 32. Monitoring domain's generic gates

Gate	Trace	Functions	Format
APUE	MN 0601 MN 0602	SET_EXIT_STATUS	APUE
DMDM	MN 0101 MN 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	MN 0401 MN 0402	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
TISR	MN 0801 MN 0802	NOTIFY	TISR
XMNT	MN 0901 MN 0902	MXT_CHANGE_NOTIFY	XMNT

In initialization processing, the monitoring domain sets the initial monitoring options:

- Monitoring control table suffix
- Initial monitoring status
- Initial exception class monitoring status
- Initial performance class monitoring status
- Initial transaction resource class monitoring status
- Initial converse option
- Initial syncpoint option
- Initial time option
- Initial frequency option
- Initial subsystem id.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the global catalog, but is then modified by any relevant system initialization parameters.

In addition:

- If necessary, the monitoring control table (MCT) is loaded and initialized.
- If performance class monitoring is active, CPU timing is started.
- The monitoring domain user exit gate is enabled.
- Messages are sent to the console to indicate whether monitoring is active, and what MCT suffix (if any) is being used.

In quiesce processing, the monitoring domain waits for all transactions that it is monitoring to terminate. Then the final data in the performance class buffer and the transaction resource class buffer, if any, is written to SMF.

The monitoring domain does no termination processing.

Modules

Module	Function
DFHMNAC	Handles the following requests: find the current task's initial application context find the current task's current application context sets the current application context returns to the previous application context cleans up stored application contexts when terminating the task
DFHMNAD	Handles the following requests: find the current task's associated data control block return the data associated with the current task return a list of tasks that have associated data control blocks return data associated with the previous hop
DFHMNDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHMNDUF	Formats the MN domain control blocks in a CICS system dump
DFHMNMN	Handles the following requests: EXCEPTION_DATA_PUT PERFORMANCE_DATA_PUT INQUIRE_MONITORING_DATA MONITOR INQUIRE_RESOURCE_DATA ACCUMULATE_RMI_TIME
DFHMNNT	Handles the following request: MXT_CHANGE_NOTIFY
DFHMNOD	Handles the following requests: set the user correlation data returned from user exit into ADCB copy the Application Context Data into a buffer provided by the caller return Application Context Data encode the Origin Data Record decode the Origin Data Record return the Origin Data Record set the application context of the task return Application Context Data set flag
DFHMNSM	Handles the following requests: STORAGE_NOTIFY
DFHMNSR	Handles the following requests: SET_MCT_SUFFIX SET_MONITORING INQ_MONITORING
DFHMNST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS

Module	Function
DFHMNSU	Handles monitoring domain subroutine requests of format MNSU: UPDATE_CATALOGUE MONITORING_DATASET_PUT WLM_CONNECT WLM_DISCONNECT WLM_REPORT WLM_NOTIFY PB_ALLOCATE PB_DELETE
DFHMNSVC	Provides SMFEWTM, WLM_CONNECT, WLM_DISCONNECT, WLM_REPORT, WLM_NOTIFY, WLM_PB_CREATE, and WLM_PB_DELETE authorized services with GTF tracing (GTRACE)
DFHMNTD	Handles the following request: SET_TRACKING_DATA INQUIRE_TRACKING_DATA
DFHMNTI	Handles the following request: NOTIFY
DFHMNTRI	Provides a trace interpretation routine for CICS dumps and traces
DFHMNUE	Provides a SET_EXIT_STATUS (services user exit) routine to enable or disable an exit
DFHMNXM	Handles the following requests: TRANSACTION_INITIALIZATION TRANSACTION_BIND TRANSACTION_TERMINATION

Exits

There is one global user exit point in the monitoring domain: XMNOUT. For more information, see Global user exit points (by function).

Managed platform (MP) domain

The managed platform domain manages policies.

Managed platform domain's specific gates

The specific gates provide access for other domains to functions that are provided by the MP domain.

MPAC gate, PERFORM_ACTION function

Performs an action when at least one policy rule is met.

Input Parameters

MPTAS_ADDR

A token identifying the MPTAS control block of the task.

ITEM_ID

The ID of the item for which at least one rule is met.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPDD gate, ADD_POLICY function

Adds a previously created policy (MPPPB) to a running CICS system. If a policy of the same name exists, it will be discarded before the new policy is added.

Input Parameters

POLICY_NAME
The name of the policy to add or replace.

POLICY
The token identifying the policy.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following value is returned when RESPONSE is EXCEPTION:
DUPLICATE_POLICY

The following value is returned when RESPONSE is DISASTER:
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPDD gate, ADD_POLICYSCOPE function

Adds a previously created policy scope (MPPMB) to a running CICS system. If a policy of the same name and basescope exists, then the chain of MPPMBs anchored in the MPPPB (policy) is searched for a policy scope for the same operation. If one exists an exception is raised (DUPLICATE_POLICYSCOPE), otherwise the new MPPPB is added to the end of the chain. If no policy of the same name and scope exists then NOT_FOUND is returned.

Input Parameters

POLICY_NAME
The name of the policy.

POLICYSCOPE
The token that identifies the policy.

Output Parameters

DUP_BUNDLE_NAME
The name of the bundle that contains the duplicate definition if REASON is DUPLICATE_POLICYSCOPE.

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following value is returned when RESPONSE is EXCEPTION:
DUPLICATE_POLICYSCOPE
NOT_FOUND

The following value is returned when RESPONSE is DISASTER:

ABEND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPDD gate, DISCARD_POLICY function

Removes a policy from a running CICS system.

Input Parameters

POLICY_NAME

The name of the policy to discard.

BUNDLE_NAME

The name of the bundle from which the policy was installed.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is EXCEPTION:

IN_USE
NOT_FOUND
BUNDLE_MISMATCH

The following values are returned when RESPONSE is DISASTER:

ABEND
UNKNOWN_POLICY

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPDD gate, DISCARD_POLICYSCOPE function

Removes a policy scope from a running CICS system.

Input Parameters

POLICY_NAME

The name of the policy.

PLATFORM_NAME

Optional parameter

The name of the platform.

APPLICATION_NAME

Optional parameter

The name of the application.

APPL_MAJOR_VER

Optional parameter

The application major version number.

APPL_MINOR_VER

Optional parameter

The application minor version number.

APPL_MICRO_VER

Optional parameter

The application micro version number.

OPERATION
Optional parameter

The name of the operation.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is EXCEPTION:
```
NOT_FOUND
```

The following values are returned when RESPONSE is DISASTER:
```
ABEND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPDD gate, FREE_POLICY function

Frees the storage for a policy that has been, possibly partially, created but is not part of a running CICS system and has no DD index entry.

Input Parameters

POLICY
A token identifying the policy

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```

The following value is returned when RESPONSE is DISASTER:
```
ABEND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPDD gate, INQUIRE_POLICYSCOPE function

Inquires on the state of a policy scope.

Input Parameters

POLICY_NAME
The name of the policy.

PLATFORM_NAME
Optional parameter

The name of the platform.

APPLICATION_NAME
Optional parameter

The name of the application.

APPL_MAJOR_VER
Optional parameter

The application major version number.

APPL_MINOR_VER
Optional parameter

The application minor version number.

APPL_MICRO_VER
Optional parameter

The application micro version number.

OPERATION
Optional parameter

The name of the operation.

Output Parameters

BUNDLE_TOKEN
An 8-byte token that represents the bundle that defines the policy scope.

BUNDLE_NAME
The name of the defining bundle.

RESOURCE_TOKEN
An 8-byte token that represents the policy scope resource.

REASON
The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
```

The following value is returned when RESPONSE is DISASTER:

```
ABEND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPIB gate, END_BROWSE_POLICY_RULE function

End a browse of policy rules in a bundle.

Input Parameters

BROWSE_TOKEN
The token that identifies the browse operation as returned by the associated START_BROWSE_POLICY_RULE function call.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPIB gate, GET_NEXT_POLICY_RULE function

Returns information about the next policy rule in the browse.

Input Parameters

BROWSE_TOKEN
The token that identifies the browse operation as returned by the associated START_BROWSE_POLICY_RULE function call.

Output Parameters

ACTION
Optional parameter

The action that is taken when this rules threshold is exceeded.

The values of this parameter are:
```
ABEND_TASK
EVENT
MESSAGE
```

ABEND_CODE
Optional parameter

The value of abend code if ACTION is abend_task, all blanks otherwise.

APPLICATION_NAME
Optional parameter

The name of the application this rule applies to.

APPL_MAJOR_VERSION
Optional parameter

The application major version this rule applies to, or -1 if it applies to all major versions.

APPL_MINOR_VERSION
Optional parameter

The application minor version this rule applies to, or -1 if it applies to all minor versions.

APPL_MICRO_VERSION
Optional parameter

The application micro version this rule applies to, or -1 if it applies to all micro versions.

BUNDLE_NAME
Optional parameter

The name of the bundle that contains the policy that defined this rule.

BUNDLE_ID
Optional parameter

The ID for the bundle that defines this rule, or all blanks if not set in manifest.

BUNDLE_MAJOR_VER
Optional parameter

The major version number of the bundle that defines this rule, or -1 if not set in manifest.

BUNDLE_MINOR_VER
Optional parameter

The minor version number of the bundle that defines this rule, or -1 if not set in manifest.

BUNDLE_MICRO_VER
Optional parameter

The micro version number of the bundle that defines this rule, or -1 if not set in manifest.

BUNDLE_DIRECTORY
Optional parameter

The zFS bundle directory that contains the policy XML.

CHANGEAGENT
Optional parameter

The agent that made the last change to the resource definition.

The values of this parameter are:
CREATESPI
CSDAPI
CSDBATCH
DREPAPI

CHANGEAGREL
Optional parameter

The CICS release that was running when the resource definition was last changed.

CHANGETIME
Optional parameter

The ABSTIME value that represents the time stamp when the resource definition was last changed.

CHANGEUSRID
Optional parameter

The user ID that ran the change agent.

COUNT
Optional parameter

The number of times this rules threshold has been exceeded since CICS restart.

DEFINESOURCE
Optional parameter

The source of the resource definition. The DEFINESOURCE value depends on the CHANGEAGENT value.

DEFINETIME
Optional parameter

The ABSTIME value that represents the time stamp when the resource definition was created.

EPADAPTER_NAME
Optional parameter

The name of the EP apdapter if the action is EVENT and the ADAPTER option is selected. Otherwise, this parameter is all blanks.

EPADAPTERSET_NAME
Optional parameter

The name of the EP apdapterset if the action is EVENT and the ADAPTER SET option is selected. Otherwise, this parameter is all blanks.

INSTALLAGENT
Optional parameter

The agent that installed the resource.

The values of this parameter are:
BUNDLE

INSTALLTIME
Optional parameter

The ABSTIME value that represents the time stamp when the resource was installed.

INSTALLUSRID
Optional parameter

The user ID that installed the resource.

ITEM_NAME
Optional parameter

The name of the item to which this rule relates.

LEVEL
Optional parameter

The level to which this rule was deployed.

The values of this parameter are:

```
REGION
PLATFORM
APPLICATION
APPLMAJORVER
APPLMINORVER
APPLMICROVER
OPERATION
```

MODEL_NAME
Optional parameter

The name of the MP model.

OPERATION_NAME
Optional parameter

The name of the application operation this rule applies to.

OPERATOR
Optional parameter

The value of the threshold operator.

The values of this parameter are:

```
GT
```

POLICY_NAME
Optional parameter

The name of the policy that defines this rule.
Optional parameter

PLATFORM_NAME
Optional parameter

The name of the platform this rule applies to.

RULE_TYPE
Optional parameter

The category to which this rule applies.

THRESHOLD
Optional parameter

The value of the threshold for this rule, that is: VALUE * UNIT

TIME
Optional parameter

The time, in STCK format; when the rules threshold was last exceeded.

VALUE
Optional parameter

The value as specified before the unit was applied.

UNIT
Optional parameter

The unit that was specified by the user when they defined the threshold.

The values of this parameter are:

NO_UNIT
BYTES
KILOBYTES
MEGABYTES
GIGABYTES
THOUSANDS
SECONDS
MILLISECONDS
MICROSECONDS

USERTAG
Optional parameter.

The value of the usertag for the policy that defines this rule.

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
BROWSE_END_EARLY

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPIB gate, START_BROWSE_POLICY_RULE function

Start a browse of the policy rules within a bundle.

Input Parameters

APPLICATION_NAME
Optional parameter.

The name of the application.

APPL_MAJOR_VERSION
Optional parameter

The major version of the application.

APPL_MINOR_VERSION
Optional parameter

The minor version of the application.

APPL_MICRO_VERSION
Optional parameter

The micro version of the application.

PLATFORM_NAME
Optional parameter

The name of the platform.

OPERATION_NAME
Optional parameter

The name of the operation.

Output Parameters

BROWSE_TOKEN
The token that identifies the browse operation.

REASON
The following values are returned when RESPONSE is EXCEPTION:
NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPMO gate, DEREGISTER_POLICY function

Updates one or more existing models to remove details of the rules from the named policy, when a policy (MPPPB) or a policy scope (MPPMB) is discarded.

Input Parameters

POLICY_NAME
The name of the policy.

POLICY
A token that identifies the policy.

OPERATION
Optional parameter

Restricts de-registration to only those models that are scoped on this named operation.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPMO gate, REGISTER_POLICY function

Creates a new MP model (MPMOD) or updates one or more existing models to include the rules that apply to a new policy when a new policy (MPPP) or a POLICYSCOPE (MPPMB) is installed.

Input Parameters

POLICY_NAME
The name of the policy.

POLICY
A token identifying the policy.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT

INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPMO gate, SELECT_MODEL function

Selects the most appropriate model for the current task.

Input Parameters

APPLICATION_NAME
Optional parameter.

The name of the application from the current tasks application context.

MAJOR_VERSION
Optional parameter.

The application major version from the current tasks application context.

MINOR_VERSION
Optional parameter.

The application minor version from the current tasks application context.

MICRO_VERSION
Optional parameter.

The application micro version from the current tasks application context.

OPERATION_NAME
Optional parameter.

The name of the application operation from the current tasks application context.

PLATFORM_NAME
Optional parameter.

The name of the platform from the current tasks application context.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPXM gate, BIND_XM_CLIENT function

The BIND_XM_CLIENT call flows from the transaction manager to the MP domain during transaction initialization.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPXM gate, INIT_XM_CLIENT function

The INIT_XM_CLIENT call flows from the transaction manager to the MP domain during transaction initialization.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MPXM gate, RELEASE_XM_CLIENT function

The RELEASE_XM_CLIENT call is made from the transaction manager to the MP domain during transaction termination. MP domain transaction lifetime resources are released.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Managed platform domain's generic gates

Table 33 summarizes the managed platform domain's generic gates. It shows the level-1 trace point IDs of the modules that provide the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 33. Managed platform domain's generic gates

Gate	Trace	Function	Format
MPDM	 0100 0101	 INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	MPDM

Modules

Managed Platform (MP) domain modules.

Module	Function
DFHMPAC	Policy Action manager when a policy threshold is exceeded
DFHMPDD	Management of domain indexes for all MP objects
DFHMPDM	Domain initialization and termination program
DFHMPDUF	Dump formatting
DFHMPIB	Implements functions that you can use to query the policy rules which apply to tasks in a CICS region that have a specified application context
DFHMPMO	Implementation of the functions of gate MPMO
DFHMPPP	Policy XSD bind file
DFHMPRL	RLCB callback gate
DFHMPTRI	Trace interpretation
DFHMPXM	Transaction attach and detach processing

Enqueue domain (NQ)

The NQ domain provides UOW based locking services. This is provided to the local clients FC, TD and TS. It also services the **ENQ** and **DEQ** application programming commands.

Enqueue domain's specific gates

The specific gates provide access for other domains to functions that are provided by the NQ domain.

NQED gate, DEQUEUE function

This functions releases an active enqueue owned by the current UOW from the specified enqueue pool.

Input Parameters

ENQUEUE_NAME1

Optional Parameter

A block (addr,len) identifying the name of the enqueue being released. Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name of the enqueue being released.

ENQUEUE_NAME2

Optional Parameter

A block (addr,len) identifying the second half of the enqueue name.

ENQUEUE_TOKEN

Token representing the enqueue that is to be released. Slightly better performance is achieved for callers that use the token method for releasing their enqueues.

MAX_LIFETIME

Optional Parameter

Indicates the maximum duration of the enqueue being released.

DISPATCHER_TASK

The enqueue will be released if it is held when a DEQUEUE_ALL request

is issued by the owning dispatcher task. This is the only value permitted when POOL_TOKEN is not supplied on the call.

TRANSACTION
The enqueue was acquired with a duration of the last UOW of the current transaction.

UOW
The enqueue was acquired with a duration of the current UOW. This is the default value when not supplied on the call.

Values for the parameter are:
```
DISPATCHER_TASK
TRANSACTION
UOW
```

POOL_TOKEN
Optional Parameter

Token representing enqueue pool from which the enqueue is to be released.

Output Parameters

REASON
The values for the parameter are:
```
ENQUEUE_LOCKED
ENQUEUE_NOT_OWNED
INVALID_POOL_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQED gate, ENQUEUE function

This functions obtains an enqueue from the specified enqueue pool in active state.

Input Parameters

ENQUEUE_NAME1
Optional Parameter

A block (addr,len) identifying the name of the enqueue being released. Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name of the enqueue being released.

ENQUEUE_NAME2
Optional Parameter

A block (addr,len) identifying the second half of the enqueue name.

MAX_LIFETIME
Optional Parameter

Indicates the maximum duration of the enqueue.

DISPATCHER_TASK
The enqueue will be released if it is held when a DEQUEUE_ALL request is issued by the owning dispatcher task. This is the only value permitted when POOL_TOKEN is not supplied on the call.

TRANSACTION
The enqueue will be acquired with a duration of the last UOW of the current transaction.

UOW
The enqueue will be acquired with a duration of the current UOW. This is the default value when not supplied on the call.

Values for the parameter are:

DISPATCHER_TASK
TRANSACTION
UOW

POOL_TOKEN
Optional Parameter

Token representing enqueue pool from which the enqueue is to be allocated.

PURGEABLE
Optional Parameter

Indicates if the task is purgeable.

Values for the parameter are:
NO
YES

SHUNT_ACTION
Optional Parameter

Indicates the action that is to be performed if this UOW is shunted whilst it owns the enqueue. This parameter acts as an override, if not supplied then the default shunt action specified when the pool was created is assumed for this enqueue request.
The shunt action is only applicable to UOW lifetime enqueues. An error is diagnosed if this parameter is supplied on a request for a transaction lifetime enqueue. The possible overrides are as follows:
RELEASE

The enqueue will be released if the UOW is shunted.

RETAIN
The enqueue will be retained if the UOW is shunted.

IGNORE
The shunt will be ignored. The enqueue will remain in the same state as it is currently held in.

Values for the parameter are:
IGNORE
RELEASE
RETAIN

WAIT
Optional Parameter

Indicates whether the caller wants to wait if the requested enqueue is currently held in the pool by a different UOW. The possible values are as follows:

NO The ENQUEUE_BUSY exception is returned to the caller if the enqueue is busy.

YES
The caller will be suspended if the enqueue is busy. This is the default value when not supplied on the call.

Note that callers specifying WAIT(NO) should still expect to suspend for the NQ domain lock.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
ENQUEUE_BUSY

ENQUEUE_DISABLED
ENQUEUE_LOCKED
INVALID_PHASE
INVALID_POOL_TOKEN
LIMIT_EXCEEDED
SHUNT_ACTION_NOT_EXPECTED
SYSENQ_FAILURE
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUPLICATE_REQUEST
Optional Parameter

When RESPONSE(OK) is returned, indicates whether the caller already owned the enqueue or not:

Values for the parameter are:
NO
YES

ENQUEUE_TOKEN
Optional Parameter

A token returned to represent the enqueue that has been successfully returned. The token can then be used on the corresponding DEQUEUE request.

NQIB gate, END_BROWSE_ENQUEUE function

This functions terminates a browse of the enqueues.

Input Parameters

BROWSE_TOKEN
The token for the browse that is to be terminated.

Output Parameters

REASON
The values for the parameter are:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQIB gate, GET_NEXT_ENQUEUE function

This functions returns information about the next enqueue owner or waiter in a browse.

Input Parameters

BROWSE_TOKEN
The token for the current browse.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON
Values for the parameter are:
ABEND
LOOP

BROWSE_END
INVALID_BROWSE_TOKEN

ENQUEUE_NAME_OUT
A buffer into which the enqueue name is returned. The caller specifies the address and maximum length of the data area into which the enqueue name will be returned. If the enqueue name is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in *enqueue_name_out_n*.

ENQUEUE_NAME2_LENGTH

The length of the second part of the enqueue name if the enqueue was originally specified in two parts (i.e. ENQUEUE_NAME1 and ENQUEUE_NAME2).

If the ENQUEUE_NAME2 parameter wasn't originally specified for this enqueue then zero will be returned.

ENQUEUE_TOKEN
Token returned only when the enqueue is owned by the caller. Parameter is set to zero for all other enqueues returned on the browse.

INTERPRETER_ADDRESS

The address of a routine which should be called with the INTERPRET_ENQUEUE function in order to interpret the enqueue for the EXEC CICS INQUIRE UOWENQ command.

If a zero address is returned then the enqueue isn't to be returned by the INQUIRE UOWENQ command.

RESOURCE_FILTER
The resource filter as specified in the RESOURCE option on the ENQUIRE UOWENQ command.

RESOURCE_FILTER_LEN
The length of the RESOURCE_FILTER parameter.

LOCAL_UOWID
The local UOWID of the UOW which owns or is waiting for the enqueue.

NUM_LOCKED_FAILURES
Returns the number of failed requests for this enqueue whilst it is held in retained state.

NUM_WAITERS
The number of transactions waiting for this enqueue.

POOL_NAME
The name of the pool containing the enqueue.

POOL_TOKEN
Token which identifies the pool which the enqueue owner or waiter belongs.

RELATION
Indicates whether the data being returned is associated with owner or a UOW waiting for the enqueue.

OWNER
The data is associated with the owner of the returned enqueue.

WAITER
The data is associated with a waiter of the returned enqueue.

SHUNT_ACTION
The action that would be performed to this enqueue should its owning UOW be shunted. The possible values are as follows:

RELEASE
The enqueue will be released.

RETAIN
The enqueue will be retained.

IGNORE
The shunt will be ignored and the enqueue will remain in the same state.

STATE
The state that the enqueue is held in.

ACTIVE
The enqueue is held in active state.

RETAINED
The enqueue is held in retained state.

TRANSACTION_LIFETIME

For an enqueue returned with RELATION(OWNER) the number of times it is held with TRANSACTION lifetime.

For an enqueue returned with RELATION(WAITER) a count of one indicates that the enqueue was requested with TRANSACTION lifetime.

UOW_LIFETIME

For an enqueue returned with RELATION(OWNER) the number of times it is held with UOW lifetime.

For an enqueue returned with RELATION(WAITER) a count of one indicates that the enqueue was requested with UOW lifetime.

NQIB gate, INQUIRE_ENQUEUE function

This functions returns information about a particular enqueue. Note that the pool containing the enqueue must be passed since it is a logical extension to the enqueue name.

Input Parameters

POOL_TOKEN
The token identifying the pool from which the enqueue being inquired about belongs.

ENQUEUE_TOKEN
Token representing the enqueue that is being inquired upon.

ENQUEUE_NAME1
A block (addr,len) identifying the name of the enqueue be inquired upon. Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name of the enqueue being inquired upon.

ENQUEUE_NAME2
Optional Parameter

A block (addr,len) identifying the second half of the enqueue name.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON
Values for the parameter are:

```
ABEND
LOOP
ENQUEUE_NOT_FOUND
INVALID_BROWSE_TOKEN
```

ENQUEUE_NAME_OUT

A buffer into which the enqueue name is returned. The caller specifies the address and maximum length of the data area into which the enqueue name

will be returned. If the enqueue name is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in *enqueue_name_out_n*.

Typically this parameter will only be of interest to callers inquiring by enqueue token.

LOCAL_UOWID
The local UOWID of the UOW which owns or is waiting for the enqueue.

NUM_LOCKED_FAILURES
Returns the number of failed requests for this enqueue whilst it is held in retained state.

NUM_WAITERS
The number of transactions waiting for this enqueue.

POOL_NAME
The name of the pool containing the enqueue.

TRANSACTION_LIFETIME
The number of times the enqueue is held with TRANSACTION lifetime.

STATE
The state that the enqueue is held in.

ACTIVE
The enqueue is held in active state.

RETAINED
The enqueue is held in retained state.

UOW_LIFETIME
The number of times the enqueue is held with UOW lifetime.

NQIB gate, START_BROWSE_ENQUEUE function

This function initiates a browse of all enqueues currently in the system or currently associated with a given UOW.

The browse returns both enqueue owners and enqueue waiters. The RELATION output parameter on GET_NEXT_ENQUEUE indicates whether the data being returned is associated with the enqueue owner or a UOW waiting for that enqueue.

When a system wide browse is initiated the first enqueue in the system is returned with RELATION(OWNER). If the enqueue has any waiters then the same enqueue will be returned again for each of the waiters but this time with RELATION(WAITER). The data returned will be that associated with that particular waiter. After the last waiter has been returned the next owned enqueue will be returned.

If the browse is restricted to only a particular UOW then only the enqueues that UOW owns will be returned. If the UOW is waiting for an enqueue this will also be returned.

The order in which the enqueues are returned is undefined, however enqueue waiters are always returned consecutively after their enqueue owner.

As with other types of CICS browses the state isn't locked for the duration of the browse. Thus for example, there is no guarantee that the owner returned on a previous GET_NEXT_ENQUEUE is still the owner by the time each of its waiters are returned.

Input Parameters

ENQSCOPE
Optional Parameter

For sysplex scope enqueues, the 4-character scope name that qualifies all ENQUEUE requests issued by this CICS region.

ENQUEUE_NAME1

Optional Parameter

The first part of a two-part enqueue name.

LOCAL_UOWID

Optional Parameter

Identifies the unit of work if the browse is to be restricted to only those enqueues owned and being waited for by a particular UOW.
If omitted then browse will return all enqueue owners and waiters in the system.

STABLE_ENQUEUES

Optional Parameter

Specifies that the caller will complete the browse without issuing any further ENQ or DEQ requests. Applies only if LOCAL_UOWID is also specified and names the caller's own UOWID.

Output Parameters

REASON

The values for the parameter are:

`NO_UOW_ENVIRONMENT`

BROWSE_TOKEN

Token to be used by the caller on subsequent operations associated with this browse.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQNQ gate, CREATE_ENQUEUE_POOL function

This function creates a separate enqueue pool for the caller. A token is returned which the caller specifies on all requests associated with that pool.

Input Parameters

ERROR_LEVEL

Indicates the severity of the error response that is to be returned for the following errors made while using this pool:

- DEQUEUE
  - Enqueue_not_owned
  - Enqueue_locked
- REACQUIRE_ENQUEUE
  - Enqueue_locked
  - Enqueue_active
- DEACTIVATE
  - Enqueue_not_owned
  - Enqueue_not_active

The possible values for ERROR_LEVEL are as follows:

EXCEPTION_RESPONSE

These errors are to be returned with an exception response.

INVALID_RESPONSE

These errors are to be returned with an invalid response. (i.e. FFDC is to be performed).

Note: It is expected that only the EXEC and the KC enqueue pools will specify EXCEPTION_RESPONSE since the DFHKC service previously used by them allowed these sorts of error to go by undetected.

Values for the parameter are:

```
EXCEPTION_RESPONSE
INVALID_RESPONSE
```

EXEC_INTERPRETER

Indicates how enqueues belonging to the enqueue pool are to be interpreted by the **EXEC CICS INQUIRE UOWENQ** command. The possible values are as follows:

NONE

No interpreter has been supplied so enqueues belonging to this pool will be ignored by the INQUIRE UOWENQ command.

DEFAULT

Enqueues are to be returned by the INQUIRE UOWENQ command. The default NQ domain interpreter will be called to perform the interpretation. This will map the outputs of the INQUIRE UOWENQ command as follows:

TYPE

Will be the CVDA corresponding to the ENQUEUE_TYPE parameter supplied on this call.

RESOURCE

Will be ENQUEUE_NAME1 as supplied on the NQED_ENQUEUE function.

QUALIFIER

Will be ENQUEUE_NAME2 if supplied on the NQED_ENQUEUE function. If not then no QUALIFIER data will be returned.

OWN

Enqueues are to be returned by the INQUIRE UOWENQ command. A routine provided by the pool owner will perform the interpretation. In this case the entry point of the routine must be passed in the INTERPRETER_ADDR parameter.

Note: The routine will be called by a kernel subroutine call, not by a domain call. Consequently it will execute in the domain of the caller (i.e. AP domain).

Values for the parameter are:

```
DEFAULT
NONE
OWN
```

EXPECTED_NAME_LENGTH

The expected length for enqueue names in the pool.

- For pools with fixed length enqueue names this should be the length of the names that are going to be enqueued upon.
- For pools that are to contain variable length enqueue names this should be a length that would satisfy most of the requests to be made in the pool.

Note that there is no maximum length for enqueue names. However, requests will only be handled inline if the length of the enqueue name is less than or equal to the EXPECTED_NAME_LENGTH. The inline macro only copes with names of less than or equal to 256 characters. For this reason an error will be diagnosed if a value of greater than 256 is specified for this parameter.

POOL_NAME

The eight character name of the new enqueue pool.

SHUNT_ACTION

Indicates the default action that is to be performed to UOW lifetime enqueues in this pool if their owning UOW is shunted. Note that most enqueue pools will require the same action to be performed for all enqueues in that pool. However, the ENQUEUE function allows this default to be overridden for particular enqueue requests.

The possible values are as follows:

RELEASE
The enqueue(s) will be released if the owning UOW is shunted.

RETAIN
The enqueue(s) will be retained if the owning UOW is shunted.

IGNORE
The shunt will be ignored. The enqueue(s) will remain in the same state as currently held in.

Transaction lifetime enqueues are automatically released when a shunt occurs.

Values for the parameter are:

```
IGNORE
RELEASE
RETAIN
```

ENQUEUE_TYPE
Optional Parameter

The enqueue type that is to be returned by the default interpreter. Should only be supplied for pools which specify a value of DEFAULT for the EXEC_INTERPRETER parameter. The possible values map onto the CVDA values for the TYPE field as detailed under the **EXEC CICS INQUIRE UOWENQ** command.

Values for the parameter are:

```
DATASET
DISPATCHER
EXECENQ
EXECENQADDR
EXECENQPLEX
FILE
TDQUEUE
TSQUEUE
```

OWN_INTERPRETER_ADDRESS
Optional Parameter

Entry point of interpreter routine for this pool. Should only be supplied for pools which specify a value of OWN for the EXEC_INTERPRETER parameter.

Output Parameters

REASON
The values for the parameter are:

```
DUPLICATE_POOL_NAME
ENQUEUE_TYPE_EXPECTED
INTERPRETER_ADDR_EXPECTED
INVALID_NAME_LENGTH
```

POOL_TOKEN
Token returned which identifies the newly created enqueue pool.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQNQ gate, DEACTIVATE function

This function converts an active enqueue into retained state. The caller must already own the enqueue.

Input Parameters

POOL_TOKEN
Token representing enqueue pool from which the enqueue is to be deactivated.

ENQUEUE_TOKEN
Token representing the enqueue that is to be deactivated. Slightly better performance is achieved for callers that use the token method for this function.

ENQUEUE_NAME1
A block (addr,len) identifying the name of the enqueue to be deactivated. Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name of the enqueue to be deactivated.

ENQUEUE_NAME2
Optional Parameter

A block (addr,len) identifying the second half of the enqueue name.

Output Parameters

REASON
The values for the parameter are:

```
ENQUEUE_NOT_ACTIVE
ENQUEUE_NOT_OWNED
INVALID_POOL_TOKEN
TRANSACTION_ENQUEUE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQNQ gate, DEQUEUE_TASK function

Dequeue a task that was previously enqueued.

Input Parameters

ENQUEUE_TOKEN
The token that was returned on the corresponding ENQUEUE request.

Output Parameters

REASON
The values for the parameter are:

```
ENQUEUE_NOT_OWNED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQNQ gate, INTERPRET_ENQUEUE function

This function interprets the passed enqueue before it being returned by the EXEC CICS INQUIRE UOWENQ command. The function takes the enqueue to be interpreted as input and returns ENQUEUE_TYPE, RESOURCE and QUALIFIER to the caller (EXEC layer).

Each enqueue pool can either

- not have an interpreter and consequently not have its enqueues returned by the INQUIRE UOWENQ command
- rely upon a default interpreter supplied by NQ domain, (DFHNQIE)

- supply its own interpreter routine.

This is specified when the pool is created.

Input Parameters

ENQUEUE_NAME
A block (addr,len) identifying the full name of the enqueue to be interpreted.

ENQUEUE_NAME2_LENGTH
The length of the second part of the enqueue name if the enqueue was originally specified in two parts (i.e. ENQUEUE_NAME1 and ENQUEUE_NAME2). If the ENQUEUE_NAME2 parameter wasn't originally specified for this enqueue then this will contain zero.

POOL_NAME
Name of the pool containing the enqueue to be interpreted. Note that an interpreter may interpret enqueues from more than one pool.

POOL_TOKEN
Token corresponding to the pool containing the enqueue to be interpreted

QUALIFIER_BUFFER

A buffer into which the data for the QUALIFIER field is returned. The caller specifies the address and maximum length of the data area into which the QUALIFIER data will be returned. If the data is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in *qualifer_buffer_n*.

If there is no QUALIFIER data then no data should be returned and the length of the data (*qualifier_buffer_n*) should be returned as zero.

RESOURCE_BUFFER
A buffer into which the data for the RESOURCE field is returned. The caller specifies the address and maximum length of the data area into which the RESOURCE data will be returned. If the data is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in *resource_buffer_n*.

Output Parameters

REASON
The values for the parameter are:
```
INVALID_ENQUEUE
```

ENQUEUE_TYPE
The TYPE of the enqueue being returned. The values map onto the CVDA values for the TYPE field as detailed under the EXEC CICS INQUIRE UOWENQ command.

Values for the parameter are:
```
DATASET
DISPATCHER
EXECENQ
EXECENQADDR
EXECENQPLEX
FILE
TDQUEUE
TSQUEUE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQNQ gate, REACQUIRE_ENQUEUE function

NQ domain doesn't recover enqueues over a CICS restart. Instead resource owners use this function to reacquire enqueues that were held by inflight and indoubt UOWs.

The enqueue can be reacquired in either active or retained state. The calling UOW must currently be shunted.

No MAX_LIFETIME input is provided since such enqueues are only ever associated with a single UOW.

The same rules as documented for the mainline ENQUEUE function apply to the shunt action that will be associated with the reacquired enqueue.

Input Parameters

POOL_TOKEN
Token representing enqueue pool from which the enqueue is to be allocated.

STATE
The state that the enqueue is to be reacquired in.

Values for the parameter are:
```
ACTIVE
RETAINED
```

ENQUEUE_NAME2
Optional Parameter

A block (addr,len) identifying the second half of the enqueue name.

SHUNT_ACTION
Optional Parameter

Indicates the action that is to be performed if the UOW reacquiring the enqueue is shunted again. This parameter acts as an override, if not supplied then the default shunt action specified when the pool was created is assumed for this request.

Values for the parameter are:

RELEASE
The enqueue will be released if the UOW is shunted again.

RETAIN
The enqueue will be retained if the UOW is shunted again.

IGNORE
The shunt will be ignored. The enqueue will remain in the same state as it is currently held in.

Output Parameters

REASON
The values for the parameter are:
```
CALLER_NOT_SHUNTED
ENQUEUE_ACTIVE
ENQUEUE_LOCKED
INVALID_POOL_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ENQUEUE_TOKEN
Optional Parameter

Token returned to represent the enqueue that has been successfully reacquired.

NQNQ gate, SET_NQRNAME_LIST function

This function is called from three places in DFHNQRN.

The function is called at the following points.

discard_enqmodel
If nqrmodel delete is set, then the specified nqrmodel is removed from nqrname_list

add_replace_enqmodel
If nqrmodel add is set then the specified nqrmodel is added to nqrname_list.

set_nqrmodel
if neither delete or add is set then the specified nqrmodel is set disabled.

Input Parameters

MODEL_TOKEN
The address of the nqrmodel to be set or added to nqrname_list.

POOL_TOKEN
The pool to be searched for matching enqueues

POOL_TWO
Optional Parameter

An optional second pool to be searched for matching enqueues

Output Parameters

REASON
The values for the parameter are:

```
FREE_NQRMODEL
NQRMODEL_NOT_FOUND
```

FREE_TOKEN
Address of Model being removed.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQRN gate, ADD_REPLACE_ENQMODEL function

This function adds an ENQMODEL resource definition to both the NQRN directory (keyed by enqmodel name) and to the NQRNAME_LIST (keyed by the variable length NQRNAME).

If the ENQMODEL resource already exists, the entry is replaced. The replace is a discard, then add operation.

If an attempt is made to create a deep enqmodel nesting, or if another enqmodel with the same nqrname is already installed, then message DFHNQ0106 is issued and a 'DUPLICATE_NQRNAME' exception is returned.

Input Parameters

CALLER
COLDINST, RDOINST or RESTART indicating A cold start, An online install or The input is in the MODEL_TOKEN respectively.

Values for the parameter are:

```
COLDINST
RDOINST
RESTART
```

CATALOG
Indicates whether the record should be cataloged.

Values for the parameter are:
NO
YES

ENQMODEL
The 8-character identifier of the resource to be added.

MODEL_TOKEN
The address of the record obtained from the catalog to be restored.

NQRNAME
Optional Parameter

A buffer giving the 1 to 255 character name and length of the ENQ name or stem* to be added.

SCOPE
Optional Parameter

The 4-character scope identifier for the resource. If omitted or specified as blanks, matching ENQs will have LOCAL scope.

STATE
Optional Parameter

The state in which to install the ENQMODEl resource. If omitted, ENABLED is assumed.

Values for the parameter are:
DISABLED
ENABLED

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ACQUIRE_LOCK_FAILED
CATALOG_WRITE_FAILED
DIRECTORY_ADD_FAILED
DIRECTORY_DELETE_FAILED
GETMAIN_FAILED
RELEASE_LOCK_FAILED

The following values are returned when RESPONSE is EXCEPTION:
DUPLICATE_ENABLED
DUPLICATE_NQRNAME

The following values are returned when RESPONSE is INVALID:
INVALID_PARAMETERS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ENQMODEL_OUT
Optional Parameter

The name of an existing resource that is already installed, and not disabled, that prevents the successful completion of this operation.

NQRN gate, COMMIT_ENQMODEL function

Commit the ENQMODEL to the catalog.

Input Parameters

COMMIT_TOKEN
Token for catalog writes.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
CATALOG_WRITE_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQRN gate, DISCARD_ENQMODEL function

Remove an enqmodel definition from both the NQRN directory and from the NQRNAME_LIST.

If the enqmodel is not installed, an 'ENQMODEL_NOT_FOUND' exception is returned.

The ENQMODEL is put into the WAITING state until there are no enqueues in the local system which match the ENQNAME pattern. It is then removed from the local system.

Input Parameters

ENQMODEL
The 8-character identifier of the resource to be discarded.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ACQUIRE_LOCK_FAILED
CATALOG_DELETE_FAILED
RELEASE_LOCK_FAILED

The following values are returned when RESPONSE is EXCEPTION:
ENQMODEL_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQRN gate, END_BROWSE_ENQMODEL function

End a browse operation on a set of ENQMODEL resources.

Input Parameters

BROWSE_TOKEN
A token that identifies the browse operation. See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
DIRECTORY_END_BROWSE_ERR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQRN gate, GET_NEXT_ENQMODEL function

In a browse operation, retrieve the next ENQMODEL

Input Parameters

BROWSE_TOKEN

Browse token returned by the START_BROWSE function.

NQRNAME

Optional Parameter

A buffer giving the 1 to 255 character name and length of the ENQ name or stem.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ACQUIRE_LOCK_FAILED
DIRECTORY_GET_NEXT_ERR
RELEASE_LOCK_FAILED
```

The following values are returned when RESPONSE is EXCEPTION:

```
NO_MORE_DATA
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ENQMODEL

Optional Parameter

The 4-character scope identifier for the resource.

SCOPE

Optional Parameter

The 4-character scope identifier for the resource.

STATE

Optional Parameter

The current state of the ENQMODEL.

NQRN gate, INQUIRE_ENQMODEL function

Uses directory DDLO_LOCATE to retrieve information about a specified enqmodel definition in the NQRN directory.

If found, it returns the 1 to 255 character NQRNAME, the 4-character SCOPE name, the enqmodel STATE and ann OK RESPONSE. Otherwise it returns an EXCEPTION REASON(ENQMODEL_NOT_FOUND).

Input Parameters

ENQMODEL

The 8-character identifier of the entry to be returned.

NQRNAME

Optional Parameter

A buffer returning the 1 to 255 character name and length of the ENQ name or generic stem*

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ACQUIRE_LOCK_FAILED
DIRECTORY_LOCATE_FAILED
```

RELEASE_LOCK_FAILED

The following values are returned when RESPONSE is EXCEPTION:
ENQMODEL_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SCOPE
Optional Parameter

Returns the 4-character scope identifier for the resource. Four blanks indicates that the enqueue has local scope.

STATE
Optional Parameter

Values for the parameter are:

ENABLED
Matching ENQ/DEQ requests should be processed.

DISABLED
Matching ENQ/DEQ requests should be rejected, and the issuing task abended abcode ENQ_DISABLED.

WAITING
There are INSTALL, CREATE, or DISCARD requests waiting to be processed. Matching ENQ/DEQ requests should be rejected, and the issuing task abended abcode ENQ_DISABLED.

NQRN gate, INQUIRE_NQRNAME function

Determine if an enqueue name entry exists.

Input Parameters

MSG0105
A binary value that indicates whether message DFHNQ0105 is to be issued if the matching enqmodel is disabled or in the waiting state.

Values for the parameter are:
YES
NO

NQRNAME
The name of the enqueue name entry

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ACQUIRE_LOCK_FAILED
RELEASE_LOCK_FAILED

The following values are returned when RESPONSE is EXCEPTION:
NQRNAME_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SCOPE
The 4-character scope identifier for the resource.

STATE
The current state of the ENQMODEL

Values for the parameter are:
ENABLED
DISABLED

NQRN gate, REMOVE_ENQMODEL function

Remove an ENQMODEL object.

Input Parameters

MODEL_TOKEN

A token that represents the ENQMODEL to be removed.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQRN gate, RESTORE_DIRECTORY function

Restore the NQRN directory from the global catalog.

Input Parameters

COLD_START

A binary parameter indicating whether the request is made in cold start processing.

The values for the parameter are:

```
NO
YES
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
CATALOG_PURGE_FAILED
CATALOG_READ_FAILED
DIRECTORY_ADD_FAILED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQRN gate, SET_ENQMODEL function

This function uses directory DDLO_LOCATE to see if an enqmodel entry exists in the NQRN directory. If found, it calls SET_ENQMODEL to enable or disable the entry. Otherwise it returns an EXCEPTION REASON(ENQMODEL_NOT_FOUND).

Enqmodels forming nested generic nqrnames must be enabled in order, from the most to the least specific. I.e. A more specific enqmodel may not be enabled if a less specific enqmodel is enabled. If attempted, msg NQ0107 is issued and EXCEPTION 'DUPLICATE_ENABLED' is returned to the caller.

You cannot enable/disable an enqmodel which is in the waiting state. If attempted, EXCEPTION 'ENQMODEL_WAITING' is returned to the caller.

Input Parameters

ENQMODEL

The 8-character identifier of the entry to be enabled/disabled.

STATE

The intended state of the ENQMODEL.

Values for the parameter are:

```
DISABLED
ENABLED
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ACQUIRE_LOCK_FAILED
CATALOG_UPDATE_FAILED
DIRECTORY_LOCATE_FAILED
RELEASE_LOCK_FAILED
```

The following values are returned when RESPONSE is EXCEPTION:

```
DUPLICATE_ENABLED
ENQMODEL_NOT_FOUND
ENQMODEL_WAITING
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NQRN gate, START_BROWSE_ENQMODEL function

Start a browse operation on a set of ENQMODEL objects.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
DIRECTORY_START_BROWSE_ERR
```

BROWSE_TOKEN

See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Enqueue domain's generic gates

Table 34 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 34. Enqueue domain's generic gates

Gate	Trace	Function	Format
DMDM	NQ 0101 NQ 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	NQ 0501 NQ 0502	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

The domain manager gates perform normal internal state initialization and termination functions.

Enqueue domain's call-back gates

Table 35 on page 772 summarizes the domain's call-back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 35. Enqueue domain's call-back gates

Gate	Trace	Function	Format
RMRO	NQ 0201 NQ 0202	PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT	RMRO

PERFORM_PREPARE is a no-op. PERFORM_COMMIT releases enqueues. PERFORM_SHUNT make active enqueues retained. PERFORM_UNSHUNT makes retained enquires active.

Modules

Module	Function
DFHNQDM	Handles the following requests: INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHNQDUF	Formats the NQ domain control blocks in a CICS system.
DFHNQED	Handles the following requests: ENQUEUE DEQUEUE
DFHNQEDI	Inline version of DFHNQED
DFHNQIB	Handles the following requests: INQUIRE_ENQUEUE START_BROWSE_ENQUEUE GET_NEXT_ENQUEUE END_BROWSE_ENQUEUE
DFHNQIE	Handles the following requests: INTERPRET_ENQUEUE
DFHNQNQ	Handles the following requests: CREATE_ENQUEUE_POOL REACQUIRE_ENQUEUE DEACTIVATE SET_NQRNAME_LIST DEQUEUE_TASK
DFHNQRN	Handles the following requests: INQUIRE_NQRNAME ADD_REPLACE_ENQMODEL DISCARD_ENQMODEL REMOVE_ENQMODEL INQUIRE_ENQMODEL START_BROWSE_ENQMODEL GET_NEXT_ENQMODEL END_BROWSE_ENQMODEL SET_ENQMODEL COMMIT_ENQMODEL RESTORE_DIRECTORY
DFHNQST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHNQTRI	Provides a trace interpretation routine for CICS dumps and traces.

Exits

The XNQEREQ and XNQEREQC global user exit points are invoked respectively before and after each EXEC ENQ or DEQ request to the NQ domain.

Object transaction service domain (OT)

The object transaction service domain provides services to manage OTS transactions.

Specific gates for the object transaction service domain

The specific gates provide access for other domains to functions that are provided by the OT domain.

OTCO gate, FORGET function

The FORGET function of the OTCO gate is used to signal the fact that the obligation to the coordinator has been discharged.

Input Parameters

COORDINATOR_TOKEN
Token representing the coordinator OTS resource.

UOWID
identification of the local logical unit of work managing the OTS transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`LINK_UNKNOWN`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTCO gate, RESYNC function

Resynchronize an OTS transaction.

Input Parameters

DECISION
Specifies whether the transaction should be committed or rolled back.

Values for the parameter are:
`COMMIT`
`ROLLBACK`

UOWID
The unit-of-work ID of the transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`COORDINATOR_NOT_FOUND`

HEURISTIC
A binary value indicating whether a heuristic decision has been taken for the transaction.

Values for the parameter are:
`NO`
`YES`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTCO gate, SET_COORDINATOR function

Associate a link with the current task's unit-of-work to represent a remote co-ordinator.

Input Parameters

HOST_BLOCK
Pointer to and length of a character string that represents the identity of the system, which contains the co-ordinator instance. Maximum supported length of this parameter is 4096 bytes.

IOR_BLOCK
Pointer to and length of a character string that represents the co-ordinator instance in the host system. Maximum supported length of this parameter is 4096 bytes.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
HOST_TOO_LONG
IOR_TOO_LONG

COORDINATOR_TOKEN
A token representing the coordinator.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTCO gate, SET_LAST_AGENT function

Designate an object as the last agent of this part of an OTS transaction.

Input Parameters

COORDINATOR_TOKEN
The token that represents the coordinator of the transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
LINK_UNKNOWN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTCP gate, RESYNC_COORDINATOR function

Resynchronize with the coordinator in an OTS transaction.

Input Parameters

IOR_BLOCK
Block containing the IOR of the OTS resource

LOGICAL_SERVER
The logical server

PUBLIC_ID
The OTS public ID of the transaction

UOW_ID
The unit-of-work ID of the transaction.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
COMM_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTCP gate, RESYNC_SUBORDINATE function

Resynchronize with the subordinate in an OTS transaction.

Input Parameters

DECISION

The commit or roll back decision for the transaction

Values for the parameter are:

```
COMMIT
ROLLBACK
```

IOR_BLOCK

Block containing the IOR of the OTS resource

LOGICAL_SERVER

The logical server associated with the request.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
COMM_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTRS gate, FORGET_TRANSACTION function

Initiate forget processing for an OTS resource.

Input Parameters

IOR_BLOCK

Block containing the IOR of the OTS resource

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTRS gate, PERFORM_RESYNC function

Resynchronize all OTS resources.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ALREADY_IN_RESYNC
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTRS gate, SET_REMOTE_STATUS function

Set the status of a remote OTS resource.

Input Parameters

IOR_BLOCK
Block containing the IOR of the OTS resource

STATUS
The intended status of the remote object.

Values for the parameter are:
COMMIT
HEURISTIC_COMMIT
HEURISTIC_HAZARD
HEURISTIC_MIXED
HEURISTIC_ROLLBACK
ROLLBACK

Output Parameters

ALREADY_HEURISTIC
Indicates whether the remote object has already subject to a heuristic decision to commit or roll back.

Values for the parameter are:
COMMIT
NO
ROLLBACK

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTSU gate, ADD_SUBORDINATE function

The ADD_SUBORDINATE function of the OTSU gate is used add a subordinate participant to the OTS transaction.

Input Parameters

HOST_BLOCK
Block containing the name of the TCPIP host where the subordinate OTS resource resides.

IOR_BLOCK
Block containing the IOR of the OTS Resource that is being added as a subordinate participant in the OTS transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ADD_LINK_FAILED

The following values are returned when RESPONSE is EXCEPTION:
HOST_TOO_LONG
IOR_TOO_LONG

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SUBORDINATE_TOKEN
token representing the added Resource.

OTSU gate, FORGET function

The FORGET function of the OTSU gate is used signal the fact that the obligation to the subordinate resource has been discharged.

Input Parameters

SUBORDINATE_TOKEN
Token representing the subordinate OTS resource.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
INBOUND_FLOW_FAILED

The following values are returned when RESPONSE is EXCEPTION:
UNKNOWN_SUBORDINATE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTSU gate, RESYNC function

The RESYNC function of the OTSU gate initiates the resynchronization protocol with the subordinate resource identified by the given IOR.

Input Parameters

IOR_BLOCK
Block containing the IOR of the OTS Resource that is being added as a subordinate participant in the OTS transaction.

UOWID
Identification of the local logical unit of work managing the OTS transaction.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

UOW_STATUS
The status of the unit of work.

Values for the parameter are:
COMMITTED
IN_DOUBT
IN_FLIGHT
ROLLED_BACK

OTSU gate, SET_VOTE function

The SET_VOTE function of the OTSU gate is used record the vote that results from a PREPARE method being invoked on the OTS Resource represented by the given SUBORDINATE_TOKEN.

Input Parameters

SUBORDINATE_TOKEN
Token representing the subordinate OTS resource.

VOTE
The vote resulting from the first phase of syncpoint on the subordinate resource.

Values for the parameter are:
HEURISTIC_COMMIT
HEURISTIC_HAZARD
HEURISTIC_MIXED
HEURISTIC_ROLLBACK
NO
READ_ONLY

YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
RECORD_VOTE_FAILED

The following values are returned when RESPONSE is EXCEPTION:
INVALID_VOTE
UNKNOWN_SUBORDINATE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTTR gate, BEGIN_TRAN function

The BEGIN_TRAN function of the OTTR gate is used to create a new OTS transaction.

Input Parameters

LOGICAL_SERVER
The name of the logical server within which the transaction is executing.

PUBLIC_ID
The Request Stream public identifier associated with the transaction.

TID_BUFFER_OUT
The OTS transaction identifier (TID) of the transaction created.

TIMEOUT
Optional Parameter

The OTS transaction timeout value.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
TID_TOO_LONG
UOW_ROLLEDBACK

BQUAL_LEN
The batch qualifer length of the OTS transaction.

FORMAT_ID
The OTS transaction format identifier.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

UOW_ID
The identifier of the logical unit of work into which the OTS transaction was imported.

DEFAULT_TIMEOUT
Optional Parameter

The default OTS transaction timeout value.

OTTR gate, COMMIT function

The COMMIT function of the OTTR gate performs the second phase of the sync point of an OTS transaction, confirming that the transaction can be committed. The PREPARE function must be called before the COMMIT function.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

UOW_ROLLEDBACK

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTTR gate, COMMIT_ONE_PHASE function

The COMMIT_ONE_PHASE function of the OTTR gate performs a sync point on the unit of work of the current task, without referencing an external coordinator.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATUS
The outcome of the OTS transaction.

This parameter can have the following values:
COMMITTED
ROLLEDBACK

OTTR gate, IMPORT_TRAN function

The IMPORT_TRAN function of the OTTR gate is used to link the current unit of work of a task to an external transaction. Some information about the external transaction is recorded in the current unit of work.

Input Parameters

BQUAL_LEN
The branch qualifier length of the OTS transaction.

FORMAT_ID
The OTS transaction format identifier.

LOGICAL_SERVER
The name of the logical server in which the transaction is executing.

PUBLIC_ID
The Request Stream public identifier associated with the transaction.

TID_BLOCK_IN
The block parameter containing the unique OTS transaction identifier (TID) of the external transaction.

TIMEOUT
The OTS transaction timeout value.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
OTS_TRAN_ALREADY
TID_TOO_LONG

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

UOW_ID
The identifier of the logical unit of work into which the OTS transaction was imported.

OTTR gate, PREPARE function

The PREPARE function of the OTTR gate performs the first phase of the sync point of an OTS transaction.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

VOTE
The vote from first phase of sync point.

This parameter can have the following values:
YES
NO
READ_ONLY
HEURISTIC_MIXED

OTTR gate, ROLLBACK function

The ROLLBACK function of the OTTR gate rolls back an OTS transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
UOW_COMMITTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

OTTR gate, SET_ROLLBACK_ONLY function

The SET_ROLLBACK_ONLY function of the OTTR gate ensures that the OTS transaction will roll back when it comes to a sync point.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHOTCO	Handles requests on the OTCO gate.
DFHOTDM	Domain initialization and termination. PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHOTDUF	OT domain offline dump formatting routine
DFHOTRM	Handles the following requests: ATTACH
DFHOTSU	Handles requests on the OTSU gate.
DFHOTTR	Handles requests on the OTTR gate.
DFHOTTRI	Interprets OT domain trace entries

Parameter manager domain (PA)

The parameter manager domain informs CICS domains of system parameters during CICS initialization. These system initialization parameters are specified in the system initialization table (SIT), and as temporary override parameters read from the SYSIN data stream or specified interactively at the system console.

Parameter manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the PA domain.

PAGP gate, FORCE_START function

The FORCE_START function of the PAGP gate is used to override the type of start requested by the START system initialization parameter. It is currently used to force START=AUTO if the MVS(TM) automatic restart manager indicates that CICS(R) is being automatically restarted with the original startup JCL (so that CICS does not get a COLD start that the original JCL might have asked for).

Input Parameters

START_TYPE

specifies the type of CICS start to be forced.

Values for the parameter are:

AUTO
COLD

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

NOT_POSSIBLE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PAGP gate, GET_PARAMETERS function

The GET_PARAMETERS function of the PAGP gate is used to get the initialization parameters for a requesting domain.

Input Parameters

FORCE_ALL

specifies whether all parameters are required, even on a non-cold start.

Values for the parameter are:

NO
YES

SKIP_EARLY_BOUND_PARMS

Optional Parameter

Indicates whether early-bound parameters (which cannot be changed beyond a certain stage of initialization) should be skipped.

Values for the parameter are:

NO
YES

Output Parameters

PARAMETERS_TRANSFERRED
indicates to the calling domain whether any system parameters were transferred successfully by the parameter manager domain.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PAGP gate, INQUIRE_PARM function

The INQUIRE_PARM function of the PAGP gate is used to find out about a parameter in the current system initialization table (SIT), or from a specified location.

Input Parameters

LOCATION
Specifies one of the following parameter locations:

LOAD_MODULE
The original version of the SIT with no overrides.

JCL_PARMS
SIT overrides found in the CICS JCL.

SYSIN
SIT overrides found in the SYSIN data set.

CONSOLE
SIT overrides specified from a console.

If a location is not specified, the current SIT is examined. This table might have been modified with override parameters when it was built.

PARM_BUFFER
A 255-byte buffer for the requested parameter values.

Output parameters

LOCATED
Indicates one of the following parameter locations:

LOAD_MODULE
The original version of the SIT with no overrides.

JCL_PARMS
SIT overrides found in the CICS JCL

SYSIN
SIT overrides found in the SYSIN data set.

CONSOLE
SIT overrides specified from a console.

REASON
The following values are returned when the RESPONSE is EXCEPTION:
NOT_FOUND
BUFFER_TOO_SMALL

The following values are returned when the RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_LOCATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PAGP gate, INQUIRE_START function

The INQUIRE_START function of the PAGP gate is used to find out the type of start that CICS is to perform. This information is used to determine whether domains need to perform a cold or warm start.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

START

specifies the type of start CICS is to perform.

Values for the parameter are:

COLD
WARM

ALL

Optional Parameter

Indicates if the ALL option was specified on the START system initialization parameter.

Values for the parameter are:

NO
YES

INITIAL_START

Optional Parameter

Indicates if this is an INITIAL start.

Values for the parameter are:

NO
YES

Parameter manager domain's generic gates

Table 36 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 36. Parameter manager domain's generic gates

Gate	Trace	Functions	Format
DMDM	PA 0201 PA 0202	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

In preinitialization processing, the parameter manager domain reads system initialization (override) parameters from the startup job stream and, if requested, from the SYSIN data set and the console.

If a system initialization table (SIT) has been specified, that is loaded into storage. Otherwise, the default SIT is loaded. The override parameters are applied to the SIT, and related parameters are checked for consistency. Errors are reported, but no action is taken.

The parameter manager domain also provides services to other domains as they pre-initialize. It informs them of the type of start (cold or auto), and supplies information as required from the SIT.

In initialization processing, the parameter manager domain waits for all the other domains to complete their initialization, and then writes a warm start record to the catalog.

The parameter manager domain does no quiesce processing or termination processing.

Modules

Module	Function
DFHPADM	Parameter manager domain initialization and termination
DFHPADUF	An offline routine to format system dump information
DFHPAGP	Passes initialization parameters to domains requesting GET_PARAMETERS
DFHPAIO	Communicates with the SYSIN data set and operator console
DFHPASY	System initialization override parameter checker and syntax parser
DFHPATRI	An offline routine to format trace points

Program manager domain (PG)

The program manager domain provides a variety of functions for managing programs in CICS.

The functions provided by the program manager domain include:

- Program control functions invoked by the following application programming commands:
  LINK
  XCTL
  LOAD
  RELEASE
  RETURN
- Transaction ABEND and condition handling functions invoked by the following commands:
  ABEND
  HANDLE ABEND
  HANDLE CONDITION
  HANDLE AID
- Management of user-replaceable programs, global user exits, and task-related user exits
- Autoinstall for programs, mapsets, and partitionsets.

Program Manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the PG domain.

PGAQ gate, INQUIRE_AUTOINSTALL function

The INQUIRE_AUTOINSTALL function of the PGAQ gate is used to inquire about attributes of the program autoinstall function.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

AUTOINSTALL_CATALOG
Optional Parameter

identifies if program autoinstall events are cataloged.

Values for the parameter are:
ALL
MODIFY
NONE

AUTOINSTALL_EXIT_NAME
Optional Parameter

is the name of the program autoinstall exit program.

AUTOINSTALL_STATE
Optional Parameter

is the state of the program autoinstall function.

Values for the parameter are:
ACTIVE
INACTIVE

PGAQ gate, SET_AUTOINSTALL function

The SET_AUTOINSTALL function of the PGAQ gate is used to set attributes of the program autoinstall function.

Input Parameters

AUTOINSTALL_CATALOG
Optional Parameter

identifies if program autoinstall events are cataloged.

Values for the parameter are:
ALL
MODIFY
NONE

AUTOINSTALL_EXIT_NAME
Optional Parameter

is the name of the program autoinstall exit program.

AUTOINSTALL_STATE
Optional Parameter

is the state of the program autoinstall function.

Values for the parameter are:
ACTIVE
INACTIVE

LANGUAGES_AVAILABLE
Optional Parameter

Indicates if Language Environment is active.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGAQ gate, SET_SYSTEM function

Set system data values owned by the program manager domain.

Input Parameters

DEFAULT_CCSID

Optional Parameter

The coded character set identifer used by the program manager domain.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCH gate, ADD_CHANNEL function

The ADD_CHANNEL function of the PGCH gate is used to add the specified channel to the current program's link level

Input Parameters

CHANNEL_TOKEN

is a token referencing the channel to be added to the current link level.

LINK LEVEL

Optional Parameter.

whether the channel is to be created on the current chain.

Values for the parameter are:

CURRENT.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

CHANNEL_ALREADY_EXISTS

The following values are returned when RESPONSE is INVALID:

INVALID TOKEN

INVALID_LINK_LEVEL

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCH gate, BIND_CHANNEL function

The BIND_CHANNEL function of the PGCH gate is used to make the specified channel the channel used on the initial link.

Input Parameters

CHANNEL_TOKEN
is a token referencing the channel to be used on the initial link.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_LINK_LEVEL

The following values are returned when RESPONSE is INVALID:
CHANNEL_ALREADY_SET
INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCH gate, COPY_CHANNEL function

The COPY_CHANNEL function of the PGCH gate is used to take a copy of a channel and all its containers. The copy has the same name as the original, but is not on any chain. This function is required by the START command.

Input Parameters

CHANNEL_TOKEN
is a token referencing the channel to be used on the initial link.

PERSISTENT
Optional Parameter

whether or not the copied channel is to persist beyond task termination

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_TOKEN

COPIED_CHANNEL_TOKEN
A token referencing a copy of the specified channel (used on START and RETURN commands).

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCH gate, CREATE_CHANNEL function

The CREATE_CHANNEL function of the PGCH gate is used to create a channel.

Input Parameters

CHANNEL_NAME
is the 16-character name of the channel to be created.

CCSID
Optional Parameter

is the default coded character set identifier (CCSID) for character data in this channel.

CURRENT_CHANNEL
Optional Parameter

whether or not the created channel is to be the current channel of the current link level.

Values for the parameter are:
NO
YES

LINK_LEVEL
Optional Parameter

whether the channel is to be created on the current chain, the previous link level's chain, or on no chain (NONE). NONE is used when creating a channel for transfer on a START or RETURN command.

Values for the parameter are:
CURRENT
NONE
PREVIOUS

PERSISTENT
Optional Parameter.

whether or not the created channel is to persist beyond task termination.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
CCSID_INVALID
CHANNEL_ALREADY_EXISTS
CHANNEL_ALREADY_SET
INVALID_CHANNEL_NAME

The following values are returned when RESPONSE is INVALID:
INVALID_LINK_LEVEL
INVALID_PARAMETERS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CHANNEL_TOKEN
Optional Parameter

is a token referencing the newly-created channel.

CONTAINER_POOL_TOKEN
Optional Parameter

is a token to access a pool of containers.

PGCH gate, DELETE_CHANNEL function

The DELETE_CHANNEL function of the PGCH gate is used to delete a channel. This command can be used to delete channels when they are bound to principal facilities, but not to PLCBs.

Input Parameters

CHANNEL_TOKEN
is a token referencing the channel to be used on the initial link.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_ATTACHED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCH gate, DELETE_OWNED_CHANNELS function

The DELETE_OWNED_CHANNELS function of the PGCH gate is used to delete all channels from the channel chain. If the current channel is owned by this link level, it is deleted as well. The container pool associated with each channel is also deleted. This ends any browse in progress and deletes all containers.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

```
INVALID_LINK_LEVEL
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCH gate, DETACH_CHANNEL function

The DETACH_CHANNEL function of the PGCH gate is used to detach a channel. The channel may be the current channel, or on the PLCB chain. The channel's containers are only deleted if DELETE(YES) is specified. It is implied that a SET_CURRENT_CHANNEL will be done with this channel at some time.

Input Parameters

CHANNEL_TOKEN

is a token referencing the channel to be used on the initial link.

DELETE

Optional Parameter

whether the channel's containers should be deleted.

Values for the parameter are:

```
NO
YES
```

FREE_SET_STORAGE

Optional Parameter

whether the channel's storage should be freed.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_NOT_FOUND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_LINK_LEVEL
```

The following values are returned when RESPONSE is INVALID:

INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCH gate, FREE_ALL_TASK_STORAGE function

The FREE_ALL_TASK_STORAGE function of the PGCH gate is used to free all task lifetime storage that is related to a channel.

Input Parameters

CHANNEL_TOKEN
Is a token identifying the channel.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCH gate, INQUIRE_BOUND_CHANNEL function

The INQUIRE_BOUND_CHANNEL function of the PGCH gate is used to get information about the channel that is bound to the current transaction. This may or may not be the current channel. This request may be issued outside a program manager environment.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CHANNEL_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CHANNEL_NAME
Optional Parameter

is the name of the bound channel.

CHANNEL_TOKEN
Optional Parameter

is a token referencing the newly-created channel.

CONTAINER_POOL_TOKEN
Optional Parameter

is a token to access a pool of containers.

PGCH gate, INQUIRE_CHANNEL function

The INQUIRE_CHANNEL function of the PGCH gate is used to retrieve the properties of a named channel, including its address (returned as a token). To find the named channel, CICS scans the channels accessible from the specified link level.

Input Parameters

CHANNEL_NAME
is the 16-character name of the channel to be created.

LINK_LEVEL
Optional Parameter

whether the channel is to be created on the current chain, the previous link level's chain, or on no chain (NONE). NONE is used when creating a channel for transfer on a START or RETURN command.

Values for the parameter are:
CURRENT
PREVIOUS

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
CHANNEL_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_LINK_LEVEL

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCSID

Optional Parameter

is the default coded character set identifier (CCSID) for character data in the named channel.

CHANNEL_TOKEN

Optional Parameter

is a token referencing the newly-created channel.

CONTAINER_POOL_TOKEN

Optional Parameter

is a token to access a pool of containers.

CURRENT_CHANNEL

Optional Parameter

whether the named channel is the current channel.

Values for the parameter are:
NO
YES

OWNER

Optional Parameter

whether the named channel is owned by the specified link level.

Values for the parameter are:
NO
YES

PGCH gate, INQUIRE_CHANNEL_BY_TOKEN function

The INQUIRE_CHANNEL_BY_TOKEN function is used to retrieve the properties of a channel (which is specified by token).

Input Parameters

CHANNEL_TOKEN

is a token referencing the channel to be used on the initial link.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:
INVALID_LINK_LEVEL

The following values are returned when RESPONSE is INVALID:
INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCSID
Optional Parameter

is the default coded character set identifier (CCSID) for character data in the named channel.

CHANNEL_NAME
Optional Parameter

is the name of the bound channel.

CONTAINER_POOL_TOKEN
Optional Parameter

is a token to access a pool of containers.

CURRENT_CHANNEL
Optional Parameter

whether the named channel is the current channel.

Values for the parameter are:
NO
YES

OWNER
Optional Parameter

whether the named channel is owned by the specified link level.

Values for the parameter are:
NO
YES

PGCH gate, INQUIRE_CURRENT_CHANNEL function

The INQUIRE_CURRENT_CHANNEL function of the PGCH gate is used to retrieve the properties of the current channel.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CHANNEL_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_LINK_LEVEL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCSID
Optional Parameter

is the default coded character set identifier (CCSID) for character data in the named channel.

CHANNEL_NAME
Optional Parameter

is the name of the bound channel.

CHANNEL_TOKEN
Optional Parameter

is a token referencing the newly-created channel.

CONTAINER_POOL_TOKEN
Optional Parameter

is a token to access a pool of containers.

OWNER
Optional Parameter

whether the named channel is owned by the specified link level.

Values for the parameter are:
NO
YES

PGCH gate, RENAME_CHANNEL function

The RENAME_CHANNEL function of the PGCH gate is used to rename a channel.

Input Parameters

CHANNEL_NAME
is the 16-character name of the channel to be created.

CHANNEL_TOKEN
is a token referencing the channel to be used on the initial link.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
CHANNEL_ALREADY_EXISTS
INVALID_CHANNEL_NAME

The following values are returned when RESPONSE is INVALID:
INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCH gate, SET_CURRENT_CHANNEL function

The SET_CURRENT_CHANNEL function of the PGCH gate is used to make the specified channel the current channel for the current link level.

Input Parameters

CHANNEL_TOKEN
is a token referencing the channel to be used on the initial link.

OWNER
Optional Parameter

whether the specified channel is owned by the current link level. If OWNER(YES) is specified, the channel is added to the current link level's chain.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CHANNEL_ALREADY_EXISTS

The following values are returned when RESPONSE is INVALID:
INVALID_LINK_LEVEL

The following values are returned when RESPONSE is INVALID:
INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCP gate, COPY_CONTAINER_POOL function

The COPY_CONTAINER_POOL function of the PGCP gate is used to copy all the containers in a container pool to another container pool.

Input Parameters

POOL_TOKEN
is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_POOL_TOKEN

COPIED_POOL_TOKEN
is a token that maps to the pool to which all containers have been copied from the pool referenced by POOL_TOKEN.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCP gate, CREATE_CONTAINER_POOL function

The CREATE_CONTAINER_POOL function of the PGCP gate is used to create a container pool.

Input Parameters

CCSID
Optional Parameter

is the default coded character set identifier (CCSID) for character data in this channel.

Output Parameters

POOL_TOKEN
is a token that references the container pool that has been created.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCP gate, DELETE_CONTAINER_POOL function

The DELETE_CONTAINER_POOL function of the PGCP gate is used to delete a container pool.

Input Parameters

POOL_TOKEN
is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_POOL_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCP gate, INQUIRE_CONTAINER_POOL function

The INQUIRE_CONTAINER_POOL function of the PGCP gate is used to inquire about the attributes of a container pool.

Input Parameters

POOL_TOKEN
is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`INVALID_POOL_TOKEN`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCSID
Optional Parameter

is the default coded character set identifier (CCSID) for character data in the named channel.

NUMBER_OF_CONTAINERS
Optional Parameter

is the number of containers that the pool contains.

POOL_SIZE
Optional Parameter

is the size, in bytes, of the data in the pool.

PGCR gate, COPY_CONTAINER function

The COPY_CONTAINER function of the PGCR gate is used to copy a container from one container pool to another. Both pools must already have been created.

Input Parameters

AS_CONTAINER_NAME
Optional Parameter

is the name by which the copied container is to be known in the target container pool.

CONTAINER_NAME
Optional Parameter

is the name of the container to be copied.

CONTAINER_TOKEN
Optional Parameter

is a token referencing the container to be copied.

POOL_TOKEN
Optional Parameter

is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

TO_POOL_TOKEN
Optional Parameter

is a token referencing the target container pool (that is, the pool to which the container is to be copied).

TYPE

Optional Parameter

whether the container is visible only to CICS, or to user programs as well.

Values for the parameter are:

CICS
USER

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

CONTAINER_NOT_FOUND
INVALID_AS_CONTAINER_NAME

The following values are returned when RESPONSE is INVALID:

INVALID_CONTAINER_TOKEN
INVALID_PARAMETERS
INVALID_POOL_TOKEN
INVALID_TO_POOL_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONTAINER_TOKEN_OUT

Optional Parameter

is a token representing the new copy of the container.

GENERATION_NUMBER

Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

INITIAL_GENERATION

Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

PGCR gate, DELETE_CONTAINER function

The DELETE_CONTAINER function of the PGCR gate is used to delete a container and its data. The container is identified using its name, the container pool to which it belongs, and its type.

Input Parameters

CALLER

Optional Parameter

is the call part of an API call.

Values for the parameter are:

EXEC
SYSTEM

CONTAINER_NAME

Optional Parameter

is the name of the container to be copied.

CONTAINER_TOKEN
Optional Parameter

is a token referencing the container to be copied.

POOL_TOKEN
Optional Parameter

is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

TYPE
Optional Parameter

whether the container is visible only to CICS, or to user programs as well.

Values for the parameter are:
```
CICS
USER
```

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
CONTAINER_NOT_FOUND
READONLY_CONTAINER
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_CONTAINER_TOKEN
INVALID_PARAMETERS
INVALID_POOL_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCR gate, ENDBR_CONTAINER function

The ENDBR_CONTAINER function of the PGCR gate is used to end a browse of containers.

Input Parameters

BROWSE_TOKEN
is a browse token referencing the next container in the container pool being browsed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
INVALID_BROWSE_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCR gate, GET_CONTAINER_INTO function

The GET_CONTAINER function of the PGCR gate is used to get the data from a container into an area provided by the caller. The container is identified using a pool token, together with the name and type of the container. Note that LENGTH_ERROR indicates that as much data as possible has been copied.

Input Parameters

CALLER
Optional Parameter

The call part of an API call.

Values for the parameter are as follows:
EXEC
SYSTEM

CCSID
Optional Parameter

The default coded character set identifier (CCSID) for character data in this channel.

CONTAINER_NAME
Optional Parameter

The name of the container to be copied.

CONTAINER_TOKEN
Optional Parameter

A token that references the container to be copied.

CONVERT
Optional Parameter

Whether the data in the container is converted.

Values for the parameter are as follows:
NO
YES

DATA_TOKEN_IN
Optional Parameter

A token that references the data in the container. The value returned in DATA_TOKEN_OUT on one GET_CONTAINER_INTO call must be specified on the next call as DATA_TOKEN_IN. The first GET_CONTAINER_INTO call for this container does not have a DATA_TOKEN_IN.

ITEM_BUFFER
Optional Parameter

On input, ITEM_BUFFER_P is a pointer to a receiving area of length ITEM_BUFFER_M. On output, the value ITEM_BUFFER_N is set to the actual length returned.

ITEM_BUFFER64
Optional Parameter

On input, ITEM_BUFFER64_P is a 64-bit pointer to a receiving area of length ITEM_BUFFER_M. On output, the value ITEM_BUFFER64_N is set to the actual length returned.

POOL_TOKEN
Optional Parameter

A token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

TYPE
Optional Parameter

Whether the container is visible only to CICS, or also to user programs.

Values for the parameter are as follows:
CICS
USER

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

CCSID_CONVERSION_ERROR
CCSID_IGNORED
CCSID_INVALID
CCSID_PAIR_UNSUPPORTED
CCSID_PARTIAL_CONVERSION
CONTAINER_NOT_FOUND
INVALID_DATA_TOKEN_IN
LENGTH_ERROR
MORE_DATA

The following values are returned when RESPONSE is INVALID:
INVALID_CONTAINER_TOKEN
INVALID_PARAMETERS
INVALID_POOL_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONTAINER_CCSID
Optional Parameter

The coded character set identifier of the extracted data.

DATA_TOKEN_OUT
Optional Parameter

A token that references the data in the container.
The value returned in DATA_TOKEN_OUT on one GET_CONTAINER_INTO call must be specified on the next call as DATA_TOKEN_IN. The first GET_CONTAINER_INTO call for this container does not have a DATA_TOKEN_IN.

DATATYPE
Optional Parameter

The format of the data.

Values for the parameter are as follows:
BIT
CHAR

GENERATION_NUMBER
Optional Parameter

Every time a container in a container pool is changed or created, the pool generation number is incremented. This number is the number for the container when the container was last changed.

INITIAL_GENERATION
Optional Parameter

Every time a container in a container pool is changed or created, the pool generation number is incremented. This number is the number for the container when the container was created.

USERACCESS
Optional Parameter

Whether USER containers can be updated by API commands.

Values for the parameter are as follows:
ANY
READONLY

PGCR gate, GET_CONTAINER_LENGTH function

The GET_CONTAINER_LENGTH function of the PGCR gate is used to discover the length, in bytes, of the data in a container.

Input Parameters

CALLER

Optional Parameter

is the call part of an API call.

Values for the parameter are:

EXEC
SYSTEM

CCSID

Optional Parameter

is the default coded character set identifier (CCSID) for character data in this channel.

CONTAINER_NAME

Optional Parameter

is the name of the container to be copied.

CONTAINER_TOKEN

Optional Parameter

is a token referencing the container to be copied.

POOL_TOKEN

Optional Parameter

is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

TYPE

Optional Parameter

whether the container is visible only to CICS, or to user programs as well.

Values for the parameter are:

CICS
USER

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

CCSID_CONVERSION_ERROR
CCSID_IGNORED
CCSID_INVALID
CCSID_PAIR_UNSUPPORTED
CCSID_PARTIAL_CONVERSION
CONTAINER_NOT_FOUND

The following values are returned when RESPONSE is INVALID:

INVALID_CONTAINER_TOKEN
INVALID_PARAMETERS
INVALID_POOL_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONTAINER_CCSID

Optional Parameter

is the coded character set identifier of the extracted data.

DATA_LENGTH

Optional Parameter

is the length, in bytes, of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the converted data.

DATATYPE
Optional Parameter

is the format of the data.

Values for the parameter are:
BIT
CHAR

GENERATION_NUMBER
Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

INITIAL_GENERATION
Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

USERACCESS
Optional Parameter

whether USER containers can be updated by API commands.

Values for the parameter are:
ANY
READONLY

PGCR gate, GET_CONTAINER_SET function

The GET_CONTAINER_SET function of the PGCR gate is used to get the data from a container and copy it into an area provided by the CICS program domain. The container is identified using a pool token, together with the name and type of the container.

Input Parameters

CALLER
Optional Parameter

The call part of an API call.

Values for the parameter are as follows:
EXEC
SYSTEM

CCSID
Optional Parameter

The default coded character set identifier (CCSID) for character data in this channel.

CONTAINER_NAME
Optional Parameter

The name of the container to be copied.

CONTAINER_TOKEN
Optional Parameter

A token that references the container to be copied.

CONVERT
Optional Parameter

Whether the data in the container is converted.

Values for the parameter are as follows:

NO
YES

POOL_TOKEN

Optional Parameter

A token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

TYPE

Optional Parameter

Whether the container is visible only to CICS, or also to user programs.

Values for the parameter are as follows:

CICS
USER

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

CCSID_CONVERSION_ERROR
CCSID_IGNORED
CCSID_INVALID
CCSID_PAIR_UNSUPPORTED
CCSID_PARTIAL_CONVERSION
CONTAINER_NOT_FOUND

The following values are returned when RESPONSE is INVALID:

INVALID_CONTAINER_TOKEN
INVALID_PARAMETERS
INVALID_POOL_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONTAINER_CCSID

Optional Parameter

The coded character set identifier of the extracted data.

DATATYPE

Optional Parameter

The format of the data.

Values for the parameter are as follows:

BIT
CHAR

GENERATION_NUMBER

Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

INITIAL_GENERATION

Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

ITEM_DATA
Optional Parameter

The address and length of the SET storage returned.

ITEM_DATA64
Optional Parameter

The 64-bit address and length of the SET storage returned.

USERACCESS
Optional Parameter

whether USER containers can be updated by API commands.

Values for the parameter are as follows:
ANY
READONLY

PGCR gate, GETNEXT_CONTAINER function

The GETNEXT_CONTAINER function of the PGCR gate is used to get the next container in a browse of containers.

Input Parameters

BROWSE_TOKEN
is a browse token referencing the next container in the container pool being browsed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCSID
Optional Parameter

is the default coded character set identifier (CCSID) for character data in the named channel.

CONTAINER_NAME
Optional Parameter

is the name of the container.

CONTAINER_TOKEN
Optional Parameter

is a token referencing the container.

DATA_LENGTH
Optional Parameter

is the length, in bytes, of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the converted data.

DATATYPE
Optional Parameter

is the format of the data.

Values for the parameter are:
BIT
CHAR

GENERATION_NUMBER
Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

INITIAL_GENERATION
Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

TYPE
Optional Parameter

whether the container is visible only to CICS, or to user programs as well.

Values for the parameter are:
CICS
USER

USERACCESS
Optional Parameter

whether USER containers can be updated by API commands.

Values for the parameter are:
ANY
READONLY

PGCR gate, INQUIRE_BROWSE_CONTEXT function

The INQUIRE_BROWSE__CONTEXT function of the PGCR gate is used to check the context of the browse token.

Input Parameters

BROWSE_TOKEN
is a browse token referencing the next container in the container pool being browsed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCR gate, INQUIRE_CONTAINER function

The INQUIRE_CONTAINER function of the PGCR gate is used to retrieve the attributes of a container.

Input Parameters

CONTAINER_NAME
is the name of the container to be copied.

POOL_TOKEN
is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

CALLER
Optional Parameter

is the call part of an API call.

Values for the parameter are:
EXEC
SYSTEM

TYPE
Optional Parameter

whether the container is visible only to CICS, or to user programs as well.

Values for the parameter are:
CICS
USER

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CONTAINER_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_CONTAINER_TOKEN
INVALID_PARAMETERS
INVALID_POOL_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCSID
Optional Parameter

is the default coded character set identifier (CCSID) for character data in the named channel.

CONTAINER_TOKEN
Optional Parameter

is a token referencing the container.

DATA_LENGTH
Optional Parameter

is the length, in bytes, of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the converted data.

DATATYPE
Optional Parameter

is the format of the data.

Values for the parameter are:
BIT
CHAR

GENERATION_NUMBER
Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

INITIAL_GENERATION
Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

USERACCESS
Optional Parameter

whether USER containers can be updated by API commands.

Values for the parameter are:
ANY
READONLY

PGCR gate, INQUIRE_CONTAINER_BY_TOKEN function

The INQIRE_CONTAINER_BY_TOKEN function of the PGCR gate is used to retrieve the attributes of a container by means of a token.

Input Parameters

CONTAINER_TOKEN
is a token referencing the container to be copied.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CONTAINER_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_CONTAINER_TOKEN
INVALID_PARAMETERS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CCSID
Optional Parameter

is the default coded character set identifier (CCSID) for character data in the named channel.

CONTAINER_NAME
Optional Parameter

is the name of the container.

DATA_LENGTH
Optional Parameter

is the length, in bytes, of the data in the container. If the container holds character data that has been converted from one CCSID to another, this is the length of the converted data.

DATATYPE
Optional Parameter

is the format of the data.

Values for the parameter are:
BIT
CHAR

GENERATION_NUMBER
Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

INITIAL_GENERATION
Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

TYPE
Optional Parameter

whether the container is visible only to CICS, or to user programs as well.

Values for the parameter are:
CICS
USER

USERACCESS
Optional Parameter

whether USER containers can be updated by API commands.

Values for the parameter are:
ANY
READONLY

PGCR gate, MOVE_CONTAINER function

The MOVE_CONTAINER function of the PGCR gate is used to move a container from one container pool to another. Both pools must already have been created. If the TO_POOL_TOKEN is not specified, the container is not moved to a different pool but is renamed to the value of AS_CONTAINER_NAME.

Input Parameters

AS_CONTAINER_NAME
Optional Parameter

is the name by which the copied container is to be known in the target container pool.

CALLER
Optional Parameter

is the call part of an API call.

Values for the parameter are:
EXEC
SYSTEM

CONTAINER_NAME
Optional Parameter

is the name of the container to be copied.

CONTAINER_TOKEN
Optional Parameter

is a token referencing the container to be copied.

POOL_TOKEN
Optional Parameter

is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

TO_POOL_TOKEN
Optional Parameter

is a token referencing the target container pool (that is, the pool to which the container is to be copied).

TYPE
Optional Parameter

whether the container is visible only to CICS, or to user programs as well.

Values for the parameter are:
CICS
USER

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
CONTAINER_NOT_FOUND
INVALID_AS_CONTAINER_NAME
READONLY_AS_CONTAINER
READONLY_CONTAINER
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_CONTAINER_TOKEN
INVALID_PARAMETERS
INVALID_POOL_TOKEN
INVALID_TO_POOL_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONTAINER_TOKEN_OUT

Optional Parameter

is a token representing the new copy of the container.

GENERATION_NUMBER

Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was last changed.

INITIAL_GENERATION

Optional Parameter

Every time a container in a container pool is changed or created the pool generation number is incremented. This number is the number for the container when the container was created.

PGCR gate, PUT_CONTAINER function

The PUT_CONTAINER function of the PGCR gate is used to put data into a container from an area provided by the caller.

Input Parameters

CALLER

Optional Parameter

The call part of an API call.

Values for the parameter are as follows:

```
EXEC
SYSTEM
```

CCSID

Optional Parameter

The default coded character set identifier (CCSID) for character data in this channel.

CONTAINER_NAME

Optional Parameter

The name of the container to be copied.

CONTAINER_TOKEN

Optional Parameter

A token that references the container to be copied.

CONVERT

Optional Parameter

Whether to convert the data in the container.

Values for the parameter are as follows:
NO
YES (default)

DATATYPE
Optional Parameter

The format of the data.

Values for the parameter are as follows:
BIT
CHAR

ITEM_DATA
Optional Parameter

The address and length of the put data.

ITEM_DATA64
Optional Parameter

The 64-bit address and length of the put data.

POOL_TOKEN
Optional Parameter

A token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

PUT_TYPE
Optional Parameter

Whether the PUT data is appended to the current contents of the container, or replaces the current contents.

Values for the parameter are as follows:
APPEND
REPLACE

TYPE
Optional Parameter

Whether the container is visible only to CICS, or also to user programs.

Values for the parameter are as follows:
CICS
USER

USERACCESS
Optional Parameter

Whether USER containers can be updated by API commands.

Values for the parameter are as follows:
ANY
READONLY

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CCSID_INVALID
DATATYPE_CHANGE
INVALID_CONTAINER_NAME
LENGTH_ERROR
READONLY_CONTAINER

The following values are returned when RESPONSE is INVALID:

INVALID_CONTAINER_TOKEN
INVALID_PARAMETERS
INVALID_POOL_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONTAINER_TOKEN_OUT
Optional Parameter

A token that represents the new copy of the container.

GENERATION_NUMBER
Optional Parameter

Every time a container in a container pool is changed or created, the pool generation number is incremented. This number is the number for the container when the container was last changed.

INITIAL_GENERATION
Optional Parameter

Every time a container in a container pool is changed or created, the pool generation number is incremented. This number is the number for the container when the container was created.

PGCR gate, SET_CONTAINER function

The SET_CONTAINER function of the PGCR gate is used to change the attributes of a container.

Input Parameters

CONTAINER_NAME
Optional Parameter

is the name of the container to be copied.

CONTAINER_TOKEN
Optional Parameter

is a token referencing the container to be copied.

POOL_TOKEN
Optional Parameter

is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

TYPE
Optional Parameter

whether the container is visible only to CICS, or to user programs as well.

Values for the parameter are:
CICS
USER

USERACCESS
Optional Parameter

whether USER containers can be updated by API commands.

Values for the parameter are:
ANY
READONLY

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

CONTAINER_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_CONTAINER_TOKEN
INVALID_PARAMETERS
INVALID_POOL_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCR gate, STARTBR_CONTAINER function

The STARTBR_CONTAINER function of the PGCR gate is used to initiate a browse of the containers in a specified container pool.

Input Parameters

POOL_TOKEN
is a token (returned on a CREATE_CONTAINER_POOL request) that identifies the container pool to be copied.

CALLER
Optional Parameter

is the call part of an API call.

Values for the parameter are:
EXEC
SYSTEM

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_POOL_TOKEN

BROWSE_TOKEN
is a browse token referencing a container in the container pool. This container is the first in the browse list.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGCR gate, TRACE_CONTAINERS function

The TRACE_CONTAINER function of the PGCR gate is used to initiate a trace of the containers in a specified channel.

Input Parameters

CHANNEL_TOKEN
is a token referencing the channel to be used on the initial link.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_CHANNEL_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGDD gate, ADOPT_PROGRAM function

The ADOPT_PROGRAM function of the PGDD gate is used to adopt a public program as an application entry point.

Input Parameters

APPLICATION_ID
The 64-character name of the application associated with the program.

APPL_MAJOR_VER
The fullword binary value that is the major version of the application associated with the program.

APPL_MINOR_VER
The fullword binary value that is the minor version of the application associated with the program.

APPL_MICRO_VER
The fullword binary value that is the micro version of the application associated with the program.

PLATFORM_ID
The 64-character name of the platform associated with the program.

PROGRAM_NAME
The name of the public program resource to be adopted.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
INSUFFICIENT_STORAGE
LOCK_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
PROGRAM_NOT_DEFINED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGDD gate, DEFINE_PROGRAM function

The DEFINE_PROGRAM function of the PGDD gate is used to define a program resource.

Input Parameters

APPLICATION_ID
Optional parameter

The 64-character name of the application associated with the task.

APPL_MAJOR_VER
Optional parameter

The fullword binary value that is the major version of the application associated with the task.

APPL_MINOR_VER
Optional parameter

The fullword binary value that is the minor version of the application associated with the task.

APPL_MICRO_VER
Optional parameter

The fullword binary value that is the micro version of the application associated with the task.

CATALOG_ADDRESS
The token that identifies the program resource to be defined.

INSTALL_TYPE
Indicates how the program resource is defined and installed.

Values for the parameter are as follows:

AUTO
CATALOG
GROUPLIST
MANUAL
RDO
SYSAUTO

PLATFORM_ID

Optional parameter

The 64-character name of the platform associated with the task.

PROGRAM_NAME

The name of the program resource to be defined.

AVAIL_STATUS

Optional Parameter

Defines whether the program can be used (ENABLED) or not (DISABLED).

Values for the parameter are as follows:

DISABLED
ENABLED

CEDF_STATUS

Optional Parameter

Indicates whether the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF).

Values for the parameter are as follows:

CEDF
NOCEDF

CONCURRENCY

Optional Parameter

Indicates whether the program is threadsafe or only quasi-reentrant.

Values for the parameter are as follows:

QUASIRENT
THREADSAFE

DATA_LOCATION

Optional Parameter

Defines whether the program can handle only 24-bit addresses (data located below the 16 MB line) or can also handle 31-bit addresses (data located above or below the 16 MB line). The DATALOCATION options are independent from the addressing mode of the link-edited program.

Values for the parameter are as follows:

ANY
BELOW

DYNAMIC_STATUS

Optional Parameter

Indicates whether a request to LINK to the program can be dynamically routed.

Values for the parameter are as follows:

DYNAMIC
NOTDYNAMIC

EXECUTION_KEY

Optional Parameter

The key in which CICS gives control to the program, and that determines whether the program can modify CICS-key storage. If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into the extended read-only DSA (ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

Values for the parameter are as follows:

```
CICS
USER
```

EXECUTION_SET

Optional Parameter

Indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program).

Values for the parameter are as follows:

```
DPLSUBSET
FULLAPI
```

HOTPOOL

Optional Parameter

Indicates whether the Java program object is to be run in a preinitialized Language Environment enclave reused by multiple invocations of the program, under control of an H8 TCB. This parameter is obsolete and is ignored.

Values for the parameter are as follows:

```
NO
YES
```

JVM

Optional Parameter

Indicates whether the program is to be executed under the control of a JVM (Java Virtual Machine).

Values for the parameter are as follows:

```
NO
YES
```

JVM_CLASS

Optional Parameter

The name of the main class in a Java program to be run under the control of a JVM.

JVM_PROFILE

Optional Parameter

Specifies the name of the data set member that contains the JVM profile. The named profile provides the attributes of the JVM that are needed to run the program.

LANGUAGE_DEFINED

Optional Parameter

The language to be defined for the program.

Values for the parameter are as follows:

```
ASSEMBLER
COBOL
C370
LE370
```

NOT_DEFINED
PLI

MODULE_TYPE
Optional Parameter

The type of program resource to be defined.

Values for the parameter are as follows:
MAPSET
PARTITIONSET
PROGRAM

MULTITCB
Optional Parameter

Reserved for future use.

Values for the parameter are as follows:
NO
YES

OPENAPI
Optional Parameter

Reserved for future use.

Values for the parameter are as follows:
NO
YES

PROGRAM_ATTRIBUTE
Optional Parameter

Defines the residence status of the program, and when the storage for this program is released.

Values for the parameter are as follows:
RELOAD
RESIDENT
REUSABLE
TEST
TRANSIENT

PROGRAM_TYPE
Optional Parameter

The type of program.

Values for the parameter are as follows:
PRIVATE
SHARED
TYPE_ANY

PROGRAM_USAGE
Optional Parameter

Defines whether the program is to be used as a CICS nucleus program, or as a user application program.

Values for the parameter are as follows:
APPLICATION
NUCLEUS

REMOTE_PROGID
Optional Parameter

The name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value).

REMOTE_SYSID

Optional Parameter

The name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

REMOTE_TRANID

Optional Parameter

The name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

REQUIRED_AMODE

Optional Parameter

The addressing mode of the program.

Values for the parameter are as follows:

```
AMODE_ANY
24
31
64
```

REQUIRED_RMODE

Optional Parameter

The residence mode of the program.

Values for the parameter are as follows:

```
RMODE_ANY
24
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
CATALOG_ERROR
CATALOG_NOT_OPERATIONAL
INSUFFICIENT_STORAGE
LOCK_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
PROGRAM_ALREADY_DEFINED
PROGRAM_HAS_HOTPOOL
PROGRAM_IN_USE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_CATALOG_ADDRESS
INVALID_MODE_COMBINATION
INVALID_PROGRAM_NAME
INVALID_TYPE_ATTRIB_COMBIN
```

The values for the parameter are as follows:

```
NO_REASON
```

NEW_PROGRAM_TOKEN

The token assigned to program.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGDD gate, DELETE_PROGRAM function

The DELETE_PROGRAM function of the PGDD gate is used to delete a program resource.

Input Parameters

APPLICATION_ID

Optional parameter

The 64-character name of the application associated with the task.

APPL_MAJOR_VER

Optional parameter

The fullword binary value that is the major version of the application associated with the task.

APPL_MINOR_VER

Optional parameter

The fullword binary value that is the minor version of the application associated with the task.

APPL_MICRO_VER

Optional parameter

The fullword binary value that is the micro version of the application associated with the task.

PLATFORM_ID

Optional parameter

The 64-character name of the platform associated with the task.

PROGRAM_NAME

The name of the program resource to be deleted.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
PROGRAM_IN_USE
PROGRAM_IS_URM
PROGRAM_NAME_STARTS_DFH
PROGRAM_NOT_DEFINED
```

The values for the parameter are as follows:

```
NO_REASON
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGEX gate, INITIALIZE_EXIT function

The INITIALIZE_EXIT function of the PGEX gate is used to initialize an exit program.

Input Parameters

LOAD_PROGRAM

defines whether or not the program is to be loaded when initialized.

Values for the parameter are:

```
NO
YES
```

PROGRAM_NAME
is the name of the program resource to be defined.

SYSTEM_AUTOINSTALL
defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition.

Values for the parameter are:
NO
YES

LPA_ELIGIBLE
Optional Parameter

defines whether or not the program can be loaded into the link pack area (LPA).

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
ABEND
AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
INVALID_FUNCTION
INVALID_INITIALIZE_REQUEST
JVM_PROGRAM
LOOP
PROGRAM_NOT_AUTHORIZED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
REMOTE_PROGRAM

PROGRAM_TOKEN
is the token assigned to program.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ENTRY_POINT
Optional Parameter

is the token defining the entry point of the program.

PGEX gate, TERMINATE_EXIT function

The TERMINATE_EXIT function of the PGEX gate is used to terminate an exit program.

Input Parameters

PROGRAM_TOKEN
is the token identifying the program to be terminated.

RELEASE_PROGRAM
defines whether or not the program is to be released when terminated.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The values for the parameter are:

```
ABEND
INVALID_FUNCTION
INVALID_PROGRAM_TOKEN
LOOP
PROGRAM_NOT_AUTHORIZED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_IN_USE
PROGRAM_NOT_LOADED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGHM gate, CLEAR_LABELS function

The CLEAR_LABELS function of the PGHM gate is invoked by CICS during XCTL processing and frees all storage relating to the Handle State for that program (except for the initial default state) and removes all user-defined label handles.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
MISSING_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FASTPATH_FLAGS

Optional Parameter

identifies the fastpath flag settings for the following conditions handled by the user: RDATT, WRBRK, EOF, NOSPACE, QBUSY, NOSTG, ENQBUSY, NOJBUFSP, SIGNAL, OVERFLOW, SYSBUSY, SESSBUSY.

PGHM gate, FREE_HANDLE_TABLES function

The FREE_HANDLE_TABLES function of the PGHM gate is invoked by CICS during program termination processing and frees all storage relating to the Handle State for that program level.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
MISSING_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGHM gate, IGNORE_CONDITIONS function

The IGNORE_CONDITIONS function of the PGHM gate is used to ignore the conditions for user EXEC CICS IGNORE CONDITION commands.

Input Parameters

IDENTIFIERS

is the token identifying the conditions to be handled.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
MISSING_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FASTPATH_FLAGS

Optional Parameter

identifies the fastpath flag settings for the following conditions handled by the user: RDATT, WRBRK, EOF, NOSPACE, QBUSY, NOSTG, ENQBUSY, NOJBUFSP, SIGNAL, OVERFLOW, SYSBUSY, SESSBUSY.

PGHM gate, INQ_ABEND function

The INQ_ABEND function of the PGHM gate is invoked when an abend has occurred, and returns to the caller details of the handle abend for user EXEC CICS HANDLE ABEND commands.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
MISSING_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATUS

Identifies the status of the condition.

Values for the parameter are:

```
HANDLED
SYSTEM_DEFAULT
```

CURRENT_EXECUTION_KEY

Optional Parameter

An 8-bit value that indicates the current program execution key (at the time the EXEC CICS HANDLE CONDITION command was issued).

GOTOL

Optional Parameter

The token that identifies the condition label in the program to be branched to if the condition is ignored.

HANDLE_COUNT
Optional Parameter

The number of times that this abend code has been handled.

HANDLE_TYPE
Optional Parameter

Indicates whether control should be passed to a label or a program when the abend occurs.

Values for the parameter are:
- LBL
- PGM

LABEL
Optional Parameter

The token that identifies the condition label in the program to be branched to if the condition occurs.

LANGUAGE
Optional Parameter

The program language.

Values for the parameter are:
- ASSEMBLER
- COBOL
- CPP
- C370
- PLI

PROGRAM
Optional Parameter

The name of the program to which control was passed when the abend occurred.

PROGRAM_MASK
Optional Parameter

Identifies the program mask at the time the HANDLE CONDITION command was executed.

USERS_RSA_POINTER
Optional Parameter

The address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

PGHM gate, INQ_AID function

The INQ_AID function of the PGHM gate is invoked when an aid has occurred, and returns to the caller details of the handle aid for user EXEC CICS HANDLE AID commands.

Input Parameters

AID
is an 8-bit value identifying the aid.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
- ABEND
- LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION
MISSING_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATUS
identifies the status of the condition.

Values for the parameter are:
HANDLED
SYSTEM_DEFAULT

CURRENT_EXECUTION_KEY
Optional Parameter

is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE CONDITION command was issued).

GOTOL
Optional Parameter

is the token identifying the condition label within the program to be branched to if the condition is ignored.

LABEL
Optional Parameter

is the token identifying the condition label within the program to be branched to if the condition occurs.

LANGUAGE
Optional Parameter

is the program language.

Values for the parameter are:
ASSEMBLER
COBOL
CPP
C370
PLI

PROGRAM_MASK
Optional Parameter

identifies the program mask at the time the HANDLE CONDITION command was executed.

USERS_RSA_POINTER
Optional Parameter

is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

PGHM gate, INQ_CONDITION function

The INQ_CONDITION function of the PGHM gate is invoked when a condition has occurred, and returns to the caller about details of the condition for user EXEC CICS HANDLE CONDITION commands.

Input Parameters

CONDITION
is an 8-bit value identifying the condition.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
MISSING_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATUS

identifies the status of the condition.

Values for the parameter are:

```
HANDLED
IGNORED
SYSTEM_DEFAULT
```

ABEND_CODE

Optional Parameter

is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

CURRENT_EXECUTION_KEY

Optional Parameter

is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE CONDITION command was issued).

GOTOL

Optional Parameter

is the token identifying the condition label within the program to be branched to if the condition is ignored.

LABEL

Optional Parameter

is the token identifying the condition label within the program to be branched to if the condition occurs.

LANGUAGE

Optional Parameter

is the program language.

Values for the parameter are:

```
ASSEMBLER
COBOL
CPP
C370
PLI
```

PROGRAM_MASK

Optional Parameter

identifies the program mask at the time the HANDLE CONDITION command was executed.

USERS_RSA_POINTER

Optional Parameter

is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

PGHM gate, POP_HANDLE function

The POP_HANDLE function of the PGHM gate is invoked for a user EXEC CICS POP command.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
NO_PREVIOUS_PUSH
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
MISSING_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FASTPATH_FLAGS

Optional Parameter

identifies the fastpath flag settings for the following conditions handled by the user: RDATT, WRBRK, EOF, NOSPACE, QBUSY, NOSTG, ENQBUSY, NOJBUFSP, SIGNAL, OVERFLOW, SYSBUSY, SESSBUSY.

PGHM gate, PUSH_HANDLE function

The PUSH_HANDLE function of the PGHM gate is invoked for a user EXEC CICS PUSH command.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
MISSING_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FASTPATH_FLAGS

Optional Parameter

identifies the fastpath flag settings for the following conditions handled by the user: RDATT, WRBRK, EOF, NOSPACE, QBUSY, NOSTG, ENQBUSY, NOJBUFSP, SIGNAL, OVERFLOW, SYSBUSY, SESSBUSY.

PGHM gate, SET_ABEND function

The SET_ABEND function of the PGHM gate is invoked in response to a user EXEC CICS HANDLE ABEND command, and saves the details of the handle into the current abend Handle Table.

Input Parameters

OPERATION

identifies what is to be done if the abend occurs.

Values for the parameter are:

CANCEL
HANDLE
RESET

AMODE
Optional Parameter

is the addressing mode (24-bit or 31-bit) of the program at the time the handle command was driven.

Values for the parameter are:
AMODE24
AMODE31

CURRENT_EXECUTION_KEY
Optional Parameter

is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE CONDITION command was issued).

LABEL
Optional Parameter

is the token identifying the condition label within the program to be branched to if the abend occurs. Specify either the LABEL parameter or the PROGRAM parameter, not both.

LANGUAGE
Optional Parameter

is the program language.

Values for the parameter are:
ASSEMBLER
COBOL
CPP
C370
PLI

PROGRAM
Optional Parameter

is the name of the program to which control will be passed if the abend occurs. Specify either the LABEL parameter or the PROGRAM parameter, not both.

USERS_RSA_POINTER
Optional Parameter

is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION
MISSING_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGHM gate, SET_AIDS function

The SET_AIDS function of the PGHM gate is invoked in response to a user EXEC CICS HANDLE AID command, and saves the details of the handle into the current aid Handle Table.

Input Parameters

IDENTIFIERS
is the token identifying the conditions to be handled.

LABELS_FLAGS
is the token identifying the number of conditions in this command that have associated labels.

AMODE
Optional Parameter

is the addressing mode (24-bit or 31-bit) of the program at the time the handle command was driven.

Values for the parameter are:
AMODE24
AMODE31

CURRENT_EXECUTION_KEY
Optional Parameter

is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE CONDITION command was issued).

LABELS
Optional Parameter

is the token identifying the condition labels (the locations within the program to be branched to if the condition occurs).

LANGUAGE
Optional Parameter

is the program language.

Values for the parameter are:
ASSEMBLER
COBOL
CPP
C370
PLI

USERS_RSA_POINTER
Optional Parameter

is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION
MISSING_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGHM gate, SET_CONDITIONS function

The SET_CONDITIONS function of the PGHM gate is used to process for user EXEC CICS HANDLE CONDITION commands, and to save the details of the condition into the current condition handle table.

Input Parameters

IDENTIFIERS
is the token identifying the conditions to be handled.

LABELS_FLAGS
is the token identifying the number of conditions in this command that have associated labels.

AMODE
Optional Parameter

is the addressing mode (24-bit or 31-bit) of the program at the time the handle command was driven.

Values for the parameter are:
AMODE24
AMODE31

CURRENT_EXECUTION_KEY
Optional Parameter

is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE CONDITION command was issued).

LABELS
Optional Parameter

is the token identifying the condition labels (the locations within the program to be branched to if the condition occurs).

LANGUAGE
Optional Parameter

is the program language.

Values for the parameter are:
ASSEMBLER
COBOL
CPP
C370
PLI

USERS_RSA_POINTER
Optional Parameter

is the address of the user program Register Save Area into which the program's registers are saved at each EXEC CICS command execution.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION
MISSING_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FASTPATH_FLAGS
Optional Parameter

identifies the fastpath flag settings for the following conditions handled by the user: RDATT, WRBRK, EOF, NOSPACE, QBUSY, NOSTG, ENQBUSY, NOJBUFSP, SIGNAL, OVERFLOW, SYSBUSY, SESSBUSY.

PGIS gate, END_BROWSE_APPLICATION function

The END_BROWSE_APPLICATION function of the PGIS gate is used to end browsing through applications deployed on platforms.

Input Parameters

BROWSE_TOKEN
A browse token that references the next application that is being browsed.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_BROWSE_TOKEN
```

The values for the parameter are as follows:

```
NO_REASON
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGIS gate, END_BROWSE_PROGRAM function

The END_BROWSE_PROGRAM function of the PGIS gate is used to end browsing through program definitions.

Input Parameters

BROWSE_TOKEN
A browse token that references the next container in the container pool that is being browsed.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_BROWSE_TOKEN
```

The values for the parameter are as follows:

```
NO_REASON
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGIS gate, GET_NEXT_APPLICATION function

The GET_NEXT_APPLICATION function of the PGIS gate is used to get the next application to be browsed.

Input Parameters

APPLICATION_NAME
Optional parameter

The name of the application.

BROWSE_TOKEN
A browse token that references the next container in the container pool that is being browsed.

PLATFORM_NAME
Optional parameter

The name of the platform where the application is deployed.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
END_LIST
INVALID_BROWSE_TOKEN
```

The values for the parameter are as follows:

```
NO_REASON
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MAJOR_VERSION
Optional parameter

The major version of the application.

MINOR_VERSION
Optional parameter

The minor version of the application.

MICRO_VERSION
Optional parameter

The micro version of the application.

PGIS gate, GET_NEXT_PROGRAM function

The GET_NEXT_PROGRAM function of the PGIS gate is used to get the next program definition to be browsed.

Input Parameters

APPLICATION_NAME
Optional parameter

The 64-character buffer to receive the name of the application associated with the program.

BROWSE_TOKEN
A browse token that references the next container in the container pool that is being browsed.

JVM_CLASS
Optional Parameter

The name of the main class in a Java program that is running under the control of a Java Virtual Machine (JVM).

OPERATION_NAME
Optional parameter

The 64-character buffer to receive the name of the operation associated with the program.

PLATFORM_NAME
Optional parameter

The 64-character buffer to receive the name of the platform associated with the program.

RESOURCE_SIGNATURE
Optional parameter

A buffer to receive the PROGRAM resource signature.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
END_LIST
INVALID_BROWSE_TOKEN
PROGRAM_NOT_DEFINED_TO_LD
```

The values for the parameter are as follows:

```
NO_REASON
```

PROGRAM_NAME
The name of the program.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESS
Optional Parameter

The type of access for the program.

Values for the parameter are as follows:

```
CICS
NONE
READ_ONLY
USER
```

APIST
Optional Parameter

Indicates whether the program is restricted to use of only the CICS permitted application programming interfaces.

Values for the parameter are as follows:

```
CICSAPI
OPENAPI
```

AVAIL_STATUS
Optional Parameter

Indicates whether the program can be used (ENABLED) or not (DISABLED).

Values for the parameter are as follows:

```
DISABLED
ENABLED
```

CEDF_STATUS
Optional Parameter

Indicates whether the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF).

Values for the parameter are as follows:

CEDF
NOCEDF
NOT_APPLIC

CONCURRENCY

Optional Parameter

Indicates whether the program is threadsafe or only quasi-reentrant.

Values for the parameter are as follows:

QUASIRENT
THREADSAFE

DATA_LOCATION

Optional Parameter

Indicates whether the program can handle only 24-bit addresses (data located below the 16 MB line), or can also handle 31-bit addresses (data located above or below the 16 MB line). The DATALOCATION options are independent from the addressing mode of the link-edited program.

Values for the parameter are as follows:

ANY
BELOW
NOT_APPLIC

DYNAMIC_STATUS

Optional Parameter

Indicates whether a request to LINK to the program can be dynamically routed.

Values for the parameter are as follows:

DYNAMIC
NOTDYNAMIC

ENTRY_POINT

Optional Parameter

The token that defines the entry point of the program.

EXECUTION_KEY

Optional Parameter

The key in which CICS gives control to the program, and that determines whether the program can modify CICS-key storage. If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into the extended read-only DSA (ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

Values for the parameter are as follows:

CICS
NOT_APPLIC
USER

EXECUTION_SET

Optional Parameter

Indicates whether CICS can link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program).

Values for the parameter are as follows:
DPLSUBSET
FULLAPI
NOT_APPLIC

HOLD_STATUS
Optional Parameter

The hold status of the program (that is, for how long the program is to be loaded).

Values for the parameter are as follows:
CICS_LIFE
NOT_APPLIC
TASK_LIFE

INSTALL_TYPE
Optional Parameter

The method used to install the PROGRAM resource definition.

Values for the parameter are as follows:
AUTO
CATALOG
GROUPLIST
MANUAL
RDO
SYSAUTO

JVM
Optional Parameter

Indicates whether the program is run under the control of a Java Virtual Machine (JVM).

Values for the parameter are as follows:
NO
YES

JVM_PROFILE
Optional Parameter

The name of the JVM profile. The named profile provides the attributes of the JVM that are needed to run the program.

JVMPROGRAM_USE_COUNT
Optional Parameter

For Java programs that run under the control of a JVM, the number of times the program has been used.

LANGUAGE_DEDUCED
Optional Parameter

The language deduced by CICS for the program.

Values for the parameter are as follows:
ASSEMBLER
COBOL
COBOL2
C370
JAVA
LE370
NOT_APPLIC
NOT_DEDUCED
PLI

LANGUAGE_DEFINED
Optional Parameter

The language defined for the program.

Values for the parameter are as follows:
ASSEMBLER
COBOL
C370
LE370
NOT_APPLIC
NOT_DEFINED
PLI

LANGUAGE_TOKEN
Optional Parameter

A token that represents the AP domain language block for the program.

LIBRARY
Optional Parameter

The name of the LIBRARY concatenation from which the program was loaded.

LIBRARYDSN
Optional Parameter

The name of the data set in the LIBRARY concatenation from which the program was loaded.

LOAD_POINT
Optional Parameter

The load point address of the program, as returned by the loader domain on the ACQUIRE_PROGRAM call.

LOAD_STATUS
Optional Parameter

The load status of the program (that is, whether or not the program can be loaded).

Values for the parameter are as follows:
LOADABLE
NOT_APPLIC
NOT_LOADABLE
NOT_LOADED

LOCATION
Optional Parameter

Indicates where the program resides.

Values for the parameter are as follows:
CDSA
ECDSA
ELPA
ERDSA
ESDSA
LPA
NONE
RDSA
SDSA

MAJOR_VERSION
Optional parameter

The fullword binary value that is the major version of the application associated with the program.

MINOR_VERSION
Optional parameter

The fullword binary value that is the minor version of the application associated with the program.

MICRO_VERSION
Optional parameter

The fullword binary value that is the micro version of the application associated with the program.

MODULE_TYPE
Optional Parameter

The type of program resource to be defined.

Values for the parameter are as follows:
```
MAPSET
PARTITIONSET
PROGRAM
```

NEW_PROGRAM_TOKEN
Optional Parameter

The token assigned to program.

PROGRAM_ATTRIBUTE
Optional Parameter

Indicates the residence status of the program, and when the storage for this program is released.

Values for the parameter are as follows:
```
RELOAD
RESIDENT
REUSABLE
TEST
TRANSIENT
```

PROGRAM_LENGTH
Optional Parameter

The length of the program, as returned by the loader domain on the ACQUIRE_PROGRAM call.

PROGRAM_TYPE
Optional Parameter

The type of program.

Values for the parameter are as follows:
```
NOT_APPLIC
PRIVATE
SHARED
TYPE_ANY
```

PROGRAM_USAGE
Optional Parameter

Indicates whether the program is used as a CICS nucleus program, or as a user application program.

Values for the parameter are as follows:
```
APPLICATION
NUCLEUS
```

PROGRAM_USE_COUNT
Optional Parameter

The number of times that the program has been used.

PROGRAM_USER_COUNT
Optional Parameter

The number of different users that have invoked the program.

REMOTE_DEFINITION
Optional Parameter

Indicates whether the program is defined as remote or local.

Values for the parameter are as follows:

```
LOCAL
REMOTE
```

REMOTE_PROGID
Optional Parameter

The name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value.

REMOTE_SYSID
Optional Parameter

The name of a remote CICS region if CICS can ship a distributed program link (DPL) request to another CICS region.

REMOTE_TRANID
Optional Parameter

The name of the transaction the remote CICS can attach to, and under which it can run the remote program.

RUNTIME_ENVIRONMENT
Optional Parameter

Indicates the runtime environment used to run this program.

Values for the parameter are as follows:

```
JVM_RUNTIME
LE370_RUNTIME
NON_LE370_RUNTIME
NOT_APPLIC
UNKNOWN_RUNTIME
XPLINK_RUNTIME
```

SPECIFIED_AMODE
Optional Parameter

The addressing mode of the program.

Values for the parameter are as follows:

```
AMODE_ANY
AMODE_NOT_SPECIFIED
24
31
64
```

SPECIFIED_RMODE
Optional Parameter

The residence mode of the program.

Values for the parameter are as follows:

RMODE_ANY
RMODE_NOT_SPECIFIED
24

PGIS gate, INQUIRE_CURRENT_PROGRAM function

The INQUIRE_CURRENT_PROGRAM function of the PGIS gate is used to inquire about the current attributes of a program (for the current invocation of the program).

Input Parameters

PROGRAM_TOKEN

Optional parameter

A token that identifies the program to inquire about.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOCK_ERROR

The following values are returned when RESPONSE is EXCEPTION:

NO_CURRENT_PROGRAM

The values for the parameter are as follows:

NO_REASON

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

AVAIL_STATUS

Optional parameter

Indicates whether the program can be used (ENABLED) or not (DISABLED).

Values for the parameter are as follows:

DISABLED
ENABLED

CEDF_STATUS

Optional parameter

Indicates whether the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF)

Values for the parameter are as follows:

CEDF
NOCEDF
NOT_APPLIC

CONCURRENCY

Optional Parameter

Indicates whether the program is threadsafe or only quasi-reentrant.

Values for the parameter are as follows:

QUASIRENT
REQUIRED
THREADSAFE

CURRENT_AMODE

Optional parameter

The current addressing mode of the program.

Values for the parameter are as follows:

24
31
64

CURRENT_CEDF_STATUS
Optional parameter

Indicates whether the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF).

Values for the parameter are as follows:
CEDF
NOCEDF

CURRENT_ENTRY_POINT
Optional parameter

The current entry point address of the program, as returned by the loader domain on the ACQUIRE_PROGRAM call.

CURRENT_ENVIRONMENT
Optional parameter

Indicates the current environment in which the program is running.

Values for the parameter are as follows:
EXEC
GLUE
PLT
SYSTEM
TRUE
URM

CURRENT_EXECUTION_SET
Optional parameter

Indicates whether the program is running with or without the API restrictions of a DPL program.

Values for the parameter are as follows:
DPLSUBSET
FULLAPI

CURRENT_LOAD_POINT
Optional parameter

The current load point address of the program, as returned by the loader domain on the ACQUIRE_PROGRAM call.

CURRENT_PROGRAM_LENGTH
Optional parameter

The length of the current program in bytes, as returned by the loader domain on the AQUIRE_PROGRAM call.

CURRENT_PROGRAM_NAME
Optional parameter

The current name of the program.

DATA_LOCATION
Optional parameter

Indicates whether the program can handle only 24-bit addresses (data located below the 16 MB line), or can also handle 31-bit addresses (data located above or below the 16 MB line). The DATALOCATION options are independent from the addressing mode of the link-edited program.

Values for the parameter are as follows:
ANY

BELOW
NOT_APPLIC

DYNAMIC_STATUS
Optional parameter

Indicates whether a request to LINK to the program can be dynamically routed.

Values for the parameter are as follows:
DYNAMIC
NOTDYNAMIC

EXECUTION_KEY
Optional parameter

The key in which CICS gives control to the program, and that determines whether the program can modify CICS-key storage. If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into the extended read-only DSA (ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

Values for the parameter are as follows:
CICS
NOT_APPLIC
USER

EXECUTION_SET
Optional parameter

Indicates whether CICS can link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program).

Values for the parameter are as follows:
DPLSUBSET
FULLAPI
NOT_APPLIC

HOLD_STATUS
Optional parameter

The hold status of the program (that is, for how long the program is to be loaded).

Values for the parameter are as follows:
CICS_LIFE
NOT_APPLIC
TASK_LIFE

IGNORE_EXITS
Optional parameter

Indicates whether global user exit programs and task-related user exit programs are ignored when returning information about the program that invoked this program and to which control will be returned.

Values for the parameter are as follows:
YES
NO

INSTALL_TYPE
Optional parameter

The method used to install the PROGRAM resource definition.

Values for the parameter are as follows:

AUTO
CATALOG
GROUPLIST
MANUAL
RDO
SYSAUTO

INVOKING_ENVIRONMENT

Optional parameter

The environment in which the program that invoked this program was running.

Values for the parameter are as follows:

EXEC
GLUE
PLT
SYSTEM
TRUE
URM

INVOKING_PROGRAM_NAME

Optional parameter

The name of the program that invoked this program.

LANGUAGE_DEDUCED

Optional parameter

The language deduced by CICS for the program.

Values for the parameter are as follows:

ASSEMBLER
COBOL
COBOL2
C370
JAVA
LE370
NOT_APPLIC
NOT_DEDUCED
PLI

LANGUAGE_DEFINED

Optional parameter

The language defined for the program.

Values for the parameter are as follows:

ASSEMBLER
COBOL
C370
LE370
NOT_APPLIC
NOT_DEFINED
PLI

LIBRARY

Optional parameter

The name of the LIBRARY concatenation from which the program was loaded.

LIBRARYDSN

Optional parameter

The name of the data set in the LIBRARY concatenation from which the program was loaded.

LOAD_STATUS
Optional parameter

The load status of the program (that is, whether the program can be loaded).

Values for the parameter are as follows:
- LOADABLE
- NOT_APPLIC
- NOT_LOADABLE
- NOT_LOADED

MODULE_TYPE
Optional parameter

The type of program resource.

Values for the parameter are as follows:
- MAPSET
- PARTITIONSET
- PROGRAM

NEW_PROGRAM_TOKEN
Optional parameter

The token assigned to the program.

REMOTE_DEFINITION
Optional parameter

Indicates whether the program is defined as remote or local.

Values for the parameter are as follows:
- LOCAL
- REMOTE

REMOTE_PROGID
Optional parameter

The name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value).

REMOTE_SYSID
Optional parameter

The name of a remote CICS region if CICS can ship a distributed program link (DPL) request to another CICS region.

REMOTE_TRANID
Optional parameter

The name of the transaction the remote CICS can attach to, and under which it can run the remote program.

RETURN_PROGRAM_NAME
Optional parameter

The name of the program to which control will be returned when this program ends.

PGIS gate, INQUIRE_PROGRAM function

The INQUIRE_PROGRAM function of the PGIS gate is used to inquire about attributes of a program.

Input Parameters

AC_APPL_CONTEXT_TOKEN
Optional parameter

A token that defines the application context for a PROGRAM. An alternative to specifying **AC_PLATFORM_NAME**, **AC_APPLICATION_NAME**, **AC_MAJOR_VERSION**, **AC_MINOR_VERSION**, and **AC_MICRO_VERSION**.

AC_APPLICATION_NAME
Optional parameter

The 64-character name of the application associated with the program being inquired upon.

AC_MAJOR_VERSION
Optional parameter

The fullword binary value that is the major version of the application associated with the program being inquired upon.

AC_MINOR_VERSION
Optional parameter

The fullword binary value that is the minor version of the application associated with the program being inquired upon.

AC_MICRO_VERSION
Optional parameter

The fullword binary value that is the micro version of the application associated with the program being inquired upon.

AC_PLATFORM_NAME
Optional parameter

The 64-character name of the platform associated with the program being inquired upon.

APPLICATION_NAME
Optional parameter

The 64-character buffer to receive the name of the application associated with the program.

OPERATION_NAME
Optional parameter

The 64-character buffer to receive the name of the operation associated with the program.

PLATFORM_NAME
Optional parameter

The 64-character buffer to receive the name of the platform associated with the program.

PROGRAM_NAME
The name of the program resource to inquire about.

PROGRAM_TOKEN
The token that identifies the program to inquire about.

RESOURCE_SIGNATURE
Optional parameter

A buffer to receive the PROGRAM resource signature.

SHOW_PROGRAMS
This field is required if an application context is supplied using the fields beginning **AC_**. It specifies whether private programs only, or first private and then public programs, are searched.

Values for the parameter are as follows:

PRIVATE
PRIVATE_AND_PUBLIC

If this field is specified but no application context is supplied, an inquiry with PRIVATE_AND_PUBLIC searches only public programs, and an inquiry with PRIVATE returns a not found response.

JVM_CLASS

Optional Parameter

The name of the main class in a Java program to be run under the control of a Java Virtual Machine (JVM).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOCK_ERROR

The following values are returned when RESPONSE is EXCEPTION:

PROGRAM_NOT_DEFINED_TO_LD
PROGRAM_NOT_DEFINED_TO_PG
APP_CONTEXT_NOT_FOUND

The following values are returned when RESPONSE is INVALID:

INVALID_PROGRAM_TOKEN

The values for the parameter are as follows:

NO_REASON

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESS

Optional Parameter

The type of access for the program.

Values for the parameter are as follows:

CICS
NONE
READ_ONLY
USER

APIST

Optional Parameter

Indicates whether the program is restricted to use only of the CICS permitted application programming interfaces.

Values for the parameter are as follows:

CICSAPI
OPENAPI

AVAIL_STATUS

Optional Parameter

Indicates whether the program can be used (ENABLED) or not (DISABLED).

Values for the parameter are as follows:

DISABLED
ENABLED

CEDF_STATUS

Optional Parameter

Indicates whether the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF).

Values for the parameter are as follows:

```
CEDF
NOCEDF
NOT_APPLIC
```

CONCURRENCY

Optional Parameter

Indicates whether the program is threadsafe or only quasi-reentrant.

Values for the parameter are as follows:

```
QUASIRENT
REQUIRED
THREADSAFE
```

DATA_LOCATION

Optional Parameter

Defines whether the program can handle only 24-bit addresses (data located below the 16 MB line) or can also handle 31-bit addresses (data located above or below the 16 MB line). The DATALOCATION options are independent from the addressing mode of the link-edited program.

Values for the parameter are as follows:

```
ANY
BELOW
NOT_APPLIC
```

DYNAMIC_STATUS

Optional Parameter

Indicates whether a request to LINK to the program can be dynamically routed.

Values for the parameter are as follows:

```
DYNAMIC
NOTDYNAMIC
```

ENTRY_POINT

Optional Parameter

The token that defines the entry point of the program.

ENTRYPOINT_BUN_TOKEN

Optional parameter

An 8-byte token that represents the bundle that defined the application entry point.

ENTRYPOINT_RES_TOKEN

Optional parameter

An 8-byte token that represents the application entry point.

EXECUTION_KEY

Optional Parameter

The key in which CICS gives control to the program, and that determines whether the program can modify CICS-key storage. If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into the extended read-only DSA (ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

Values for the parameter are as follows:

CICS
NOT_APPLIC
USER

EXECUTION_SET
Optional Parameter

Indicates whether CICS can link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program).

Values for the parameter are as follows:
DPLSUBSET
FULLAPI
NOT_APPLIC

HOLD_STATUS
Optional Parameter

The hold status of the program (that is, for how long the program is to be loaded).

Values for the parameter are as follows:
CICS_LIFE
NOT_APPLIC
TASK_LIFE

INSTALL_TYPE
Optional Parameter

The method used to install the PROGRAM resource definition.

Values for the parameter are as follows:
AUTO
CATALOG
GROUPLIST
MANUAL
RDO
SYSAUTO

JVM
Optional Parameter

Indicates whether the program can run under the control of a JVM (Java Virtual Machine).

Values for the parameter are as follows:
NO
YES

JVM_PROFILE
Optional Parameter

Specifies the name of the JVM profile. The named profile provides the attributes of the JVM that are needed to run the program.

JVMPROGRAM_USE_COUNT
Optional Parameter

For Java programs that run under the control of a JVM, the number of times the program has been used.

LANGUAGE_DEDUCED
Optional Parameter

The language deduced by CICS for the program.

Values for the parameter are as follows:
ASSEMBLER

COBOL
COBOL2
C370
JAVA
LE370
NOT_APPLIC
NOT_DEDUCED
PLI

LANGUAGE_DEFINED
Optional Parameter

The language defined for the program.

Values for the parameter are as follows:
ASSEMBLER
COBOL
C370
LE370
NOT_APPLIC
NOT_DEFINED
PLI

LANGUAGE_TOKEN
Optional Parameter

A token that represents the AP domain language block for the program.

LIBRARY
Optional Parameter

The name of the LIBRARY concatenation from which the program was loaded.

LIBRARYDSN
Optional Parameter

The name of the data set in the LIBRARY concatenation from which the program was loaded.

LOAD_POINT
Optional Parameter

The load point address of the program, as returned by the loader domain on the ACQUIRE_PROGRAM call.

LOAD_STATUS
Optional Parameter

The load status of the program (that is, whether the program can be loaded).

Values for the parameter are as follows:
LOADABLE
NOT_APPLIC
NOT_LOADABLE
NOT_LOADED

LOADER_TOKEN
Optional Parameter

The token that the loader domain uses to identify the program.

LOCATION
Optional Parameter

Indicates where the program resides.

Values for the parameter are as follows:
CDSA
ECDSA

ELPA
ERDSA
ESDSA
LPA
NONE
RDSA
SDSA

MAJOR_VERSION
Optional parameter

The fullword binary value that is the major version of the application associated with the program.

MINOR_VERSION
Optional parameter

The fullword binary value that is the minor version of the application associated with the program.

MICRO_VERSION
Optional parameter

The fullword binary value that is the micro version of the application associated with the program.

MODULE_TYPE
Optional Parameter

The type of program resource.

Values for the parameter are as follows:
MAPSET
PARTITIONSET
PROGRAM

NEW_PROGRAM_TOKEN
Optional Parameter

The token assigned to the program.

PROGRAM_ATTRIBUTE
Optional Parameter

The residence status of the program, and when the storage for this program is released.

Values for the parameter are as follows:
RELOAD
RESIDENT
REUSABLE
TEST
TRANSIENT

PROGRAM_LENGTH
Optional Parameter

The length of the program, as returned by the loader domain on the ACQUIRE_PROGRAM call.

PROGRAM_TYPE
Optional Parameter

The type of program.

Values for the parameter are as follows:
NOT_APPLIC
PRIVATE
SHARED

TYPE_ANY

PROGRAM_USAGE
Optional Parameter

Whether the program is used as a CICS nucleus program, or as a user application program.

Values for the parameter are as follows:
APPLICATION
NUCLEUS

PROGRAM_USE_COUNT
Optional Parameter

The number of times that the program has been used.

PROGRAM_USER_COUNT
Optional Parameter

The number of different users that have invoked the program.

REMOTE_DEFINITION
Optional Parameter

Indicates whether the program is defined as remote or local.

Values for the parameter are as follows:
LOCAL
REMOTE

REMOTE_PROGID
Optional Parameter

The name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value.

REMOTE_SYSID
Optional Parameter

The name of a remote CICS region if CICS can ship a distributed program link (DPL) request to another CICS region.

REMOTE_TRANID
Optional Parameter

The name of the transaction the remote CICS can attach to, and under which it can run the remote program.

RUNTIME_ENVIRONMENT
Optional Parameter

Indicates the runtime environment used to run this program.

Values for the parameter are as follows:
JVM_RUNTIME
LE370_RUNTIME
NON_LE370_RUNTIME
NOT_APPLIC
UNKNOWN_RUNTIME
XPLINK_RUNTIME

SPECIFIED_AMODE
Optional Parameter

The addressing mode of the program.

Values for the parameter are as follows:
AMODE_ANY

AMODE_NOT_SPECIFIED
24
31
64

SPECIFIED_RMODE
Optional Parameter

The residence mode of the program.

Values for the parameter are as follows:
RMODE_ANY
RMODE_NOT_SPECIFIED
24

PGIS gate, REFRESH_PROGRAM function

The REFRESH_PROGRAM function of the PGIS gate is used to inform the loader domain that a new copy of a named program is now available for use in the relocatable program library.

Input Parameters

COPY
Indicates whether a NEWCOPY or PHASEIN function is required.

Values for the parameter are as follows:
NEWCOPY
PHASEIN

PROGRAM_NAME
The name of the program resource to be refreshed.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_ERROR

The following values are returned when RESPONSE is EXCEPTION:
PROGRAM_IN_USE
PROGRAM_LOADED_CICS_LIFE
PROGRAM_NOT_DEFINED_TO_LD
PROGRAM_NOT_DEFINED_TO_PG
PROGRAM_NOT_FOUND
REMOTE_PROGRAM

The values for the parameter are as follows:
NO_REASON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

VERSION
Optional Parameter

The version of the program after the REFRESH_PROGRAM function call.

Values for the parameter are as follows:
NEW
OLD

PGIS gate, SET_PROGRAM function

The SET_PROGRAM function of the PGIS gate is used to set the characteristics of a program when it is loaded.

Input Parameters

PROGRAM_NAME
The name of the program resource to be defined.

PROGRAM_TOKEN
The token that identifies the program to be loaded.

APPLICATION_NAME
Optional parameter

The 64-character name of the application that is to be associated with the program.

AVAIL_STATUS
Optional Parameter

Defines whether the program can be used (ENABLED) or not (DISABLED).

Values for the parameter are as follows:
```
DISABLED
ENABLED
```

BUNDLE_CALLER
Optional Parameter

Defines whether the caller is either the RL client (YES) or a CICS internal module (CICSOVERRIDE) and therefore allowed to modify the state of a program that is defined in a CICS bundle.

Values for the parameter are as follows:
```
CICSOVERRIDE
NO
YES
```

CEDF_STATUS
Optional Parameter

Indicates whether the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF).

Values for the parameter are as follows:
```
CEDF
NOCEDF
```

ENTRYPOINT_BUN_TOKEN
Optional parameter

An 8-byte token that represents the bundle that defined the application entry point.

ENTRYPOINT_RES_TOKEN
Optional parameter

An 8-byte token that represents the application entry point.

EXECUTION_KEY
Optional Parameter

The key in which CICS gives control to the program, and that determines whether the program can modify CICS-key storage. If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

Values for the parameter are as follows:
```
CICS
USER
```

EXECUTION_SET
Optional Parameter

Indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program).

Values for the parameter are as follows:
```
DPLSUBSET
FULLAPI
```
JVM
Optional Parameter

Indicates whether the program is to run under the control of a Java Virtual Machine (JVM).

Values for the parameter are as follows:
```
NO
YES
```
JVM_CLASS
Optional Parameter

The name of the main class in a Java program to be run under the control of a JVM.

JVM_PROFILE
Optional Parameter

Specifies the name of the data set member that contains the JVM profile. The named profile provides the attributes of the JVM that are needed to run the program.

MAJOR_VERSION
Optional parameter

The fullword binary value that is the major version of the application to be associated with the program.

MINOR_VERSION
Optional parameter

The fullword binary value that is the minor version of the application to be associated with the program.

MICRO_VERSION
Optional parameter

The fullword binary value that is the micro version of the application to be associated with the program.

OPERATION_NAME
Optional parameter

The 64-character buffer to receive the name of the application operation to be associated with the program.

PLATFORM_NAME
Optional parameter

The 64-character buffer to receive the name of the platform to be associated with the program.

PROGRAM_ATTRIBUTE
Optional Parameter

Defines the residence status of the program, and when the storage for this program is released.

Values for the parameter are as follows:

RELOAD
RESIDENT
REUSABLE
TEST
TRANSIENT

PROGRAM_TYPE
Optional Parameter

The type of program.

Values for the parameter are as follows:
PRIVATE
SHARED
TYPE_ANY

PROGRAM_USAGE
Optional Parameter

Defines whether the program is to be used as a CICS nucleus program, or as a user application program.

Values for the parameter are as follows:
APPLICATION
NUCLEUS

REQUIRED_AMODE
Optional Parameter

The addressing mode of the program.

Values for the parameter are as follows:
AMODE_ANY
24
31
64

REQUIRED_RMODE
Optional Parameter

The residence mode of the program.

Values for the parameter are as follows:
RMODE_ANY
24

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
CATALOG_ERROR
CATALOG_NOT_OPERATIONAL
INSUFFICIENT_STORAGE
LOCK_ERROR

The following values are returned when RESPONSE is EXCEPTION:
APP_CONTEXT_NOT_FOUND
CEDF_STATUS_NOT_FOR_MAPSET
CEDF_STATUS_NOT_FOR_PTNSET
CEDF_STATUS_NOT_FOR_REMOTE
CONTEXT_ALREADY_SET
DEBUG_BUT_NO_JVM
DUPLICATE_OPERATION
EXEC_KEY_NOT_FOR_MAPSET

EXEC_KEY_NOT_FOR_PTNSET
EXEC_KEY_NOT_FOR_REMOTE
EXEC_SET_NOT_FOR_MAPSET
EXEC_SET_NOT_FOR_PTNSET
EXEC_SET_NOT_FOR_REMOTE
INCOMPATIBLE_BUNDLE_SET
INCOMPATIBLE_LIB_SET
JVM_BUT_NO_JVMCLASS
JVMSERVER_PROGRAM
OPERATION_NOT_FOR_REMOTE
OPERATION_NOT_FOR_MAPSET
OPERATION_NOT_FOR_PTNSET
OPERATION_NOT_FOR_REMOTN
NAME_CLASH
PROG_TYPE_NOT_FOR_REMOTE
PROGRAM_NOT_DEFINED_TO_LD
PROGRAM_NOT_DEFINED_TO_PG
PROGRAM_NOT_FOUND

The following values are returned when RESPONSE is INVALID:

INVALID_MODE_COMBINATION
INVALID_PROGRAM_NAME
INVALID_PROGRAM_TOKEN
INVALID_TYPE_ATTRIB_COMBIN

The values for the parameter are as follows:

NO_REASON

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGIS gate, START_BROWSE_APPLICATION function

The START_BROWSE_APPLICATION function of the PGIS gate is used to start browsing through applications that are deployed on platforms.

Input Parameters

None.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
INVALID_DIRECTORY
LOCK_ERROR

The values for the parameter are:

NO_REASON

BROWSE_TOKEN

A browse token that references a container in the container pool. This container is the first in the browse list.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGIS gate, START_BROWSE_PROGRAM function

The START_BROWSE_PROGRAM function of the PGIS gate is used to start browsing through program definitions, optionally starting at the given program definition.

Input Parameters

AC_APPLICATION_NAME
Optional parameter

The 64-character name of the application associated with the program being inquired upon.

AC_MAJOR_VERSION
Optional parameter

The fullword binary value that is the major version of the application associated with the program being inquired upon.

AC_MINOR_VERSION
Optional parameter

The fullword binary value that is the minor version of the application associated with the program being inquired upon.

AC_MICRO_VERSION
Optional parameter

The fullword binary value that is the micro version of the application associated with the program being inquired upon.

AC_PLATFORM_NAME
Optional parameter

The 64-character name of the platform associated with the program being inquired upon.

PROGRAM_NAME
Optional Parameter

The name of the program resource to be browsed.

SHOW_PROGRAMS
Optional Parameter

This field is required if an application context is supplied using the fields beginning **AC_**. It specifies whether private programs only, or first private and then public programs, are searched.

Values for the parameter are as follows:

```
PRIVATE
PRIVATE_AND_PUBLIC
```

If this field is specified but no application context is supplied, an inquiry with `PRIVATE_AND_PUBLIC` searches only public programs, and an inquiry with `PRIVATE` returns a not found response.

TASK_RELATED
Optional Parameter

Indicates whether the browse is task-related. If it is task-related, storage will be obtained from the CICS storage class rather than the directory browse subpool. The default is YES.

Values for the parameter are as follows:

```
NO
YES
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
APP_CONTEXT_NOT_FOUND
INVALID_DIRECTORY
LOCK_ERROR
```

The values for the parameter are:

```
NO_REASON
```

BROWSE_TOKEN

A browse token that references a container in the container pool. This container is the first in the browse list.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGLD gate, LOAD function

The LOAD function of the PGLD gate is used to load a program in response to a CICS internal load request.

Input Parameters

HOLD_LIFETIME

determines for how long the program is to be loaded; that is, for the life-time of CICS (or until explicitly deleted) or for the lifetime of the task (unless explicitly deleted by the task).

Values for the parameter are:

```
CALLER_MANAGED
CICS_LIFE
TASK_LIFE
```

MODULE_TYPE

is the type of program resource to be defined.

Values for the parameter are:

```
MAPSET
PARTITIONSET
PROGRAM
```

PROGRAM_NAME

is the name of the program resource to be defined.

SYSTEM_AUTOINSTALL

defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition.

Values for the parameter are:

```
NO
YES
```

LPA_ELIGIBLE

Optional Parameter

defines whether or not the program can be loaded into the MVS link pack area (LPA).

Values for the parameter are:

```
NO
YES
```

SUSPEND

Optional Parameter

This option is passed to the LDLD call, and thence to SMGF. It specifies the action in the event of a storage shortage.YES, the default value, means that the task will be suspended until storage is available. NO means that the task will be abended.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
JVM_PROGRAM
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
REMOTE_PROGRAM

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

ENTRY_POINT

is the token defining the entry point of the program.

LOAD_POINT

is the load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

PROGRAM_LENGTH

is the length of the program. returned by the loader domain on the ACQUIRE_PROGRAM call.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGLD gate, LOAD_EXEC function

The LOAD_EXEC function of the PGLD gate is used to load a program in response to an EXEC CICS LOAD command.

Input Parameters

HOLD_LIFETIME

determines for how long the program is to be loaded; that is, for the life-time of CICS (or until explicitly deleted) or for the lifetime of the task (unless explicitly deleted by the task).

Values for the parameter are:
CALLER_MANAGED
CICS_LIFE
TASK_LIFE

PROGRAM_NAME

is the name of the program resource to be defined.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA

AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
JVM_PROGRAM
NOT_AUTHORIZED
NOT_INITIALIZED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
REMOTE_PROGRAM

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

ENTRY_POINT
is the token defining the entry point of the program.

LOAD_POINT
is the load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

PROGRAM_LENGTH
is the length of the program. returned by the loader domain on the ACQUIRE_PROGRAM call.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LANGUAGE_TOKEN
Optional Parameter

is a token representing the AP domain language block for the program.

PGLD gate, RELEASE function

The RELEASE function of the PGLD gate is used by CICS internal modules to release a program in response previously loaded by a PGLD LOAD request.

Input Parameters

PROGRAM_NAME
is the name of the program resource to be defined.

ENTRY_POINT
Optional Parameter

must be provided on RELEASE_EXEC by the caller for a program loaded with caller-managed lifetime.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
JVM_PROGRAM
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_IN_USE
PROGRAM_NOT_LOADED
PROGRAM_RELOAD_YES
REMOTE_PROGRAM

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGLD gate, RELEASE_EXEC function

The RELEASE_EXEC function of the PGLD gate is used to release a program in response to an EXEC CICS RELEASE command.

Input Parameters

PROGRAM_NAME
is the name of the program resource to be defined.

ENTRY_POINT
Optional Parameter

must be provided on RELEASE_EXEC by the caller for a program loaded with caller-managed lifetime.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
JVM_PROGRAM
NOT_AUTHORIZED
NOT_INITIALIZED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_IN_USE
PROGRAM_NOT_LOADED
PROGRAM_RELOAD_YES
RELEASE_ISSUING_PROGRAM
REMOTE_PROGRAM
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGLE gate, LINK_EXEC function

The LINK_EXEC function of the PGLE gate is used to link to a program in response to a user EXEC CICS LINK command.

Input Parameters

PROGRAM_NAME
is the name of the program resource to be defined.

APPLICATION_ID
Optional Parameter

is the name of the application whose entry point program is being linked to.

APPL_MAJOR_VER
Optional Parameter

is the major version number of the application whose entry point program is being linked to.

APPL_MICRO_VER
Optional Parameter

is the micro version number of the application whose entry point program is being linked to.

APPL_MINOR_VER
Optional Parameter

is the minor version number of the application whose entry point program is being linked to.

CHANNEL
Optional Parameter

is the optional channel to be made available to the linked program.

COMMAREA
Optional Parameter

is the optional communications area to be made available to the linked program.

FORCE_LOCAL
Optional Parameter

indicates whether the program must execute locally.

Values for the parameter are:
NO
YES

HANDLE_ABEND_PGM
Optional Parameter

defines whether or not the program is to run as an abend handler program.

Values for the parameter are:
NO
YES

INPUTMSG
Optional Parameter

is a data area to be supplied to the linked program on its first execution of an EXEC CICS RECEIVE command.

PLATFORM_ID
Optional Parameter

is the name of the platform on which the application is installed, whose entry point program is being linked to.

SYNCONRETURN
Optional Parameter

defines whether or not a syncpoint is to be taken on return from the linked program.

Values for the parameter are:
NO
YES

SYSEIB_REQUEST
Optional Parameter

Specifies whether the EXEC CICS LINK had the SYSEIB translator option specified.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
AUTOSTART_DISABLED

DESTRUCTIVE_OVERLAP
DYNAMIC_PGM
INVALID_CHANNEL_NAME
INVALID_COMMAREA_ADDR
INVALID_COMMAREA_LEN
INVALID_INPUTMSG_LEN
INVALID_KEYWORDS
INVALID_TERMINAL_TYPE
JVM_PROFILE_NOT_FOUND
JVM_PROFILE_NOT_VALID
JVMPOOL_DISABLED
NO_TERMINAL
NOT_INITIALIZED
PROGRAM_NOT_AUTHORISED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
REMOTE_PROGRAM
SECOND_H8_PROGRAM
SECOND_JVM_PROGRAM
SYSTEM_PROPERTIES_NOT_FND
TRANSACTION_ABEND
USER_CLASS_NOT_FOUND

ABEND_CODE
is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

REMOTE_PROGRAM_NAME
is the name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value).

REMOTE_SYSID
is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

REMOTE_TRANID
is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGLK gate, LINK function

The LINK function of the PGLK gate is used by CICS internal modules to link to a program.

Input Parameters

PROGRAM_NAME
is the name of the program resource to be defined.

SYSTEM_AUTOINSTALL
defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition.

Values for the parameter are:
NO
YES

LPA_ELIGIBLE
Optional Parameter

defines whether or not the program can be loaded into the link pack area (LPA).

Values for the parameter are:

NO

YES

PARMLIST_PTR

Optional Parameter

is the address of a parameter list passed by the CICS program initiating the PGLK link to the new program.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
AUTOSTART_DISABLED
JVM_PROFILE_NOT_FOUND
JVM_PROFILE_NOT_VALID
JVMPOOL_DISABLED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
REMOTE_PROGRAM
SECOND_H8_PROGRAM
SECOND_JVM_PROGRAM
SYSTEM_PROPERTIES_NOT_FND
TRANSACTION_ABEND
USER_CLASS_NOT_FOUND

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

ABEND_CODE

is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGLK gate, LINK_PLT function

The LINK_PLT function of the PGLK gate is used by CICS internal modules to link to a program in the program list table.

Input Parameters

PROGRAM_NAME

is the name of the program resource to be defined.

SYSTEM_AUTOINSTALL

defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition.

Values for the parameter are:

NO

YES

LPA_ELIGIBLE

Optional Parameter

defines whether or not the program can be loaded into the link pack area (LPA).

Values for the parameter are:
NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:
AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
AUTOSTART_DISABLED
JVM_PROFILE_NOT_FOUND
JVM_PROFILE_NOT_VALID
JVMPOOL_DISABLED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
REMOTE_PROGRAM
SECOND_H8_PROGRAM
SECOND_JVM_PROGRAM
SYSTEM_PROPERTIES_NOT_FND
TRANSACTION_ABEND
USER_CLASS_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

ABEND_CODE

is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGLU gate, LINK_URM function

The LINK_URM function of the PGLU gate is used by CICS internal modules to link to a user-replaceable program.

Input Parameters

PROGRAM_NAME

is the name of the program resource to be defined.

SYSTEM_AUTOINSTALL

defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition.

Values for the parameter are:
NO
YES

CALLER_THREADSAFE

Optional Parameter

indicates that the caller of the user-replaceable program is threadsafe, and so execution can continue on any TCB on return from the program: there is no need for PGLU to issue change_mode.

Values for the parameter are:

NO
YES

COMMAREA
Optional Parameter

is the optional communications area to be made available to the linked program.

LPA_ELIGIBLE
Optional Parameter

defines whether or not the program can be loaded into the link pack area (LPA).

Values for the parameter are:
NO
YES

CHANNEL_TOKEN
Optional Parameter

a token representing a channel to be made available to the linked program.

SYSTEM_EXIT
Optional Parameter

indicates that the linked program is to be treated as a system exit.

Values for the parameter are:
NO
YES - the default value is YES.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
AMODE_ERROR
AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
AUTOSTART_DISABLED
DESTRUCTIVE_OVERLAP
INVALID_COMMAREA_ADDR
INVALID_COMMAREA_LEN
JVM_PROFILE_NOT_FOUND
JVM_PROFILE_NOT_VALID
JVMPOOL_DISABLED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
REMOTE_PROGRAM
SECOND_H8_PROGRAM
SECOND_JVM_PROGRAM
SYSTEM_PROPERTIES_NOT_FND
URM_ABEND
USER_CLASS_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

ABEND_CODE
is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGPG gate, INITIAL_LINK function

The INITIAL_LINK function of the PGPG gate is used to link to the first program of a transaction.

Input Parameters

PROGRAM_NAME
is the name of the program resource to be defined.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
AUTOSTART_DISABLED
JVM_PROFILE_NOT_FOUND
JVM_PROFILE_NOT_VALID
JVMPOOL_DISABLED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
REMOTE_PROGRAM
SECOND_H8_PROGRAM
SECOND_JVM_PROGRAM
SYSTEM_PROPERTIES_NOT_FND
TRANSACTION_ABEND
USER_CLASS_NOT_FOUND
```

ABEND_CODE
is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGRE gate, PREPARE_RETURN_EXEC function

The PREPARE_RETURN_EXEC function of the PGRE gate is used to process the communications area, inputmsg data, and transaction identifier from a user EXEC CICS RETURN command.

Input Parameters

CHANNEL
Optional Parameter

is the optional channel to be made available to the linked program.

COMMAREA
Optional Parameter

is the optional communications area to be made available to the linked program.

ENDACTIVITY
Optional Parameter

indicates that a BTS activity is to be ended.

Values for the parameter are:
YES

IMMEDIATE
Optional Parameter

Indicates whether or not the transaction specified in TRANSID is to be attached as the next transaction regardless of any other transactions enqueued by ATI for this terminal.

Values for the parameter are:
YES

INPUTMSG
Optional Parameter

is a data area to be supplied to the linked program on its first execution of an EXEC CICS RECEIVE command.

TRANSID
Optional Parameter

is the four-character transaction identifier.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_CHANNEL_NAME
INVALID_COMMAREA_ADDR
INVALID_COMMAREA_LEN
INVALID_INPUTMSG_LEN
INVALID_KEYWORDS
INVALID_REQUEST_FROM_EXIT
INVALID_RETURN_REQUEST
INVALID_TERMINAL_TYPE
NO_TERMINAL
NOT_INITIALIZED
TRANSID_NO_TERMINAL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGXE gate, PREPARE_XCTL_EXEC function

The PREPARE_XCTL_EXEC function of the PGXE gate processes the communications area, inputmsg data, and transaction identifier from a user **EXEC CICS XCTL** command.

Input Parameters

PROGRAM_NAME
The name of the program resource to be defined.

CHANNEL
Optional Parameter

The optional channel to be made available to the linked program.

COMMAREA
Optional Parameter

The optional communications area to be made available to the linked program.

INPUTMSG
Optional Parameter

A data area to be supplied to the linked program on its first execution of an **EXEC CICS RECEIVE** command.

SECURITY
Optional Parameter

Indicates whether Program Manager must check security authorization for the target program

Values for the parameter are:
NO
YES

SYSEIB_REQUEST
Optional Parameter

Specifies whether the **EXEC CICS LINK** had the SYSEIB translator option specified.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
AUTOINSTALL_FAILED
AUTOINSTALL_INVALID_DATA
AUTOINSTALL_MODEL_NOT_DEF
AUTOINSTALL_URM_FAILED
DESTRUCTIVE_OVERLAP
INVALID_CHANNEL_NAME
INVALID_COMMAREA_ADDR
INVALID_COMMAREA_LEN
INVALID_INPUTMSG_LEN
INVALID_KEYWORDS
INVALID_REQUEST_FROM_EXIT
INVALID_TERMINAL_TYPE
NO_TERMINAL
NOT_INITIALIZED
PROGRAM_NOT_AUTHORISED
PROGRAM_NOT_DEFINED
PROGRAM_NOT_ENABLED
PROGRAM_NOT_LOADABLE
REMOTE_PROGRAM
TRANSACTION_ABEND

ABEND_CODE
The four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGXM gate, INITIALIZE_TRANSACTION function

The INITIALIZE_TRANSACTION function of the PGXM gate is used to initialize a transaction, and set up storage for the transaction.

Output Parameters

REASON
The values for the parameter are:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PGXM gate, TERMINATE_TRANSACTION function

The TERMINATE_TRANSACTION function of the PGXM gate is used to terminate a transaction, and clean up the transaction-related storage at task termination.

Output Parameters

REASON
The values for the parameter are:
`INVALID_FUNCTION`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Program manager domain's generic gates

Table 37 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 37. Program manager domain's generic gates

Gate	Trace	Functions	Format
PGDM	PG 0101 PG 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
PGRL	PG 1401 PG 1402	CREATE DISCARD INQUIRE INQUIRE_BY_NAME SET	RLCB
PGST	PG 0F01 PG 0F02	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
PGUE	PG 1001 PG 1002	SET_EXIT_STATUS	APUE

INITIALISE_DOMAIN

There are two phases to initialization of the program manager domain:

1. The DFHPGDM module creates the PG domain anchor block, the table of installed program resource definitions, and the PG Lock. It also adds subpools and gates, determines whether a cold, warm, or emergency start is needed, and waits for the global catalog to be available.
2. For a warm or emergency start, the DFHPGDM module rebuilds the table of installed program resource definitions, and restores the program autoinstall system initialization parameters from the global catalog entries. (It calls the parameter manager to obtain other system initialization parameter values.)

   For a cold start, the DFHPGDM module purges all the information about installed program resource definitions from the global catalog.

QUIESCE_DOMAIN

In quiesce processing, the program manager domain:

1. Sets the PG state to quiescing.

2. Ensures that the statistics domain has gathered the PG statistics by issuing a WAIT_PHASE for STATISTICS_UNAVAILABLE. This also ensures synchronization with the AP domain quiesce activity.
3. Sets the PG state to quiesced.

During quiesce processing, the program manager does not:

- Delete the PG gates. PG functions remain available, but the use of programs after this point does not appear in statistics (DFHSTP issues a PC LINK/ PGLK LINK to DFHWKP after AP domain waits for STATISTICS_UNAVAILABLE).
- Store information about program resource definitions in the global catalog. Information about program resource definitions is written to the catalog only when they are installed or changed.

TERMINATE_DOMAIN

In terminate processing, the program manager domain sets the PG state to terminated, and makes the program manager domain unavailable to EXEC CICS commands.

Modules

Module	Function
DFHPGAI	A kernel subroutine called internally from the Program Manager to support the autoinstall for programs function.
DFHPGAQ	Handles the following requests: INQUIRE_AUTOINSTALL SET_AUTOINSTALL
DFHPGDD	Handles the following requests: DEFINE_PROGRAM DELETE_PROGRAM
DFHPGDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHPGDUF	PG domain offline dump formatting routine
DFHPGEX	Handles the following requests: INITIALIZE_EXIT TERMINATE_EXIT
DFHPGHM	Handles the following requests: SET_CONDITIONS IGNORE_CONDITIONS INQ_CONDITION SET_AIDS INQ_AID SET_ABEND INQ_ABEND PUSH_HANDLE POP_HANDLE FREE_HANDLE_TABLES CLEAR_LABELS

Module	Function
DFHPGIS	Handles the following requests: INQUIRE_PROGRAM INQUIRE_CURRENT_PROGRAM SET_PROGRAM START_BROWSE_PROGRAM GET_NEXT_PROGRAM END_BROWSE_PROGRAM REFRESH_PROGRAM
DFHPGLD	Handles the following requests: LOAD_EXEC LOAD RELEASE_EXEC RELEASE
DFHPGLE	Handles the following requests: LINK_EXEC
DFHPGLK	Handles the following requests: LINK LINK_PLT
DFHPGLU	Handles the following requests: LINK_URM
DFHPGPG	Handles the following requests: INITIAL_LINK
DFHPGRE	Handles the following requests: PREPARE_RETURN_EXEC
DFHPGRP	Program manager domain recovery program, responsible for recovering program definitions from the global catalog.
DFHPGST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHPGTRI	Interprets PG domain trace entries
DFHPGUE	Handles program manager domain service requests.
DFHPGXE	Handles the following requests: PREPARE_XCTL_EXEC
DFHPGXM	Handles the following requests: INITIALIZE_TRANSACTION TERMINATE_TRANSACTION

Pipeline manager domain (PI)

The pipeline manager domain manages the processing of SOAP messages in a CICS pipeline.

Pipeline manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the PI domain.

PIAT gate, CREATE_CONTEXT function

Creates a WSAT coordination context SOAP header.

Input Parameters

POOL_TOKEN

A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIAT gate, CREATE_CONTEXT_RESP function

Create a null context response, which is returned when a WSAT participant send back its output.

Input Parameters

POOL_TOKEN

A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIAT gate, CREATE_NON_TERMINAL_MSG function

Create a non-terminal SOAP message used in WS-AtomicTransaction two-phase commit protocol processing. Non-terminal messages anticipate a response. They are used to convey the following function requests: Prepare, Commit, Rollback, and Replay.

Input Parameters

NOTIFICATION_TYPE

Values for the parameter are:

COMMIT
PREPARE
ROLLBACK

POOL_TOKEN

A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIAT gate, CREATE_REGISTER_REQUEST function

Create a WSAT registration request SOAP message.

Input Parameters

POOL_TOKEN

A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIAT gate, CREATE_REGISTER_RESP function

Create a WSAT registration response SOAP message.

Input Parameters

POOL_TOKEN

A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIAT gate, CREATE_TERMINAL_MSG function

Create a terminal SOAP message used in WS-AtomicTransaction two-phase commit protocol processing. Terminal messages do not anticipate a response. They are used to convey the following function requests: Prepared, Committed, Aborted, and Readonly.

Input Parameters

NOTIFICATION_TYPE

Values for the parameter are:

ABORTED
COMMITTED
PREPARED
READONLY

POOL_TOKEN

A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIAT gate, PROCESS_CONTEXT function

Process a WS-AtomicTransaction coordination context header.

Input Parameters

POOL_TOKEN

A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIAT gate, PROCESS_CONTEXT_RESP function

Process a context coordination response.

Input Parameters

POOL_TOKEN

A token to the current container pool

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIAT gate, PROCESS_MSG function

Process a WS-AtomicTransaction message. This can be a Register Request, a Register Response, a Non Terminal message, or a Terminal Message.

Input Parameters

POOL_TOKEN

A token to the current container pool, which holds data used to build the header, and where the populated DFHHEADER container is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

ABEND
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NO_CHANNEL
PGCR_GET_ERROR
PGCR_PUT_ERROR
SMGF_ERROR
TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PICC gate, FIND_SIGNATURE function

Determine an operation from its signature

Input Parameters

OUTPUT_DATA

A pointer to the operation in the internal COMMAREA or container model (ICM)

XML_BODY_STRING

The incoming SOAP message

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

HEAP_INIT_FAILURE
INSUFFICIENT_STORAGE
INTERNAL_FAILURE
INVALID_PARSE_STATE
SAXHANDLER_LINK_FAILURE

The following values are returned when RESPONSE is EXCEPTION:

FIXED_ELEMENT_COUNT
HEAP_ALLOCATE_FAILURE
HEAP_RELEASE_FAILURE
ICM_ENTRY_NOT_FOUND
INQUIRE_CHANNEL_FAILED
OUTPUT_BUFFER_OVERFLOW
PUT_CONTAINER_FAILED
SOAP_FAULT

The following values are returned when RESPONSE is EXCEPTION:

COMMAREA_LENGTH
INVALID_FUNCTION
INVALID_ICM_TYPE
INVALID_INPUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PICC gate, HANDLE_PARSE_EVENT function

Handle an XML parse event when located by the PL/I SAX parser

Input Parameters

EVENT_TOKEN
A pointer to the event token provided by the XML parser.

EVENT_TOKEN_LENGTH
The length of the event token.

EVENT_TYPE
A BIN(31) value indicating what event has been signaled by the parser.

HANDLER_WORK_TOKEN
A pointer to the DFHPICC work area.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
HEAP_INIT_FAILURE
INSUFFICIENT_STORAGE
INTERNAL_FAILURE
INVALID_PARSE_STATE
SAXHANDLER_LINK_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:

```
FIXED_ELEMENT_COUNT
HEAP_ALLOCATE_FAILURE
HEAP_RELEASE_FAILURE
ICM_ENTRY_NOT_FOUND
INQUIRE_CHANNEL_FAILED
OUTPUT_BUFFER_OVERFLOW
PUT_CONTAINER_FAILED
SOAP_FAULT
```

The following values are returned when RESPONSE is INVALID:

```
COMMAREA_LENGTH
INVALID_FUNCTION
INVALID_INPUT
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PICC gate, PERFORM_XML_PARSE function

Parse a SOAP body and convert the data elements into a COMMAREA format.

Input Parameters

ICM_ADDRESS
The address of the internal COMMAREA or container model (ICM) which is to be used for the SOAP to COMMAREA conversion.

OUTPUT_DATA
A pointer to, and length of, the COMMAREA into which the SOAP body has been mapped.

XML_BODY_STRING
A pointer to the incoming SOAP body.

CHANNEL_NAME
The name of the channel which contains the SOAP body.
XML_HEADER_NS
Optional Parameter

A pointer to the XML namespace information for the SOAP body.
XML_OPERATION
Optional Parameter

The operation name for which the SOAP body is intended.

Output Parameters
REASON
The following values are returned when RESPONSE is DISASTER:
```
HEAP_INIT_FAILURE
INSUFFICIENT_STORAGE
INTERNAL_FAILURE
INVALID_PARSE_STATE
SAXHANDLER_LINK_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:
```
FIXED_ELEMENT_COUNT
HEAP_ALLOCATE_FAILURE
HEAP_RELEASE_FAILURE
ICM_ENTRY_NOT_FOUND
INQUIRE_CHANNEL_FAILED
OUTPUT_BUFFER_OVERFLOW
PUT_CONTAINER_FAILED
SOAP_FAULT
```

The following values are returned when RESPONSE is INVALID:
```
COMMAREA_LENGTH
INVALID_FUNCTION
INVALID_INPUT
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIII gate, PARSE_ICM function
Convert an outbound COMMAREA or container into a SOAP body.

Input Parameters
CHANNEL_NAME
Optional parameter

The name of the channel which holds the container with the SOAP body.
INPUT_COMMAREA
The address and length of the COMMAREA or container to convert.
OUTPUT_ICM_ADDRESS
The address of the internal COMMAREA or container model (ICM) that defines how to map the COMMAREA or container to a SOAP body.
OUTPUT_XML
The address of the SOAP body.

Output Parameters
REASON
Values for the parameter are:
```
ABEND
BUFFER_OVERFLOW
```

CONTAINER_GET_FAILURE
FREEMAIN_FAILURE
GETMAIN_FAILURE
HEAP_INIT_FAILURE
ICM_NOT_FOUND
INPUT_ERROR
INSUFFICIENT_STORAGE
INVALID_FORMAT
INVALID_FUNCTION
INVALID_ICM_DATATYPE
MALLOC_FAILURE
NOT_AUTHORIZED
RELEASE_FAILURE
SEVERE_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIIW gate, INVOKE_WEBSERVICE function

This function supports the INVOKE WEBSERVICE API where CICS is acting as Web Service Requester. Depending upon the attributes specified in the WEBSERVICE resource, it calls the Pipeline Manager (DFHPIPM) to start the pipeline, or it links directly to an application program directly.

Input Parameters

CHANNEL
The name of a channel which holds the container in which data is passed to the target WEBSERVICE.

OPERATION
The name of the operation which is to be invoked.

WEBSERVICE
The name of the WEBSERVICE resource.

URI
Optional Parameter

The URI of the target web service. If this parameter is omitted, the WEBSERVICE resource must specify an endpoint or a program.

Output Parameters

REASON
Values for the parameter are:
CHANNEL_NOT_FOUND
CHANNEL_ERROR
CONTAINER_DATATYPE_ERR
CONTAINER_NOT_FOUND
ENDPOINT_NOT_PROVIDED
INVALID_CHANNEL_NAME
INVALID_FUNCTION
INVALID_OPERATION
INVALID_URI
INVALID_WSBIND_FORMAT
OPERATION_NOT_FOUND
PARSE_CONVERSION_ERROR
PARSE_INPUT_ERROR
PIPELINE_MODE_MISMATCH
PIPELINE_NOT_ACTIVE
PIPELINE_NOT_FOUND

PROGRAM_LINK_FAILED
SOAP_FAULT_BUILT
UNHANDLED_PIPELINE_ERROR
VENDOR_LINK_FAILED
WEBSERVICE_NOT_FOUND
WEBSERVICE_NOT_INSERVICE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOAP_FAULT_RESP1
The response that was returned from the SOAP message handler's fault processing in the DFHWS-RESPCODES container.

SOAP_FAULT_RESP2
The reason that was returned from the SOAP message handler's fault processing in the DFHWS-RESPCODES container.

PIJG gate, CONVERT_DATA_TO_JSON function

Generate JSON from passed input data.

Input Parameters

INPUT_DATA
The address and length of the COMMAREA or container to convert.

OUTPUT_ICM_ADDRESS
The address of the internal COMMAREA or container model (ICM) that defines how to map the COMMAREA or container to JSON.

OUTPUT_JSON
The address of the generated JSON.

POOL_TOKEN
A token representing the channel containing the input data containers.

PRIMARY_CCSID
The primary CCSID of the input data.

SECONDARY_CCSID
The secondary CCSID of the input data.

RESOURCE_NAME
The name of the resource associated with the request.

RESOURCE_TYPE
The type of the resource associated with the request.

MODE
Whether the routine is driven whilst processing a request or a response.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BUFFER_OVERFLOW
INPUT_ERROR
CONVERSION_ERROR
CONTAINER_NOT_FOUND
CONTAINER_DATATYPE_ERR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIJG gate, MODIFY_WSBIND_FOR_JSON function

Generate JSON from passed input data.

Input Parameters

OUTPUT_ICM_ADDRESS
The address of the internal COMMAREA or container model (ICM) that defines how to map the COMMAREA or container to JSON.

PRIMARY_CCSID
The primary CCSID of the input data.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIJP gate, PERFORM_JSON_PARSE function

Create application input data from parsed JSON.

Input Parameters

JSON_BODY_STRING
The address and length of the JSON data to be parsed.

JSON_CONTAINER_TOKEN
The container that holds the JSON data to be parsed.

CCSID
The primary CCSID of the input data.

CCSID_2
The secondary CCSID of the input data.

OUTPUT_DATA
The address and length of the COMMAREA data, created by the parser, or of the name of the container in which this data is placed.

ICM_ADDRESS
The address of the internal COMMAREA or container model (ICM) that defines how to map the JSON to a COMMAREA or container structure.

PROCESSING
An indicator of the method that is used to generate the ICM with: top-down or bottom-up.

CHANNEL_NAME
The name of the channel that contains the JSON message.

CONT_POOL_TOKEN
A token that represents the channel that contains the input/output data containers.

RESOURCE_NAME
The name of the resource that is associated with the request.

RESOURCE_TYPE
The type of the resource that is associated with the request.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
OUTPUT_BUFFER_OVERFLOW
PARSING_ERROR
INPUT_ERROR
CONVERSION_ERROR
UNSUPPORTED_CCSID
INCOMPATIBLE_CCSIDS
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIJS gate, PROCESS_JSON_REQUEST function

Process a JSON request.

Input Parameters

CHANNEL_NAME
The CICS channel containing the JSON request.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, BUILD_CONTENT_TYPE function

Builds a Content-Type header value from the media type and selected parameter values.

Input Parameters

ACTION
Optional parameter

A buffer for the value of the **action** parameter for the Content-Type header in the specified CCSID. This value always includes the surrounding quotes.

BOUNDARY
Optional parameter

A buffer for the value of the **boundary** parameter on the Content-Type header in the specified CCSID. This value does not have surrounding quotes.

CCSID
The fullword binary CCSID value. This is used for header value input and output parameters such as CONTENT_ID.

CHARSET
Optional parameter

A buffer for the value of the **charset** parameter on the Content-Type header in the specified CCSID. This value does not have surrounding quotes.

CONTENT_TYPE
A buffer for the Content-Type header value in the specified CCSID.

MEDIA_TYPE
Optional parameter

A buffer for the value of the media-type field for the Content-Type header in the specified CCSID. For example, `multipart/related`.

START
Optional parameter

A buffer for the value of the **start** parameter on the Content-Type header in the specified CCSID. This value does not have surrounding quotes.

START_INFO
Optional parameter

A buffer for the value of the **start-info** parameter on the Content-Type header in the specified CCSID. This value does not have surrounding quotes.

TYPE
Optional parameter

A buffer for the value of the **type** parameter in the Content-Type header in the specified CCSID. This value does not have surrounding quotes.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
OUTPUT_BUFFER_OVERFLOW
CCSID_NOT_SUPPORTED
MIME_HEADER_ERROR
INVALID_CHARACTER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, BUILD_MIME_HEADERS function

Creates MIME headers from selected header values and stores them in a specific headers container.

Input Parameters

CCSID

The fullword binary CCSID value. This is used for header value input and output parameters such as CONTENT_ID.

CHANNEL_NAME

Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

CONTENT_DESCRIPTION

Optional parameter

A buffer for the Content-Description header value in the specified CCSID.

CONTENT_ID

Optional parameter

A buffer for the Content-ID value in the specified CCSID.

CONTENT_TRAN_ENCODING

Optional parameter

A buffer for the Content-Transfer-Encoding header value in the specified CCSID. This is the value specified on the header, without any white space or comments.

CONTENT_TYPE

Optional parameter

A buffer for the Content-Type header value in the specified CCSID.

HEADERS_CONTAINER

The 16-byte name of the headers container in the specified channel. This should be a container of DATATYPE(CHAR) that contains the MIME headers.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
CCSID_NOT_SUPPORTED
CHANNEL_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_CCSID_ERROR
CONTAINER_WRONG_TYPE
CONTAINER_NAME_INVALID
INVALID_CHARACTER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, BUILD_MIME_MESSAGE function

Combines the contents of the headers container and the body container to create a message container.

Input Parameters

The headers container and the message container are accessed using the CCSID 819. The body container is accessed using the CCSID determined from the **charset** parameter on the Content-type header.

BODY_CONTAINER
The 16-byte name of the body container in the specified channel that contains XOP or XML data. This is a container of DATATYPE(CHAR), unless it contains a binary attachment.

CHANNEL_NAME
Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

HEADERS_CONTAINER
The 16-byte name of the headers container in the specified channel. This is a container of DATATYPE(CHAR) that contains the MIME headers.

MESSAGE_CONTAINER
The 16-byte name of the message container in the specified channel. This is a container of DATATYPE(CHAR) that contains the MIME headers and the body of the message.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_CCSID_ERROR
CONTAINER_WRONG_TYPE
CONTAINER_NAME_INVALID
HEADER_SYNTAX_ERROR
MIME_HEADER_ERROR
ENCODING_NOT_SUPPORTED
CHARSET_NOT_SUPPORTED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, BUILD_MULTIPART_RELATED function

Builds a MIME Multipart/Related message from the headers and body of the root document, and the list of binary attachments. The MIME message headers and body replace the root document and headers in the specified containers.

Input Parameters

ATTACHMENTS_CONTAINER
The 16-byte name of the container in the specified channel that contains the binary attachments list.

BODY_CONTAINER
The 16-byte name of the body container in the specified channel that contains XOP or XML data. This should be a container of DATATYPE(CHAR).

CHANNEL_NAME
Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

HEADERS_CONTAINER
The 16-byte name of the headers container in the specified channel. This should be a container of DATATYPE(CHAR) that contains the MIME headers.

Output Parameters

ATTACHMENTS_COUNT
Optional parameter

The number of <xop:Include> elements that were processed. If the number is 0, the original body container does not include any XOP elements and has not been modified.

REASON
The following values are returned when RESPONSE is EXCEPTION:
- CHANNEL_NOT_FOUND
- CONTAINER_NOT_FOUND
- CONTAINER_CCSID_ERROR
- CONTAINER_WRONG_TYPE
- HEADER_SYNTAX_ERROR
- MIME_HEADER_ERROR
- ENCODING_NOT_SUPPORTED
- CHARSET_NOT_SUPPORTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, CONVERT_CID_TO_CONTENT_ID function

Converts a content-ID in the CID URI format cid:addr-spec to the MIME format <addr-spec>.

Input Parameters

CCSID
The fullword binary CCSID value. This is used for header value input and output parameters such as CONTENT_ID.

CID
A buffer for the CID URI in the specified CCSID. This should be in the format cid:addr-spec.

CONTENT_ID
A buffer for the Content-ID in the specified CCSID. The value should be in the format <addr-spec>.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
- OUTPUT_BUFFER_OVERFLOW
- CCSID_NOT_SUPPORTED
- INVALID_CHARACTER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, CONVERT_CONTENT_ID_TO_CID function

Converts a content-ID in the MIME format <addr-spec> to the CID URI format cid:addr-spec.

Input Parameters

CCSID
The fullword binary CCSID value. This is used for header value input and output parameters such as CONTENT_ID.

CID
A buffer for the CID URI in the specified CCSID. This should be in the format `cid:addr-spec`.

CONTENT_ID
A buffer for the Content-ID in the specified CCSID. The value should be in the format `<addr-spec>`.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
OUTPUT_BUFFER_OVERFLOW
CCSID_NOT_SUPPORTED
INVALID_CHARACTER
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, DELETE_ATTACHMENTS function

Deletes any header and body containers for binary attachments that are listed in the attachments container, and then deletes the attachments container itself.

Input Parameters

ATTACHMENTS_CONTAINER
The 16-byte name of the container in the specified channel that contains the binary attachments list.

CHANNEL_NAME
Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_NOT_FOUND
CONTAINER_WRONG_TYPE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, GENERATE_CONTENT_ID function

Generates a unique content ID value, consisting of a locally unique value based on a timestamp and a supplied domain. The result can be obtained in both content-ID format, `<addr-spec>`, and in CID format, `cid:addr-spec`.

Input Parameters

CCSID
The fullword binary CCSID value. This is used for header value input and output parameters such as CONTENT_ID.

CHANNEL_NAME
Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

CID

Optional parameter

A buffer for the CID URI in the specified CCSID. This should be in the format `cid:addr-spec`.

CID_DOMAIN_CHARACTER

The 16-byte name of the container in the specified channel that contains the domain name. This string is used as the last part of a content-ID to identify the sysplex within which the locally unique value applies.

CONTENT_ID

Optional parameter

A buffer for the Content-ID in the specified CCSID. The value should be in the format `<addr-spec>`.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
OUTPUT_BUFFER_OVERFLOW
CCSID_NOT_SUPPORTED
INVALID_CHARACTER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, GET_ATTACHMENT function

Retrieves the container names for the headers and body of the binary attachment with the specified Content-ID or CID.

Input Parameters

ATTACHMENTS_CONTAINER

The 16-byte name of the container in the specified channel that contains the binary attachments list.

CCSID

The fullword binary CCSID value. This is used for header value input and output parameters such as CONTENT_ID.

CHANNEL_NAME

Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

CID

A buffer for the CID URI in the specified CCSID. This should be in the format `cid:addr-spec`. Either CID or CONTENT_ID can be used as input.

CONTENT_ID

A buffer for the Content-ID in the specified CCSID. The value should be in the format `<addr-spec>`. Either CID or CONTENT_ID can be used as input.

Output Parameters

BODY_CONTAINER

The 16-byte name of the body container in the specified channel. This is a container of DATATYPE(BIT), as it contains a binary attachment.

HEADERS_CONTAINER

The 16-byte name of the headers container in the specified channel. This is a container of DATATYPE(CHAR) that contains the MIME headers.

REASON
The following values are returned when RESPONSE is EXCEPTION:
CHANNEL_NOT_FOUND
CCSID_NOT_SUPPORTED
CONTAINER_NOT_FOUND
CONTAINER_CCSID_ERROR
CONTAINER_WRONG_TYPE
ATTACHMENT_NOT_FOUND
INVALID_CHARACTER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, PARSE_CONTENT_TYPE function

Parses the Content-Type header and picks out selected fields as requested, including the media type and specific parameters. The media type field and **charset** parameter are converted to lower case if necessary.

Input Parameters

ACTION
Optional parameter

A buffer for the value of the **action** parameter for the Content-Type header in the specified CCSID. This value always includes the surrounding quotes.

BOUNDARY
Optional parameter

A buffer for the value of the **boundary** parameter on the Content-Type header in the specified CCSID. This value does not have surrounding quotes.

CCSID
The fullword binary CCSID value. This is used for header value input and output parameters such as CONTENT_ID.

CHARSET
Optional parameter

A buffer for the value of the **charset** parameter on the Content-Type header in the specified CCSID. This value does not have surrounding quotes.

CONTENT_TYPE
A buffer for the Content-Type header value in the specified CCSID.

MEDIA_TYPE
Optional parameter

A buffer for the value of the media type field for the Content-Type header in the specified CCSID. For example, `multipart/related`.

START
Optional parameter

A buffer for the value of the **start** parameter on the Content-Type header in the specified CCSID. This value does not have surrounding quotes.

START_INFO
Optional parameter

A buffer for the value of the **start-info** parameter on the Content-Type header in the specified CCSID. This value does not have surrounding quotes.

TYPE
Optional parameter

A buffer for the value of the **type** parameter in the Content-Type header in the specified CCSID. This value does not have surrounding quotes.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
OUTPUT_BUFFER_OVERFLOW
CCSID_NOT_SUPPORTED
MIME_HEADER_ERROR
INVALID_CHARACTER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, PARSE_MIME_HEADERS function

Retrieves selected MIME header values from a MIME headers container or a MIME message container. The results are edited into a standard format, removing excess white space and comments, and converting case-insensitive keywords to lower case.

Input Parameters

CCSID

The fullword binary CCSID value. This is used for header value input and output parameters such as CONTENT_ID.

CHANNEL_NAME

Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

CONTENT_DESCRIPTION

Optional parameter

A buffer for the Content-Description header value in the specified CCSID.

CONTENT_ID

Optional parameter

A buffer for the Content-ID header value in the specified CCSID.

CONTENT_TRAN_ENCODING

Optional parameter

A buffer for the Content-Transfer-Encoding header value in the specified CCSID. This is the value specified on the header, without any white space or comments.

CONTENT_TYPE

Optional parameter

A buffer for the Content-Type header value in the specified CCSID.

HEADERS_CONTAINER

The 16-byte name of the headers container in the specified channel. This should be a container of DATATYPE(CHAR) that contains the MIME headers.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
OUTPUT_BUFFER_OVERFLOW
CCSID_NOT_SUPPORTED
CHANNEL_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_CCSID_ERROR
CONTAINER_WRONG_TYPE
HEADER_SYNTAX_ERROR
MIME_HEADER_ERROR
```

INVALID_CHARACTER
ENCODING_NOT_SUPPORTED
CHARSET_NOT_SUPPORTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, PARSE_MIME_MESSAGE function

Splits the message into headers, which are stored in a headers container, and a body which is stored in a body container.

Input Parameters

The message container and headers container are accessed using CCSID 819. The body container is accessed using the CCSID determined from the **charset** parameter on the Content-type header.

BODY_CONTAINER
The 16-byte name of the body container in the specified channel that is created to contain XOP or XML data. This is a container of DATATYPE(CHAR).

CHANNEL_NAME
Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

HEADERS_CONTAINER
The 16-byte name of the headers container in the specified channel that is created to contain the MIME headers. This is a container of DATATYPE(CHAR).

MESSAGE_CONTAINER
The 16-byte name of the message container in the specified channel. This should be a container of DATATYPE(CHAR) that contains the MIME headers and the body of the message.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CHANNEL_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_CCSID_ERROR
CONTAINER_WRONG_TYPE
CONTAINER_NAME_INVALID
HEADER_SYNTAX_ERROR
MIME_HEADER_ERROR
ENCODING_NOT_SUPPORTED
CHARSET_NOT_SUPPORTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, PARSE_MULTIPART_RELATED function

Parses a MIME MultipartRelated message, splitting out the root document and the binary attachments. The root document and headers replace the contents of the original message in the container, and any binary attachments are stored in separate containers. The list of attachments is stored in the attachments list container.

Input Parameters

ATTACHMENTS_CONTAINER
The 16-byte name of the container in the specified channel that contains the binary attachments list.

BODY_CONTAINER
The 16-byte name of the body container in the specified channel that contains XOP or XML data. This should be a container of DATATYPE(CHAR).

CHANNEL_NAME
Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

HEADERS_CONTAINER
The 16-byte name of the headers container in the specified channel. This should be a container of DATATYPE(CHAR) that contains the MIME headers.

Output Parameters

ATTACHMENTS_COUNT
Optional parameter

The number of `<xop:Include>` elements that were processed. If the number is 0, the original body container does not include any XOP elements and has not been modified.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NOT_MULTIPART_RELATED
CHANNEL_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_CCSID_ERROR
CONTAINER_WRONG_TYPE
CONTAINER_NAME_INVALID
HEADER_SYNTAX_ERROR
MIME_HEADER_ERROR
MIME_BOUNDARY_ERROR
ROOT_PART_NOT_FOUND
ENCODING_NOT_SUPPORTED
CHARSET_NOT_SUPPORTED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIMM gate, PUT_ATTACHMENT function

Adds the names of the headers and body containers for the binary attachment with the given content-ID or CID to the attachments container.

Input Parameters

ATTACHMENTS_CONTAINER
The 16-byte name of the container in the specified channel that contains the binary attachments list.

BODY_CONTAINER
The 16-byte name of the body container in the specified channel. This is a container of DATATYPE(BIT), as it always contains a binary attachment.

CCSID
The fullword binary CCSID value. This is used for header value input and output parameters such as CONTENT_ID.

CHANNEL_NAME
Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

CID
A buffer for the CID URI in the specified CCSID. This should be in the format `cid:addr-spec`. Either CID or CONTENT_ID can be used as input.

CONTENT_ID
A buffer for the Content-ID in the specified CCSID. The value should be in the format `<addr-spec>`. Either CID or CONTENT_ID can be used as input.

HEADERS_CONTAINER
The 16-byte name of the headers container in the specified channel. This should be a container of DATATYPE(CHAR) that contains the MIME headers.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_NOT_FOUND
CCSID_NOT_SUPPORTED
CONTAINER_NAME_INVALID
DUPLICATE_ATTACHMENT
INVALID_CHARACTER
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPL gate, ADD_PIPELINE function

Add a PIPELINE definition to the system.

Input Parameters

CONFIGFILE
The fully qualified name of the XML pipeline configuration file on z/OS UNIX.

PIPELINE
The name of the PIPELINE.

SHELF
The fully qualified name of a directory (or shelf) primarily for WSBIND and WSDL files.

STATUS
The initial state of the PIPELINE.

Values for the parameter are:

```
DISABLED
ENABLED
```

WSDIR
Optional Parameter

The fully qualified name of the WSBIND directory on z/OS UNIX.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CATALOG_ERROR
DIRECTORY_ERROR
INVALID_HFSNAME
INVALID_NAME
INVALID_SHELF
INVALID_STATUS
INVALID_WSDIR
```

NOT_AUTHORIZED
NOT_DISABLED
WSDIR_INACCESIBLE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPL gate, COMPLETE_PIPELINE function

Complete the installation of a PIPELINE. PIPELINEs are installed in two phases: this is the second, called after CICS initialization is complete. This function reads data from the files in z/OS UNIX and builds the internal control blocks.

Input Parameters

PIPELINE
The name of the PIPELINE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CATALOG_ERROR
DIRECTORY_ERROR
INVALID_HFSNAME
INVALID_NAME
INVALID_SHELF
INVALID_STATUS
INVALID_WSDIR
NOT_AUTHORIZED
NOT_DISABLED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPL gate, DISCARD_PIPELINE function

Discard a PIPELINE.

Input Parameters

PIPELINE
The name of the PIPELINE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CATALOG_ERROR
DISCARD_IN_PROGRESS
INVALID_BROWSE_TOKEN
NOT_AUTHORIZED
NOT_DISABLED
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPL gate, END_BROWSE_PIPELINE function

End the browse operation on the PIPELINE resources that are installed in the system.

Input Parameters

BROWSETOKEN

A token that represents the browse operation on subsequent GET_NEXT_PIPLINE and END_BROWSE requests.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ABEND
INVALID_BROWSE_TOKEN
LOOP
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPL gate, ESTABLISH_PIPELINE function

Check that a PIPELINE is in a state in which it can be used, and increment its use count.

Input Parameters

PIPELINE

The name of the PIPELINE.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
CATALOG_ERROR
INVALID_STATUS
NOT_AUTHORIZED
NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPL gate, GET_NEXT_PIPELINE function

During a browse operation, extract information about the next PIPELINE.

Input Parameters

BROWSETOKEN

The browse token that was returned by the START_BROWSE_PIPELINE function.

CONFIGFILE_BUFF

Optional Parameter

A buffer in which the fully qualified name of the XML pipeline configuration file on z/OS UNIX is returned.

RESET

Optional Parameter

A parameter indicating whether the statistics for the PIPELINE are to be reset.

Values for the parameter are:

```
NO
YES
```

SHELF_BUFF

Optional Parameter

A buffer in which the fully qualified name of the directory (or shelf) for WSBIND and WSDL files is returned.

WSDIR_BUFF
Optional Parameter

A buffer in which the fully qualified name of the WSBIND directory on z/OS UNIX is returned.

Output Parameters

PIPELINE
The name of the PIPELINE.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
ABEND
BROWSE_END
INVALID_BROWSE_TOKEN
LOCK_ERROR
LOOP
PARMS_STORAGE_ERROR
SETUP_ERROR
STORAGE_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATUS
Optional Parameter

The current status of the PIPELINE.

Values for the parameter are:

```
DISABLING
DISABLED
DISCARDING
ENABLED
ENABLING
```

TOTAL_USE_COUNT
Optional Parameter

The current use count of the PIPELINE.

PIPL gate, INQUIRE_PIPELINE function

Inquire on the attributes, state and associated resources of a PIPELINE.

Input Parameters

PIPELINE
The name of the PIPELINE.

CONFIGFILE_BUFF
Optional Parameter

A buffer in which the fully qualified name of the XML pipeline configuration file on z/OS UNIX is returned.

DERIVED_SHELF_BUFF
Optional Parameter

A buffer in which the fully qualified name of the z/OS UNIX file which contains the WSDL for the PIPELINE is returned.

SHELF_BUFF
Optional Parameter

A buffer in which the fully qualified name of the directory (or shelf) for WSBIND and WSDL files is returned.

WSDIR_BUFF

Optional Parameter

A buffer in which the fully qualified name of the WSBIND directory on z/OS UNIXis returned.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
NOT_AUTHORIZED
NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MODE

Optional Parameter

The MODE of the PIPELINE.

Values for the parameter are:

```
PROVIDER
REQUESTER
UNKNOWN
```

PIPELINE_TOKEN

Optional Parameter

A token which can be used by other parts of the domain to refer to the PIPELINE.

STATUS

Optional Parameter

The current status of the PIPELINE.

Values for the parameter are:

```
DISABLING
DISABLED
DISCARDING
ENABLED
ENABLING
```

TOTAL_USE_COUNT

Optional Parameter

The current use count of the PIPELINE.

PIPL gate, PERFORM_PIPELINE function

Perform the specified action on a PIPELINE.

Input Parameters

ACTION

The only supported action is SCAN. The PIPELINE is scanned for WSBIND files which are then installed.

Values for the parameter are:

```
SCAN
```

PIPELINE

The name of the PIPELINE.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ABEND
DUPLICATE
INVALID_ACTION
INVALID_STATUS
LOOP
NOT_AUTHORIZED
NOT_FOUND
PIPELINE_SCAN_ERROR
SCAN_ALREADY_IN_PROGRESS
WSDIR_INACCESSIBLE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPL gate, RELINQUISH_PIPELINE function

Relinquish the use of a PIPELINE. The use count is decremented, and if it is then zero, and the PIPELINE's state is DISABLING, the status changes to DISABLED.

Input Parameters

PIPELINE

The name of the PIPELINE.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
CATALOG_ERROR
NOT_AUTHORIZED
NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPL gate, RESOLVE_PIPELINE function

For each PIPELINE, start a transaction to complete PIPELINE installation. The function is used at the end of domain initialization.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ABEND
LOOP
SETUP_ERROR
STORAGE_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPL gate, SET_PIPELINE function

Set a PIPELINE to DISABLED or ENABLED state.

Input Parameters

PIPELINE

The name of the PIPELINE.

STATUS
The state to be set.

Values for the parameter are:
DISABLED
ENABLED

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_STATE
NOT_AUTHORIZED
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPL gate, START_BROWSE_PIPELINE function

Start browsing the installed PIPELINE resources.

Input Parameters

PIPELINE
Optional Parameter

The name of the PIPELINE at which the browse is to begin.

Output Parameters

BROWSETOKEN
A token that identifies the browse operation to subsequent GET_NEXT_PIPELINE and END_BROWSE reqeusts.

REASON
Values for the parameter are:
ABEND
INVALID_PIPELINE
LOCK_ERROR
LOOP
SETUP_ERROR
STORAGE_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPM gate, INVOKE_PROGRAM function

Invoke a PIPELINE's application programs. The function can change the transaction's context, and the request can be routed to another region.

Input Parameters

CHANNEL
The channel to be passed to the target program.

PROGRAM
The program to be invoked.

APPLID
Optional Parameter

The APPLID to be used for the execution of the application program.

RS_PUBLIC_ID
Optional Parameter

The request stream public identifier to be associated with the transaction.

TRANSID

Optional Parameter

The transaction identifier to be used to execute the application program.

USERID

Optional Parameter

The user ID to be used for the execution of the application program.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_FAILURE
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_ERROR
CONTEXT_SWITCH_FAILED
NO_CHANNEL
PIPELINE_MODE_MISMATCH
PIPELINE_NOT_ACTIVE
PIPELINE_NOT_FOUND
RZ_CREATE_FAILURE
RZ_TRANSPORT_ERROR
TARGET_PROGRAM_UNAVAILABLE
UNHANDLED_NODE_FAILURE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is PURGED:

```
TASK_CANCELLED
TIMED_OUT
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPM gate, INVOKE_STUB function

Invoke an application program remotely.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_FAILURE
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_ERROR
CONTEXT_SWITCH_FAILED
NO_CHANNEL
PIPELINE_MODE_MISMATCH
PIPELINE_NOT_ACTIVE
PIPELINE_NOT_FOUND
RZ_CREATE_FAILURE
RZ_TRANSPORT_ERROR
```

TARGET_PROGRAM_UNAVAILABLE
UNHANDLED_NODE_FAILURE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPM gate, START_PIPELINE function

Start a requester or provider pipeline.

Input Parameters

MODE
Parameter indicating whether the pipeline is to be started for a service requester or for a service provider.

Values for the parameter are:
PROVIDER
REQUESTER

PIPELINE
The name of the PIPELINE resource.

CHANNEL
Optional Parameter

The name of a channel holding containers to be passed to the pipeline.

TRANSPORT_NAME
Optional Parameter

Depending upon the value of the TRANSPORT_TYPE parameter, the name of a TCPIPSERVICE or an MQ queue to be passed to the pipeline.

TRANSPORT_TYPE
Optional Parameter

Parameter indicating the type of transport.

Values for the parameter are:
HTTP
MQ

WEBSERVICE
Optional Parameter

The name of the WEBSERVICE to be invoked for this pipeline.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:
CHANNEL_ERROR
CONTEXT_SWITCH_FAILED
NO_CHANNEL
PIPELINE_MODE_MISMATCH

PIPELINE_NOT_ACTIVE
PIPELINE_NOT_FOUND
RZ_CREATE_FAILURE
RZ_TRANSPORT_ERROR
TARGET_PROGRAM_UNAVAILABLE
UNHANDLED_NODE_FAILURE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIRE gate, PERFORM_RESYNC function

Resynchronize any WS-AtomicTransaction units of work that are indoubt, following a restart of CICS.

Input Parameters

None.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ALREADY_IN_RESYNC

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PISC gate, DYN_CREATE_WEBSERVICE function

This function dynamically creates a WEBSERVICE resource via a PIPELINE scan.

Input Parameters

PIPELINE
The name of the PIPELINE resource that owns the WEBSERVICE.

WSBIND
The fully qualified location of the web service binding file in the pickup directory in the z/OS UNIX file system.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CREATE_FAILED
DISCARD_FAILED
INQUIRE_FAILED
INQUIRE_HFS_FAILED
NAME_CLASH
NO_UPDATE_NEEDED
UPDATE_PENDING
WSDL_NAME_TOO_LONG

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PISC gate, UPDATE_WEBSERVICE function

This function completes the updating of a WEBSERVICE resource. It is invoked when the use count for a WEBSERVICE which is in UPDATING state reaches zero.

Input Parameters

WEBSERVICE
The name of the WEBSERVICE whose update is to be completed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CREATE_FAILED
DISCARD_FAILED
INQUIRE_FAILED
INQUIRE_HFS_FAILED
NAME_CLASH
NO_UPDATE_NEEDED
UPDATE_PENDING
WSDL_NAME_TOO_LONG
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PISF gate, SOAPFAULT_ADD function

Add extra data to a SOAP fault created by the SOAPFAULT_CREATE function.

Input Parameters

FAULT_STRING
The description of the fault in a readable form.

SUBCODE_STRING
The value to put in the `<subcode>` element of a SOAP fault.

CCSID
Optional Parameter

The CCSID of the input.

NATLANG
Optional Parameter

The `xml:lang` value for the FAULT_STRING

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CCSID_CONVERSION_ERROR
CCSID_INVALID
CCSID_PARTIAL_CONVERSION
CCSID_UNSUPPORTED
INVALID_CODE
INVALID_REQUEST
NO_FAULT
SEVERE_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PISF gate, SOAPFAULT_CREATE function

Create a SOAP fault in an internal format.

Input Parameters

FAULT_STRING
The description of the fault in a readable form.

FAULTCODE
The standard SOAP fault code to use

FAULTCODE_STRING
The value to use for the `<faultcode>` element instead of a standard one.

CCSID
Optional Parameter

The CCSID of the input.

DETAIL
Optional Parameter

XML containing detailed fault data.

FAULT_ACTOR
Optional Parameter

The value to put in the `<faultactor>` element.

NATLANG
Optional Parameter

The `xml:lang` value for the FAULT_STRING parameter.

ROLE
Optional Parameter

The value to put in the `<role>` element.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CCSID_CONVERSION_ERROR
CCSID_INVALID
CCSID_PARTIAL_CONVERSION
CCSID_UNSUPPORTED
INVALID_CODE
INVALID_REQUEST
SEVERE_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PISF gate, SOAPFAULT_DELETE function

Delete the internal form of a SOAP fault.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NO_FAULT
NOT_FOUND
SEVERE_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PISN gate, SOAP_11 function

Start a message handler to process SOAP 1.1 messages.

Output Parameters

SOAPFAULT
indicates whether a SOAP fault has been built.

Values for the parameter are:
NONE
FAULT_BUILT

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
BAD_FAULT
SEVERE_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PISN gate, SOAP_12 function

Start a message handler to process SOAP 1.2 messages.

Output Parameters

SOAPFAULT
indicates whether a SOAP fault has been built.

Values for the parameter are:
NONE
FAULT_BUILT

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
BAD_FAULT
SEVERE_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PITC gate, ISSUE function

Sends a request to the Security Token Service to issue a username token in exchange for a security token from the WS-Security message header.

Input parameters

DESTINATION_URI_BLOCK
The URI of the Security Token Service endpoint on the network.

SERVICE_URI_BLOCK
The URI of the web service that the Security Token should issue a token for to CICS. This URI is taken from the `appliesTo` field.

TRUST_LEVEL
Optional parameter.

The level of WS-Trust that CICS supports.

SECURITY_TOKEN_BLOCK
Optional parameter.

The security token that the Security Token Service should exchange.

AUTHTOKEN_TYPE_BLOCK
The URI and localname of the token type that should be returned by the Security Token Service.

RETURNED_SECTOK_BUFF
A buffer for the token that is returned by the Security Token Service.

RESPONSE_TOKEN
The token that is issued by the Security Token Service.

Output parameters

PASSWORD
Optional parameter.

The password that is returned by the Security Token Service.

USERNAME
Optional parameter.

The user name that is returned by the Security Token Service.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
BUFFER_TOO_SMALL
CHANNEL_ERROR
CONTAINER_ERROR
INVALID_URI
ENDPOINT_NOT_PROVIDED
SOAP_FAULT_BUILT
UNHANDLED_PIPELINE_ERROR
TIMED_OUT
NO_TRUST_REPLY
TRUST_PARSE_FAILED
TRUST_FAULT
INVALID_TRUST_REPLY
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PITC gate, VALIDATE function

Sends a request to the Security Token Service to validate a security token from the WS-Security message header.

Input parameters

DESTINATION_URI_BLOCK
The URI of the Security Token Service endpoint on the network.

TRUST_LEVEL
The level of WS-Trust that is supported in CICS.

SECURITY_TOKEN_BLOCK
The security token that should be validated by the Security Token Service.

RETURNED_SECTOK_BUFF
A buffer for the validation response that is returned by the Security Token Service.

RESPONSE_TOKEN
A unique reference that identifies the request to CICS.

Output parameters

STATUS

The status of the security token that was passed to the Security Token Service for verification. Values are:

TRUST_VALID
The Security Token Service has confirmed that the security token is valid.

TRUST_INVALID
The Security Token Service has confirmed that the security token is invalid.

TRUST_UNKNOWN
The Security Token Service was unable to verify the security token.

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
BUFFER_TOO_SMALL
CHANNEL_ERROR
CONTAINER_ERROR
INVALID_URI
ENDPOINT_NOT_PROVIDED
SOAP_FAULT_BUILT
UNHANDLED_PIPELINE_ERROR
TIMED_OUT
NO_TRUST_REPLY
TRUST_PARSE_FAILED
TRUST_FAULT
INVALID_TRUST_REPLY
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PITC gate, GET_RESPONSE function

Retrieves the response message from the Security Token Service.

Input parameters

RESPONSE_TOKEN

The security token that is issued by the Security Token Service.

RETURNED_SECTOK_BUFF

A buffer for the security token that is issued by the Security Token Service.

Output parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
NOT_FOUND
BUFFER_TOO_SMALL
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PITC gate, TRUST_CLIENT function

Decides what security handler processing should take place in the pipeline.

Input parameters

WSSE_CONFIG

A pointer to the pipeline configuration file details that are stored in memory.

WSSE_PROGRAM
The name of the security handler program.
CHANNEL_TOKEN
The token for the current channel that is being used by the pipeline.
POOL_TOKEN
The token that identifies the pool of containers that is being used by the current channel in the pipeline.
MODE
The mode of the pipeline, either a service requester or service provider.
DIRECTION
The direction for the message, either a request message or response message.

Output parameters
REASON
The following values are returned when RESPONSE is EXCEPTION:
```
TRUST_FAULT
INVALID_SECURITY_CONTENT
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PITG gate, SEND_REQUEST function
Send a web service request. This is a generic format for the PITH gate (HTML transport), PITQ gate (WebSphere® MQ transport), and PITS gate (CICS transport).

Input Parameters

None

Output Parameters
REASON
The following values are returned when RESPONSE is EXCEPTION:
```
INVALID_CODEPAGE
SOCKET_ERROR
UNKNOWN_HOST
```
The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```
The following values are returned when RESPONSE is DISASTER:
```
ABEND
MQ_FAILURE
```
The following values are returned when RESPONSE is EXCEPTION:
```
INSUFFICIENT_STORAGE
INVALID_PARAMETER
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PITG gate, SEND_RESPONSE function
Send a web service response. This is a generic format for the PITH gate (HTML transport), PITQ gate (IBM MQ transport), and PITS gate (CICS transport).

Input Parameters

None

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_CODEPAGE
SOCKET_ERROR
UNKNOWN_HOST
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
MQ_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:

```
INSUFFICIENT_STORAGE
INVALID_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PITG gate, CONVERSE function

Send a web service request and receive the reply. This is a generic format for the PITH gate (HTML transport), PITQ gate (IBM MQ transport), and PITS gate (CICS transport).

Input Parameters

None

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_CODEPAGE
SOCKET_ERROR
UNKNOWN_HOST
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
MQ_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:

```
INSUFFICIENT_STORAGE
INVALID_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PITG gate, RECEIVE_REQUEST function

Receive a web service request. This is a generic format for the PITH gate (HTML transport), PITQ gate (IBM MQ transport), and PITS gate (CICS transport).

Input Parameters

None

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
CODEPAGE_NOT_FOUND
CONNECTION_CLOSED
SOCKET_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
MQ_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:

```
INSUFFICIENT_STORAGE
INVALID_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PITG gate, SEND_ERROR_RESPONSE function

Send a web service error response. This is a generic format for the PITH gate (HTML transport), PITQ gate (IBM MQ transport), and PITS gate (CICS transport).

Input Parameters

None

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
MQ_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:

```
INSUFFICIENT_STORAGE
INVALID_PARAMETER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PITL gate, PROCESS_SOAP_REQUEST function

Process a SOAP body received on a SOAP pipeline

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ABEND
APP_FAULT
```

CONV_FROM_SOAP_FAILED
CONV_TO_SOAP_FAILED
INBOUND_VALIDATION_FAILED
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NOT_AUTHORIZED
OPERATION_NOT_FOUND
OUTBOUND_VALIDATION_FAILED
SEVERE_ERROR
SOAP_BODY_CONTAINER_FAULT
TARGET_ABENDED
TARGET_LINK_FAILED
VENDOR_LINK_FAILED
WSBIND_FORMAT_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIWR gate, CREATE_WEBSERVICE function

Create a new WEBSERVICE resource.

Input Parameters

PIPELINE
The pipeline which will own the WEBSERVICE.

WEBSERVICE
The name of the WEBSERVICE.

WSBIND_BUF
The location of the web service binding file in the z/OS UNIX file system.

SCAN_MODE
Optional Parameter

Indicates whether the WEBSERVICE is being scanned in or not.

Values for the parameter are:
NO
YES

VALIDATION
Optional Parameter

Indicates whether validation is enabled for the WEBSERVICE.

Values for the parameter are:
NO
YES

WARM_RESTART
Optional Parameter

Indicates whether the WEBSERVICE is to be recovered from the catalog during a warm restart.

Values for the parameter are:
NO
YES

WSDLFILE_BUF
Optional Parameter

The location of the optional web service description (WSDL) file in the z/OS UNIX file system.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ABEND
DIRECTORY_ERROR
INSUFFICIENT_STORAGE
LOCK_FAILURE
PIPELINE_ERROR
PIPELINE_NON_EXISTANT
SEVERE_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIWR gate, DECREMENT_USE_COUNT function

Decrement the current use count for a WEBSERVICE. When it reaches 0 and if the WEBSERVICE is updating or discarding then the completion of the update or discard operation will be triggered.

Input Parameters

WEBSERVICE

The name of the WEBSERVICE.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ABEND
SEVERE_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIWR gate, DISCARD_WEBSERVICE function

This function discards a WEBSERVICE resource.

Input Parameters

WEBSERVICE

The name of the WEBSERVICE.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ABEND
NOT_AUTHORIZED
SEVERE_ERROR
WEBSERVICE_IN_USE
WEBSERVICE_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIWR gate, END_BROWSE_WEBSERVICE function

This function ends a browse operation for WEBSERVICE resources.

Input Parameters

BROWSE_TOKEN
The browse token for the browse operation.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIWR gate, GET_NEXT_WEBSERVICE function

Get the next WEBSERVICE resource during a browse operation.

Input Parameters

BROWSE_TOKEN
The browse token for the browse operation.

BINDING_BUF
Optional Parameter

A buffer in which the WSDL binding value is returned.

ENDPOINT_BUF
Optional Parameter

A buffer in which the end point URI is returned.

RESET
Optional Parameter

A flag that indicates if the use count is to be reset to zero.

Values for the parameter are:
NO
YES

WSBIND_BUF
Optional Parameter

A buffer in which the location of the Webservice binding file in the z/OS UNIX file system is returned.

WSDLFILE_BUF
Optional Parameter

A buffer in which the location of the web service description (WSDL) file in the z/OS UNIX file system is returned.

Output Parameters

DATESTAMP
The date stamp of the web service binding file

LASTMODTIME
The time at which the web service binding file was last changed.

PGMINTERFACE
The type of interface used by the target program

Values for the parameter are:
CHANNEL
COMMAREA

PIPELINE
The pipeline which owns the WEBSERVICE.

PROGRAM
The target program.

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATE
The current state of the WEBSERVICE.

Values for the parameter are:
DISABLED
DISABLING
DISCARDING
INITING
INSERVICE
UNUSABLE
UPDATING

TIMESTAMP
The time stamp of the web service binding file.

URIMAP
The name of the URIMAP that is associated with the WEBSERVICE.

VALIDATION
Indicates whether validation is enabled for the WEBSERVICE.

Values for the parameter are:
NO
YES

WEBSERVICE
The name of the WEBSERVICE.

TOTAL_USE_COUNT
Optional Parameter

The current use count for the WEBSERVICE.

PIWR gate, INCREMENT_USE_COUNT function

Increment the use count for the named WEBSERVICE.

Input Parameters

WEBSERVICE
The name of the WEBSERVICE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
SEVERE_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIWR gate, INITIALISE_WEBSERVICE function

Resolve the z/OS UNIX parts of a WEBSERVICE. The function takes a WEBSERVICE which is in INSTALLING state to either INSERVICE or UNUSABLE state.

Input Parameters

WEBSERVICE

The name of the WEBSERVICE.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ABEND
EYECATCHER_ERROR
FILE_NOT_FOUND
INSUFFICIENT_STORAGE
NOT_AUTHORIZED
PIPELINE_ERROR
PIPELINE_WRONG_MODE
READ_ERROR
SEVERE_ERROR
SHELF_WRITE_ERROR
VERSION_ERROR
WEBSERVICE_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIWR gate, INQUIRE_WEBSERVICE function

Inquire on a WEBSERVICE resource.

Input Parameters

WEBSERVICE

The name of the WEBSERVICE.

BINDING_BUF

Optional Parameter

A buffer in which the WSDL binding value is returned.

ENDPOINT_BUF

Optional Parameter

A buffer in which the endpoint URI is returned.

WSBIND_BUF

Optional Parameter

A buffer in which the location of the web service binding file in z/OS UNIX is returned.

WSDLFILE_BUF

Optional Parameter

A buffer in which the location of the optional web service description (WSDL) file in z/OS UNIX is returned.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
ABEND
NOT_AUTHORIZED
SEVERE_ERROR
WEBSERVICE_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONTAINER
Optional Parameter

The name of the container for the target program's data.

DATESTAMP
Optional Parameter

The date stamp of the web service binding file.

LASTMODTIME
Optional Parameter

The time at which the web service binding file was last changed.

PGMINTERFACE
Optional Parameter

The type of interface used by the target program

Values for the parameter are:
- CHANNEL
- COMMAREA
- NOTAPPLIC

PGMINTERFACE
The type of interface used by the target program

Values for the parameter are:
- CHANNEL
- COMMAREA
- NOTAPPLIC

PIPELINE
Optional Parameter

The pipeline which owns the WEBSERVICE.

PROGRAM
Optional Parameter

The target program.

STATE
Optional Parameter

The current state of the WEBSERVICE.

Values for the parameter are:
- DISABLED
- DISABLING
- DISCARDING
- INITING
- INSERVICE
- UNUSABLE
- UPDATING

TIMESTAMP
Optional Parameter

The time stamp of the web service binding file.

TOTAL_USE_COUNT
Optional Parameter

The total use count for the WEBSERVICE.

URIMAP
Optional Parameter

The name of the URIMAP that is associated with the WEBSERVICE.

VALIDATION
Optional Parameter

Indicates whether validation is enabled for the WEBSERVICE.

Values for the parameter are:
NO
YES

WSADDR
Optional Parameter

The address of the WEBSERVICE control block.

PIWR gate, RESOLVE_ALL_WEBSERVICES function

Resolve all WEBSERVICE resources for a given pipeline that are in INITING state.

Input Parameters

PIPELINE
Optional Parameter

The pipeline for which WEBSERVICE resources are to be resolved.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
SEVERE_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIWR gate, SET_WEBSERVICE function

Change the state of a WEBSERVICE resource.

Input Parameters

VALIDATION
The new validation state for the WEBSERVICE.

Values for the parameter are:
NO
YES

STATE
The current state of the WEBSERVICE.

Values for the parameter are:
DISABLED
DISABLING
DISCARDING
INITING
INSERVICE
UNUSABLE
UPDATING

WEBSERVICE
The name of the WEBSERVICE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND

DUPLICATE
NOT_AUTHORIZED
SEVERE_ERROR
WEBSERVICE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIWR gate, START_BROWSE_WEBSERVICE function

Start a browse operation on WEBSERVICE resources.

Output Parameters

BROWSE_TOKEN
The browse token for the browse operation.

REASON
The following values are returned when RESPONSE is EXCEPTION:
ABEND
BROWSE_END
DIRECTORY_ERROR
DUPLICATE
FILE_NOT_FOUND
FREEMAIN_FAILURE
INSUFFICIENT_STORAGE
INVALID_BROWSE_TOKEN
INVALID_FORMAT
INVALID_FUNCTION
LOCK_FAILURE
LOOP
NO_WEBS_INSTALLED
NOT_AUTHORIZED
PIPELINE_ERROR
PIPELINE_NON_EXISTANT
PIPELINE_WRONG_MODE
READ_ERROR
SEVERE_ERROR
SHELF_WRITE_ERROR
WEBSERVICE_IN_USE
WEBSERVICE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIXI gate, PARSE_XOP function

Converts the XOP message back to standard XML, by replacing any `xop:Include` elements with the base64binary encoded data from the corresponding binary attachment. If there are no XOP elements, nothing is changed.

Input Parameters

ATTACHMENTS_CONTAINER
The 16-byte name of the container in the specified channel that contains the binary attachments list.

BODY_CONTAINER
The 16-byte name of the body container in the specified channel that contains XOP or XML data. This should be a container of DATATYPE(CHAR).

CHANNEL_NAME
Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

NAMESPACES_CONTAINER

Optional parameter

The 16-byte name of the container in the specified channel that contains the list of namespaces. The syntax is `xmlns:prefix="value"`.

Output Parameters

ATTACHMENTS_COUNT

The number of `<xop:Include>` elements that were processed. If the number is 0, the original body container does not include any XOP elements and has not been modified.

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_CCSID_ERROR
CONTAINER_WRONG_TYPE
ATTACHMENT_NOT_FOUND
INPUT_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIXO gate, BUILD_XOP function

Converts a standard XML message with base64binary encoded data into XOP format with separate binary attachments.

Input Parameters

ATTACHMENTS_CONTAINER

The 16-byte name of the container that contains the attachments list in the specified channel.

BODY_CONTAINER

The 16-byte name of the body container in the specified channel that contains XOP or XML data. This should be a container of DATATYPE(CHAR).

CHANNEL_NAME

Optional parameter

The 16-byte name of the channel for all referenced containers. If this parameter is omitted, then the current channel is assumed.

CID_DOMAIN_CONTAINER

The 16-byte name of the container that contains the domain name string that should be used as the last part of the content-ID, to identify the sysplex within which the locally unique value applies.

Output Parameters

ATTACHMENTS_COUNT

The number of `<xop:Include>` elements that were processed. If the number is 0, the original body container does not include any XOP elements and has not been modified.

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
CHANNEL_NOT_FOUND
CONTAINER_NOT_FOUND
CONTAINER_CCSID_ERROR
CONTAINER_WRONG_TYPE
```

CONTAINER_NAME_INVALID
INPUT_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Pipeline manager domain's generic gates

Table 38 summarizes the pipeline manager domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 38. Pipeline manager domain's generic gates

Gate	Trace	Function	Format
PIDM	PI 0100 PI 0101	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
PIST	PI 0200 PI 0201	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
PIRL	PI 3000 PI 3001	CREATE DISCARD INQUIRE INQUIRE_BY_NAME SET	RLCB
PIRN	PI 3100 PI 3101	CREATE DISCARD INQUIRE INQUIRE_BY_NAME SET	RLCB

Modules

Module	Function
DFHPIA1	Supports inbound and outbound WS-Addressing SOAP messages.
DFHPIAD	Supports the WS-Addressing API.
DFHPIAP	Remote stub program.
DFHPIAT	Supports PI domain's atomic transactions functions.
DFHPICA	CICS program for handling SCA composite resource type.
DFHPICC	Marshal XML body to COMMAREA and channel data.
DFHPIDM	Domain initialization and termination program.
DFHPIDSH	The pipeline HTTP inbound router module. Starts a service provider pipeline by issuing a DFHPIPM START_PIPELINE call to the pipeline manager.
DFHPIDUF	PI domain dump formatting program.
DFHPIII	Internal COMMAREA or container model (ICM) interpreter.
DFHPIIT	PI installation assist transaction program.
DFHPIIW	Pipeline manager support for PIIW gate.
DFHPILN	Pipeline callback program.
DFHPIMM	MIME Multipart/Related module that parses inbound MIME messages with binary attachments and builds outbound MIME messages.

Module	Function
DFHPIPA	SOAP envelope SAX parser.
DFHPIPL	PIPL gate functions.
DFHPIPM	Pipeline manager domain gate.
DFHPIRL	Handles requests from RL domain.
DFHPIRT	The pipeline HTTP outbound router module. Starts a service requester pipeline by issuing a DFHPIPM START_PIPELINE call to the pipeline manager.
DFHPISF	SOAP fault API support.
DFHPISN	SOAP node support.
DFHPISN1	SOAP 1.1 handler program.
DFHPISN2	SOAP 1.2 handler program.
DFHPIST	Pipeline manager's statistics gate.
DFHPITC	Trust handler client module.
DFHPITH	The pipeline HTTP transport management program which performs the functions of the PITG gate.
DFHPITL	Top level web service module
DFHPITP	PI domain's EXEC layer program
DFHPITQ	IBM MQ transport.
DFHPITQ1	CICS SOAP IBM MQ Transport program.
DFHPITRI	PI domain trace formatting program.
DFHPITS	The pipeline transport management program
DFHPIWR	WEBSERVICE resource functions.
DFHPIWT	Work request manager.
DFHPIXI	XOP parsing interface for handling inbound MIME Multipart/Related messages in compatibility mode.
DFHPIXO	XOP parsing interface for handling outbound MIME Multipart/Related messages in compatibility mode.
DFHPIXS	Describes a PIPELINE resource that is defined in a CICS bundle.

Partner management domain (PT)

The partner domain provides services to coordinate flows between two CICS tasks.

Partner management domain's specific gates

The specific gates provide access for other domains to functions that are provided by the PT domain.

PTTW gate, BREAK_PARTNERSHIP function

Break an established partnership.

Input Parameters

STATE_TOKEN
The state_token used to manage the handshake

COMPLETION_CODE
Optional Parameter

The completion code to be passed to the partner. The caller can use this to notify partner why the partnership is being broken. Once read the completion code is reset to zero. This is optional so that the caller can pass exactly one completion code when calling trigger_partner followed by break_partnership. The completion code is ignored if the resulting state is not_made.

Output Parameters

REASON

The values for the parameter are:

NOT_FOUND
NOT_PARTNER
PARTNERSHIP_NOT_MADE

PARTNER_COMPLETION_CODE

The partner's completion code indicates why the partner broke the partnership.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NEW_TRIGSTATE1

Optional Parameter

The state of partner 1 after the request.

Values for the parameter are:

RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

NEW_TRIGSTATE2

Optional Parameter

The state of partner 2 after the request.

Values for the parameter are:

RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

OLD_TRIGSTATE1

Optional Parameter

The state of partner 1 before the request.

Values for the parameter are:

RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

OLD_TRIGSTATE2

Optional Parameter

The state of partner 2 before the request.

Values for the parameter are:

RESUMED
TRIGGERED
UNDEFINED
VALID

WAITING

PTTW gate, CREATE_PARTNERSHIP function

Create a new state block to represent a partnership, and add it to the pool.

Input Parameters

POOL_TOKEN
The token of this pool

Output Parameters

REASON
The values for the parameter are:
POOL_NOT_FOUND
POOL_QUIESCING

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATE_TOKEN
The state_token used to manage the handshake

PTTW gate, CREATE_POOL function

The CREATE_POOL function creates a pool for state_tokens.

Input Parameters

GARBAGE_COLLECTION
Whether or not garbage collection is to be performed for state_tokens in this pool.

Values for the parameter are:
OFF
ON

POOL_NAME
The eight character name of the pool. This name must be unique across all pools. There is no enforced character set for this name.

FREE_USER_DATA_DOMAIN
Optional Parameter

An optional callback routine that may be called to free any user data addressed from the user_data_token associated with each state_token. This callback must implement the PTFD FREE_USER_DATA gate.

FREE_USER_DATA_GATE
Optional Parameter

An optional callback routine that may be called to free any user data addressed from the user_data_token associated with each state_token. This callback must implement the PTFD FREE_USER_DATA gate.

GARBAGE_COLLECT_INTERVAL
Optional Parameter

The interval in milliseconds between collections of garbage for this pool. If garbage collection is on, this parameter must be provided. If garbage collection is off, this parameter is ignored.

Output Parameters

REASON
The values for the parameter are:
BAD_CALLBACK
NAME_NOT_UNIQUE

POOL_TOKEN
The token of this pool
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PTTW gate, DESTROY_PARTNERSHIP function

Remove a state block from its pool and delete it to destroy the partnership. If the state token is still in use by the partner, it is flagged as deleted.

Input Parameters

STATE_TOKEN
The state_token used to manage the handshake

Output Parameters

REASON
The values for the parameter are:
NOT_FOUND
PARTNER_WAITING
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
NEW_TRIGSTATE1
Optional Parameter

The state of partner 1 after the request.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING
NEW_TRIGSTATE2
Optional Parameter

The state of partner 2 after the request.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING
OLD_TRIGSTATE1
Optional Parameter

The state of partner 1 before the request.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING
OLD_TRIGSTATE2
Optional Parameter

The state of partner 2 before the request.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

PTTW gate, DESTROY_POOL function

Destroys a pool of state_tokens.

Input Parameters

DESTROY_OPTION
Specifies how the pool is destroyed.

Values for the parameter are:
FORCE
MUST_BE_EMPTY
QUIESCE

POOL_TOKEN
The token of this pool

Output Parameters

REASON
The values for the parameter are:
POOL_NOT_EMPTY
POOL_NOT_FOUND
POOL_QUIESCING

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PTTW gate, END_POOL_BROWSE function

End a browse of pools.

Input Parameters

POOL_CURSOR
The browse cursor returned from start_pool_browse

Output Parameters

REASON
The values for the parameter are:
INVALID_CURSOR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PTTW gate, GET_NEXT_POOL function

Get the next pool

Input Parameters

POOL_CURSOR
The browse cursor returned from start_pool_browse

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
END_BROWSE

INVALID_CURSOR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

POOL_NAME
Optional Parameter

The eight character name of the pool. This name must be unique across all pools. There is no enforced character set for this name.

POOL_TOKEN
Optional Parameter

The token of this pool

PTTW gate, INQUIRE_GARBAGE_INTERVAL function

Get garbage collection interval.

Input Parameters

POOL_TOKEN
The token of this pool

Output Parameters

REASON
The values for the parameter are:
POOL_NOT_FOUND

GARBAGE_COLLECT_INTERVAL
The interval in milliseconds between collections of garbage for this pool. If garbage collection is on, this parameter must be provided. If garbage collection is off, this parameter is ignored.

GARBAGE_COLLECTION
Whether or not garbage collection is to be performed for state_tokens in this pool.

Values for the parameter are:
OFF
ON

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PTTW gate, INQUIRE_USER_TOKEN function

Get the user token in the state block.

Input Parameters

STATE_TOKEN
The state_token used to manage the handshake

Output Parameters

REASON
The values for the parameter are:
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USER_TOKEN
The user token to be associated with the state token

PTTW gate, MAKE_PARTNERSHIP function

Establish a partnership with another task. The partner task may or may not have previously made the partnership.

Input Parameters

ORDER

Specifies the order in which the partners make the partnership.

Values for the parameter are:

DONT_CARE
ONLY
SUBSEQUENT

STATE_TOKEN

The state_token used to manage the handshake

Output Parameters

REASON

The values for the parameter are:

ALREADY_MADE
ALREADY_PARTNER
NOT_FOUND
NOT_ONLY
NOT_PARTNER
NOT_SUBSEQUENT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NEW_TRIGSTATE1

Optional Parameter

The state of partner 1 after the request.

Values for the parameter are:

RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

NEW_TRIGSTATE2

Optional Parameter

The state of partner 2 after the request.

Values for the parameter are:

RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

OLD_TRIGSTATE1

Optional Parameter

The state of partner 1 before the request.

Values for the parameter are:

RESUMED
TRIGGERED
UNDEFINED
VALID

WAITING

OLD_TRIGSTATE2
Optional Parameter

The state of partner 2 before the request.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

PTTW gate, QUERY_PARTNERSHIP function

Get the status of the partner task.

Input Parameters

STATE_TOKEN
The state_token used to manage the handshake

Output Parameters

REASON
The values for the parameter are:
NOT_FOUND
NOT_PARTNER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

POOL_TOKEN
Optional Parameter

The token of this pool

STATE
Optional Parameter

Describes whether the state token is not made, made or partially made and who by.

Values for the parameter are:
MADE
MADE_BY_PARTNER
MADE_BY_SELF
NOT_MADE

STATUS_OF_PARTNER
Optional Parameter

Describes whether partner is waiting or has been triggered.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

STATUS_OF_SELF
Optional Parameter

Describes whether the caller has been triggered or not.

Values for the parameter are:
TRIGGERED

UNDEFINED
VALID

XM_TOKEN
Optional Parameter

The partner's transaction manager token.

PTTW gate, QUERY_POOL function

Query the attributes and state of a pool.

Input Parameters

POOL_NAME
The eight character name of the pool. This name must be unique across all pools. There is no enforced character set for this name.

POOL_TOKEN
The token of this pool

Output Parameters

REASON
The values for the parameter are:
POOL_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FREE_USER_DATA_DOMAIN
Optional Parameter

An optional callback routine that may be called to free any user data addressed from the user_data_token associated with each state_token. This callback must implement the PTFD FREE_USER_DATA gate.

FREE_USER_DATA_GATE
Optional Parameter

An optional callback routine that may be called to free any user data addressed from the user_data_token associated with each state_token. This callback must implement the PTFD FREE_USER_DATA gate.

GARBAGE_COLLECT_INTERVAL
Optional Parameter

The interval in milliseconds between collections of garbage for this pool. If garbage collection is on, this parameter must be provided. If garbage collection is off, this parameter is ignored.

GARBAGE_COLLECTION
Optional Parameter

Whether or not garbage collection is to be performed for state_tokens in this pool.

Values for the parameter are:
OFF
ON

POOL_NAME_OUT
Optional Parameter

The pool name is returned.

POOL_STATE
Optional Parameter

The cureent state of the pool.

Values for the parameter are:

EMPTY
NOT_EMPTY
QUIESCING

POOL_TOKEN_OUT
Optional Parameter

The pool token is returned.

PTTW gate, SET_GARBAGE_INTERVAL function

Set garbage collection interval.

Input Parameters

GARBAGE_COLLECT_INTERVAL
The interval in milliseconds between collections of garbage for this pool. If garbage collection is on, this parameter must be provided. If garbage collection is off, this parameter is ignored.

POOL_TOKEN
The token of this pool

Output Parameters

REASON
The values for the parameter are:
GARBAGE_COLLECTION_OFF
POOL_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PTTW gate, SET_USER_TOKEN function

Change the user token in the state block.

Input Parameters

STATE_TOKEN
The state_token used to manage the handshake

USER_TOKEN
The user token to be associated with the state token

Output Parameters

REASON
The values for the parameter are:
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PTTW gate, START_POOL_BROWSE function

Creates a pool cursor to browse pools.

Output Parameters

REASON
The values for the parameter are:
NO_POOLS

POOL_CURSOR
The browse cursor returned from start_pool_browse

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PTTW gate, TRIGGER_PARTNER function

Notify a waiting partner. If the partner is not waiting when trigger is called, the partner will be triggered when it next waits.

Input Parameters

COMPLETION_CODE

The completion code to be passed to the partner. The caller can use this to notify partner why the partnership is being broken. Once read the completion code is reset to zero. This is optional so that the caller can pass exactly one completion code when calling trigger_partner followed by break_partnership. The completion code is ignored if the resulting state is not_made.

PARTNER_EXISTENCE

Specifies whether the partner must exist for this request.

Values for the parameter are:

```
DONT_CARE
MUST_EXIST
```

STATE_TOKEN

The state_token used to manage the handshake

Output Parameters

REASON

The values for the parameter are:

```
ALREADY_TRIGGERED
NOT_FOUND
NOT_PARTNER
PARTNER_NOT_THERE
PARTNERSHIP_NOT_MADE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NEW_TRIGSTATE1

Optional Parameter

The state of partner 1 after the request.

Values for the parameter are:

```
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING
```

NEW_TRIGSTATE2

Optional Parameter

The state of partner 2 after the request.

Values for the parameter are:

```
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING
```

OLD_TRIGSTATE1

Optional Parameter

The state of partner 1 before the request.

Values for the parameter are:

RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

OLD_TRIGSTATE2
Optional Parameter

The state of partner 2 before the request.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

PTTW gate, WAIT_FOR_PARTNER function
Wait to be notified by a partner or until the wait times out.

Input Parameters
PARTNER_EXISTENCE
Specifies whether the partner must exist for this request.

Values for the parameter are:
DONT_CARE
MUST_EXIST

STATE_TOKEN
The state_token used to manage the handshake

PURGEABLE
Optional Parameter

Specifies whether the wait can be purged.

Values for the parameter are:
NO
YES

TIMEOUT
Optional Parameter

An optional maximum time to wait before waking up in milliseconds

Output Parameters
REASON
The values for the parameter are:
NOT_FOUND
NOT_PARTNER
PARTNER_NOT_THERE
PARTNER_WAITING
PARTNERSHIP_NOT_MADE
TIMED_OUT

PARTNER_COMPLETION_CODE
The partner's completion code indicates why the partner broke the partnership.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NEW_TRIGSTATE1
Optional Parameter

The state of partner 1 after the request.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

NEW_TRIGSTATE2
Optional Parameter

The state of partner 2 after the request.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

OLD_TRIGSTATE1
Optional Parameter

The state of partner 1 before the request.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

OLD_TRIGSTATE2
Optional Parameter

The state of partner 2 before the request.

Values for the parameter are:
RESUMED
TRIGGERED
UNDEFINED
VALID
WAITING

Modules

Module	Function
DFHPTDM	Domain initialization and termination. PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN

Module	Function
DFHPTTW	Handles the following requests: CREATE_POOL DESTROY_POOL QUERY_POOL START_POOL_BROWSE GET_NEXT_POOL END_POOL_BROWSE CREATE_PARTNERSHIP DESTROY_PARTNERSHIP SET_USER_TOKEN INQUIRE_USER_TOKEN MAKE_PARTNERSHIP BREAK_PARTNERSHIP TRIGGER_PARTNER WAIT_FOR_PARTNER QUERY_PARTNERSHIP SET_GARBAGE_INTERVAL INQUIRE_GARBAGE_INTERVAL

Resource life-cycle domain (RL)

The resource life-cycle domain handles the installation and life cycle of application resources.

Resource life-cycle domain's specific gates

The specific gates provide access for other domains to functions that are provided by the RL domain.

RLPM gate, DISCARD_BUNDLE function

Discards a disabled BUNDLE resource, releasing the associated storage.

Input parameters

BUNDLE_NAME

Optional parameter

An 8-byte character name of the bundle.

BUNDLE_TOKEN

Optional parameter

An 8-byte token that represents the created bundle. Either the BUNDLE_NAME or the BUNDLE_TOKEN is used, but not both.

MGMTPART

Indicates whether the BUNDLE was created by an application or platform deployment. This parameter can have a YES or NO value.

Output parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
RLPM_CLIENT_FAILED
RLPM_DISCARD_NOT_ALLOWED
RLPM_DUPLICATE_BUNDLE
RLPM_INVALID_STATE
RLPM_MANIFEST_INVALID
RLPM_MANIFEST_NOT_FOUND
RLPM_NOT_DISABLED
```

RLPM_NOT_FOUND
RLPM_RESOURCE_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLPM gate, END_BROWSE_BUNDLE function

Ends a browse session on installed BUNDLE resources.

Input parameters

BROWSE_TOKEN
The browse token for the browse operation.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_BROWSE
INVALID_DIRECTORY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLPM gate, GET_NEXT_BUNDLE function

Get the next installed BUNDLE resource to browse it.

Input parameters

BROWSE_TOKEN
The browse token for the browse operation.

RESOURCE_SIGNATURE
Optional parameter

The resource signature of the resource.

INQUIRE_VENDOR
Optional parameter

The bundle is provided by a vendor. The value of this parameter is YES or NO.

ROOT_BUFF
Optional parameter

A buffer for the root directory of the BUNDLE resource.

SCOPE_BUFF
Optional parameter

A buffer for the scope of the bundle.

Output parameters

AVAILSTATUS
Optional parameter

A 1-byte enumeration expressing the availability status of the CICS bundle.

Values for the parameter are:
AVAILABLE
UNAVAILABLE
NONE

BUNDLE_TOKEN
Optional parameter

An 8-byte token that represents the created bundle.

BUNDLE_NAME
An 8-byte character name of the bundle.

BUNDLE_ID
Optional parameter

The 64-character ID of the CICS bundle that is defined in the bundle manifest.

BUNDLE_MAJOR_VER
Optional parameter

The major version number of the CICS bundle that is defined in the bundle manifest.

BUNDLE_MICRO_VER
Optional parameter

The micro version number of the CICS bundle that is defined in the bundle manifest.

BUNDLE_MINOR_VER
Optional parameter

The minor version number of the CICS bundle that is defined in the bundle manifest.

DEFINE_COUNT
Optional parameter

The total number of dynamically created resources in the bundle.

ENABLED_COUNT
Optional parameter

The number of current resources that were dynamically created by the bundle and are enabled in the CICS region.

PART_COUNT
Optional parameter

The total number of imports, exports, and definition statements that are defined in the bundle manifest.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
RLPM_BROWSE_END
RLPM_CLIENT_FAILED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATE
A 1–byte enumeration expressing whether the initial state of the BUNDLE resource is enabled or disabled.

RLPM gate, INQUIRE_BUNDLE function

Inquire to find out if the BUNDLE resource is enabled or disabled.

Input parameters

BUNDLE_NAME
Optional parameter

An 8-byte character name of the bundle.

BUNDLE_TOKEN
Optional parameter

An 8-byte token that represents the created bundle. Either the BUNDLE_NAME or the BUNDLE_TOKEN is used, but not both.

INQUIRE_VENDOR
Optional parameter

The bundle is provided by a vendor. The value of this parameter is YES or NO.

ROOT_BUFF
Optional parameter

Abuffer for the root path of the bundle.

RESOURCE_SIGNATURE
Optional parameter

The resource signature of the resource.

SCOPE_BUFF
Optional parameter

A buffer for the scope of the bundle.

Output parameters

AVAILSTATUS
Optional parameter

A 1-byte enumeration expressing the availability status of the CICS bundle.

Values for the parameter are:

```
AVAILABLE
UNAVAILABLE
NONE
```

BUNDLE_ID
Optional parameter

The 64-character ID of the CICS bundle that is defined in the bundle manifest.

BUNDLE_MAJOR_VER
Optional parameter

The major version number of the CICS bundle that is defined in the bundle manifest.

BUNDLE_MICRO_VER
Optional parameter

The micro version number of the CICS bundle that is defined in the bundle manifest.

BUNDLE_MINOR_VER
Optional parameter

The minor version number of the CICS bundle that is defined in the bundle manifest.

DEFINE_COUNT
Optional parameter

The total number of dynamically created resources in the bundle.

ENABLED_COUNT
Optional parameter

The number of current resources that were dynamically created by the bundle and are enabled in the CICS region.

PART_COUNT
Optional parameter

The total number of imports, exports, and definition statements that are defined in the bundle manifest.

STATE
A 1–byte enumeration expressing whether the initial state of the BUNDLE resource is enabled or disabled.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
RLPM_CLIENT_FAILED
RLPM_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLPM gate, INSTALL_BUNDLE function

Creates a BUNDLE resource from a bundle that has been deployed into CICS.

Input parameters

BUNDLE_IDVER
Optional parameter

A character string that contains the ID and version of the CICS bundle that is defined in the bundle manifest.

BUNDLE_NAME
An 8-byte character name of the bundle.

CATALOGUE
Optional parameter

Add the BUNDLE resource to the CICS catalog. This parameter value is YES or NO.

RESOURCE_SIGNATURE
The resource signature of the resource.

ROOT
The fully qualified path of the root directory in the file system for the bundle.

SCOPE
Optional parameter.

A character string that contains the scope of the bundle as a URL.

STATE
A 1–byte enumeration expressing whether the initial state of the BUNDLE resource is enabled or disabled.

Output parameters

GENERATED_NAME
The generated name of the BUNDLE resource.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
RLPM_CLIENT_FAILED
RLPM_DUPLICATE_BUNDLE
RLPM_DUPLICATE_IDVER
RLPM_MANIFEST_INVALID
RLPM_MANIFEST_NOT_FOUND
RLPM_MANIFEST_NOT_AUTH
RLPM_RESOURCE_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLPM gate, SET_BUNDLE function

Set the status of the BUNDLE resource.

Input parameters

BUNDLE_NAME
Optional parameter

An 8-byte character name of the bundle.

BUNDLE_TOKEN
Optional parameter

An 8-byte token that represents the created bundle. Either the BUNDLE_NAME or the BUNDLE_TOKEN is used, but not both.

STATE
A 1–byte enumeration expressing whether the initial state of the BUNDLE resource is enabled or disabled, or available or unavailable. NOTAPPLIC means that the BUNDLE resource contains no application entry points, so availability status does not apply.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
RLPM_BUNDLE_SET_FAILED
RLPM_CLIENT_FAILED
RLPM_DUPLICATE_BUNDLE
RLPM_INVALID_STATE
RLPM_MANIFEST_INVALID
RLPM_MANIFEST_NOT_FOUND
RLPM_RESOURCE_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLPM gate, START_BROWSE_BUNDLE function

Start a browse session on installed BUNDLE resources.

Input parameters

None.

Output parameters

BROWSE_TOKEN
The browse token for the browse operation.

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_DIRECTORY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLRO gate, CREATED function

The CREATED function is called by the client domain after the BUNDLE resource is created.

Input parameters

BUNDLE_TOKEN
An 8-byte token that represents the created BUNDLE resource.

CLIENT_TOKEN
An 8-byte token that represents the client domain's view of the resource.

RESOURCE_TOKEN
An 8-byte token that represents the resource.

STATE
A 1–byte enumeration that expresses whether the state of the resource is enabled, disabled, or failed.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BAD_TOKEN
CATALOG_FULL
INVALID_DATA_LENGTH
IO_ERROR
RL_NOT_REGISTERED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLRO gate, DEREGISTER function

Deregister a resource type and its callback program.

Input parameters

TYPE
A character string that contains the URL for the type of resource.

CALLBACK_GATE
Optional parameter

The CICS callback gate that handles creating the resource type.

CALLBACK_PROGRAM
Optional parameter

The name of the program that handles creating the user resource type.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
RLRO_NOT_REGISTERED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLRO gate, DISCARDED function

The DISCARDED function is called by the client domain after the resource is discarded.

Input parameters

BUNDLE_TOKEN
An 8-byte token that represents the created BUNDLE resource.
RESOURCE_TOKEN
An 8-byte token that represents the resource.

Output parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLRO gate, DRIVE_PENDING function

Complete the creation of a BUNDLE resource during CICS initialization.

Input parameters

None.

Output parameters

None.

RLRO gate, END_BROWSE_BUNDLERES function

End a browse session on resources in an installed BUNDLE resource.

Input parameters

BROWSE_TOKEN
The browse token for the browse operation.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
RLRO_NOT_FOUND
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLRO gate, GET_NEXT_BUNDLERES function

Get the next resource from an installed BUNDLE to browse it.

Input parameters

BROWSE_TOKEN
The browse token for the browse operation.
FILE_BUFF
Optional parameter

A buffer for the artifact that defines the resource.
NAME_BUFF
Optional parameter

A buffer for the resource name.

OPERATION_BUFF
Optional parameter

A buffer for the operation name.

TYPE_BUFF
Optional parameter

A buffer for the resource type.

Output parameters

AVAILSTATUS
Optional parameter

The availability status of the resource. This parameter can have one of the following values:

```
AVAILABLE
UNAVAILABLE
NONE
```

BUNDLE
The 8-byte character name of the BUNDLE resource.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
RLRO_NOT_FOUND
```

RESCLASS
Optional parameter

The class of the resource. This parameter can have one of the following values:

```
DEFINE
IMPORT
EXPORT
ENTRYPOINT
POLICYSCOPE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATE
A 1-byte enumeration that expresses the state of the resource. This parameter can have one of the following values:

```
AVAILABLE
DISABLED
DISABLING
DISCARDING
ENABLED
ENABLING
FAILED
NOTAPPLIC
UNAVAILABLE
```

RLRO gate, NOTIFY function

The NOTIFY function is called by the client domain when the requested operation has completed.

Input parameters

BUNDLE_TOKEN
An 8-byte token that represents the created BUNDLE resource.

RESOURCE_TOKEN
An 8-byte token that represents the resource.

STATE
A 1–byte enumeration expressing whether the initial state of the BUNDLE resource is enabled or disabled, or available or unavailable. NOTAPPLIC means that the BUNDLE resource contains no application entry points, so availability status does not apply.

Output parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLRO gate, REGISTER function

Register a resource type and its callback program or domain.

Input parameters

CALLBACK_GATE
Optional parameter

The CICS callback gate that handles creating the resource type.

CALLBACK_PROGRAM
Optional parameter

The name of the program that handles creating the user resource type.

DELEGATE_RECOVERY
Optional parameter

Delegate the recovery of the resource. The value of this parameter is YES or NO.

TYPE
A character string that contains the URL for the type of resource.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
RLRO_ALREADY_REGISTERED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLRO gate, START_BROWSE_BUNDLERES function

Start a browse session on resources that were dynamically created by installing a BUNDLE resource.

Input parameters

BUNDLE
The 8-byte character name of the BUNDLE resource

Output parameters

BROWSE_TOKEN
The browse token for the browse operation.

REASON
The following values are returned when RESPONSE is EXCEPTION:
RLRO_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLXM gate, INQUIRE_SCOPE function

The INQUIRE_SCOPE function inquires on the **SCOPE** parameter on the **INVOKE SERVICE** command.

Input parameters

SCOPE_BUFFER

A buffer for the scope of the service.

Output parameters

REASON

The following values are returned when RESPONSE is DISASTER:

- ABEND
- LOOP

The following value is returned when RESPONSE is EXCEPTION:

- LENGTH_ERROR
- NO_SCOPE

The following values are returned when RESPONSE is INVALID:

- INVALID_FORMAT
- INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLXM gate, POP_SCOPE function

The POP_SCOPE function removes the SCOPE parameter on the **INVOKE SERVICE** command.

Input parameters

None.

Output parameters

REASON

The following values are returned when RESPONSE is DISASTER:

- ABEND
- LOOP

The following values are returned when RESPONSE is INVALID:

- INVALID_FORMAT
- INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLXM gate, PUSH_SCOPE function

The PUSH_SCOPE function saves the **SCOPE** parameter on the **INVOKE SERVICE** command.

Input parameters

SCOPE_BUFFER

A buffer for the scope of the service.

Output parameters

REASON

The following values are returned when RESPONSE is DISASTER:

- ABEND
- LOOP

The following value is returned when RESPONSE is EXCEPTION:
- LENGTH_ERROR

The following values are returned when RESPONSE is INVALID:
- INVALID_FORMAT
- INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLXM gate, RELEASE_XM_CLIENT function

The RELEASE_XM_CLIENT function releases the XM client.

Input parameters

None.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:
- ABEND
- LOOP

The following values are returned when RESPONSE is INVALID:
- INVALID_FORMAT
- INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Resource life-cycle domain's generic gates

Table 39 summarizes the Resource life-cycle domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 39. Resource life-cycle domain's generic gates

Gate	Trace	Function	Format
RLDM	RL 0100 RL 0101	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
RLST	RL 0200 RL 0201	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

Resource life-cycle domain's call-back formats

Table 40 on page 942 describes the call-back formats owned by the domain and shows the functions performed on the calls.

Table 40. Resource life-cycle domain's call-back formats

Format	Calling module	Function
RLCB		CREATE DISCARD INQUIRE INQUIRE_BY_NAME RECREATE RECREATE_COMPLETE SET SET_MODIFY

In the descriptions for the formats, the input parameters are input not to the resource life-cycle domain, but to the domain being called by the recovery life-cycle domain. Similarly, the output parameters are output by the domain that was called by the resource life-cycle domain, in response to the call.

RLCB gate, CREATE function

The CREATE function is called on the client domain by the RL domain to create a resource that is owned by the domain.

Input parameters

APPLICATION
A 64-byte character string that contains the application ID.

BUNDLE_NAME
An 8-byte character name of the bundle.

BUNDLE_TOKEN
Optional parameter

An 8-byte token that represents the created bundle. Either the BUNDLE_NAME or the BUNDLE_TOKEN is used, but not both.

CATALOG
A 1-byte enumeration expressing whether the client should catalog the resource or not.

DATA
Contains the metadata for the resource.

MAJOR_VER
A 4-byte numeric field that contains the application major version.

MICRO_VER
A 4-byte numeric field that contains the application micro version.

MINOR_VER
A 4-byte numeric field that contains the application minor version.

NAME
An 8-byte character string that contains the name of the resource.

PLATFORM
A 64-byte character string that contains the platform name.

RESOURCE_TOKEN
An 8-byte token that represents the resource.

ROOT
The fully qualified path of the root directory in the file system for the bundle.

SCOPE
Optional parameter.

A character string that contains the scope of the bundle as a URL.

STATE
A 1–byte enumeration expressing whether the initial state of the BUNDLE resource is enabled or disabled.

TYPE
A character string that contains the URL for the type of resource.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLCB gate, DISCARD function

The DISCARD function is called on the client domain by the RL domain to request that the resource is discarded by the client domain.

Input parameters

BUNDLE_TOKEN
Optional parameter

An 8-byte token that represents the created bundle. Either the BUNDLE_NAME or the BUNDLE_TOKEN is used, but not both.
CLIENT_TOKEN
An 8-byte token that represents the client domain's view of the resource.
RESOURCE_TOKEN
An 8-byte token that represents the resource.

Output parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLCB gate, INQUIRE function

The INQUIRE function is called on the client domain by the RL domain to inquire on the state of a resource that is owned by the domain.

Input parameters

BUNDLE_TOKEN
Optional parameter

An 8-byte token that represents the created bundle. Either the BUNDLE_NAME or the BUNDLE_TOKEN is used, but not both.
CLIENT_TOKEN
An 8-byte token that represents the client domain's view of the resource.
RESOURCE_TOKEN
An 8-byte token that represents the resource.

Output parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
STATE
A 1-byte enumeration that expresses the state of the resource. This parameter can have one of the following values:

```
AVAILABLE
DISABLED
DISABLING
DISCARDING
ENABLED
```

```
ENABLING
FAILED
NOTAPPLIC
UNAVAILABLE
```

RLCB gate, INQUIRE_MODIFY function

The INQUIRE_MODIFY function is called on the client domain by the RL domain to query the state of a bundle, to determine the state of a resource.

Input parameters

APPLICATION
A character string that contains the application ID.

BUNDLE_NAME
An 8-byte character name of the bundle.

BUNDLE_STATE
A 1-byte enumeration expressing the current state of the bundle resource. The state can be one of:

- Enabled
- Disabled
- Enabling
- Disabling
- Discarding

BUNDLE_TOKEN
An 8-byte token that represents the bundle.

MAJOR_VER
A 4-byte numeric field that contains the application major version.

MICRO_VER
A 4-byte numeric field that contains the application micro version.

MINOR_VER
A 4-byte numeric field that contains the application minor version.

MODIFY_TYPE
A 1–byte enumeration expressing whether the resource modifier is an entry point or policy scope.

NAME
A character string that contains the name of the resource.

OPERATION
A character string that contains the operation name.

PLATFORM
A character string that contains the platform name.

RESOURCE_TOKEN
An 8-byte token that represents the resource.

SCOPE
A character string that contains the scope of the bundle as a URL.

STATE
A 1–byte enumeration expressing whether the resource is enabled or disabled.

TYPE
A character string that contains the URL for the type of resource.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATE
A 1–byte enumeration that expresses the state of the resource:

- Enabled
- Disabled
- Enabling
- Disabling
- Discarding

RLCB gate, INQUIRE_BY_NAME function

Inquire on imports that are defined in the bundle.

Input parameters

TYPE
A character string that contains the URL for the type of resource.

NAME
An 8-byte character string that contains the name of the bundle.

SCOPE
A character string that contains the URL of the bundle.

Output parameters

STATE
A 1-byte enumeration that expresses the state of the resource:
- Enabled
- Disabled
- Enabling
- Disabling
- Discarding

REASON
The following values are returned when RESPONSE is EXCEPTION:

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLCB gate, RECREATE function

The RECREATE function is called on the client domain by the RL domain to re-create a resource that is owned by the domain during recovery of the CICS catalog.

Input parameters

BUNDLE_NAME
An 8-byte character name of the bundle.

BUNDLE_TOKEN
Optional parameter

An 8-byte token that represents the created bundle. Either the BUNDLE_NAME or the BUNDLE_TOKEN is used, but not both.

CLIENT_TOKEN
A token that represents the existing resource to the calling domain.

NEW_BUNDLE_TOKEN
A token representing the recovered BUNDLE resource.

NEW_RESOURCE_TOKEN
A token representing the recovered resource in the BUNDLE.

RESOURCE_TOKEN
An 8-byte token that represents the resource.

ROOT
The fully qualified path of the root directory in the file system for the bundle.

SCOPE
Optional parameter.

A character string that contains the scope of the bundle as a URL.

STATE
A 1–byte enumeration expressing whether the initial state of the BUNDLE resource is enabled or disabled.

TYPE
A character string that contains the URL for the type of resource.

Output parameters

NEW_CLIENT_TOKEN
A replacement client token representing the resource to the calling domain.

STATE
The state of the resource.

REASON
The following values are returned when RESPONSE is EXCEPTION:

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLCB gate, RECREATE_COMPLETE function

The RECREATE_COMPLETE function notifies the client domain that all resources of a particular type were re-created during recovery of the CICS catalog.

Input parameters

TYPE
A character string that contains the URL for the type of resource.

Output parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLCB gate, SET function

The SET function is called on the client domain by the RL domain to request that an action is performed on a resource owned by the domain.

Input parameters

BUNDLE_TOKEN
Optional parameter

An 8-byte token that represents the created bundle. Either the BUNDLE_NAME or the BUNDLE_TOKEN is used, but not both.

CLIENT_TOKEN
An 8-byte token that represents the client domain's view of the resource.

RESOURCE_TOKEN
An 8-byte token that represents the resource.

STATE
A 1–byte enumeration expressing whether the initial state of the BUNDLE resource is enabled or disabled.

Output parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RLCB gate, SET_MODIFY function

The SET_MODIFY function is called on the client domain by the RL domain to request a modification to a resource that is owned by the domain.

Input parameters

APPLICATION
A character string that contains the application ID.

BUNDLE_NAME
An 8-byte character name of the bundle.

BUNDLE_TOKEN
An 8-byte token that represents the created bundle.

MAJOR_VER
A 4-byte numeric field that contains the application major version.

MICRO_VER
A 4-byte numeric field that contains the application micro version.

MINOR_VER
A 4-byte numeric field that contains the application minor version.

MODIFY_TYPE
A 1–byte enumeration expressing whether the resource modifier is an entry point or policy scope.

NAME
A character string that contains the name of the resource.

OPERATION
A character string that contains the operation name.

PLATFORM
A character string that contains the platform name.

RESOURCE_TOKEN
An 8-byte token that represents the resource.

SCOPE
A character string that contains the scope of the bundle as a URL.

STATE
A 1–byte enumeration expressing whether the resource is enabled or disabled, or available or unavailable. NONE means that the BUNDLE resource contains no application entry points, so availability status does not apply.

TYPE
A character string that contains the URL for the type of resource.

Output parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHRLCB	Callback handler
DFHRLDM	Domain initialization and termination program
DFHRLDUF	Dump formatting program
DFHRLMF	Contains the data structures for processing bundle manifests
DFHRLPK	Driven by DFHRLPM to manage bundles
DFHRLPM	Bundle manager that drives DFHRLPK
DFHRLRG	Resource type handler
DFHRLRO	Bundle manager gate module
DFHRLRP	RL resolution program
DFHRLRS	Resource state and operations function
DFHRLSC	Contains the schema for handling SCA composite resource types

Module	Function
DFHRLST	Statistics manager
DFHRLTRI	Trace formatting program
DFHRLVP	Variable domain subpool allocate and free function
DFHRLXM	RL domain XM attach client program

Recovery manager domain (RM)

The Recovery manager (RM) domain is responsible for ensuring that the resource updates for a unit of work are all committed or all backed out, including updates across multiple systems.

Recovery manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the RM domain.

RMCD gate, INQUIRE_CLIENT_DATA function

This function returns data associated with a Recovery Manager client.

Input Parameters

CLIENT_DATA_BUFFER
A buffer to contain the data returned.

CLIENT_NAME
Name of the communications protocol used on the link.

Output Parameters

REASON
The values for the parameter are:

```
CLIENT_DATA_TOO_LONG
UNKNOWN_CLIENT
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMCD gate, REGISTER function

This function is used to register a Recovery Manager client.

Input Parameters

CLIENT_NAME
Name of the communications protocol used on the link.

CLIENT_TYPE
Whether the client owns local (RO) or remote (RMC) resources.

Values for the parameter are:

```
RMC
RO
```

GATE
Optional Parameter

An optional parameter specifying the kernel gate that services the client's callback functions.

Output Parameters

REASON

The values for the parameter are:

```
ALREADY_REGISTERED
TOO_LATE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMCD gate, SET_CLIENT_DATA function

This function associates some data with a Recovery Manager client.

Input Parameters

CLIENT_DATA_BUFFER

A buffer to contain the data returned.

CLIENT_NAME

Name of the communications protocol used on the link.

Output Parameters

REASON

The values for the parameter are:

```
CLIENT_DATA_TOO_LONG
UNKNOWN_CLIENT
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMCD gate, SET_GATE function

This function is used to inform Recovery Manager of the kernel gate that services a Recovery Manager clients callback functions.

Input Parameters

CLIENT_NAME

Name of the communications protocol used on the link.

GATE

An optional parameter specifying the kernel gate that services the client's callback functions.

THREADSAFE

An optional parameter that Recovery Manager clients can use to inform the Recovery Manager whether or not they are threadsafe.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON

The values for the parameter are:

```
GATE_ALREADY_SET
UNKNOWN_CLIENT
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMDM gate, INQUIRE_LOCAL_LU_NAME function

This function inquires on the local LU name, which is used in the generation of network UOWIDs in this system.

Output Parameters

LOCAL_LU_NAME
The local LU name.

LOCAL_LU_NAME_LENGTH
The length of the local LU name

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMDM gate, INQUIRE_STARTUP function

This function returns information about the type of system start being performed.

Output Parameters

ALL
A value specifying whethera cold start is being performed for all components.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STARTUP
The type of system start being performed.

Values for the parameter are:
COLD
EMERGENCY
WARM

INITIAL_START
Optional Parameter

A value specifying whether the cold start is in fact an initial one.

Values for the parameter are:
NO
YES

LAST_COLD_START_TIME
Optional Parameter

An 8 byte Store Clock representation of the last cold start time.

LAST_EMER_START_TIME
Optional Parameter

An 8 byte Store Clock representation of the last emergency start time.

LAST_INIT_START_TIME
Optional Parameter

An 8 byte Store Clock representation of the last initial start time.

RMDM gate, SET_LOCAL_LU_NAME function

This function sets the local LU name, which is used in the generation of network UOWIDs in this system.

Input Parameters

LOCAL_LU_NAME
A parameter specifying the local LU name.

LOCAL_LU_NAME_LENGTH
A parameter specifying the length of the local LU name.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMDM gate, SET_PARAMETERS function

This function is used only by Parameter Manager Domain to inform Recovery Manager of initialization parameters.

Input Parameters

DELETE_LOG

Optional Parameter

An optional parameter specifying whether an initial start has been requested in the System Initialization Table, and so the contents of the system log should be deleted.

Values for the parameter are:

NO
YES

STARTUP

Optional Parameter

The type of start.

Values for the parameter are:

EMERGENCY

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMDM gate, SET_STARTUP function

This function sets the type of start that will be performed when this system is next restarted.

Input Parameters

STARTUP

The type of start.

Values for the parameter are:

COLD
NORESTART

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, ADD_LINK function

This function adds a link to a remote system to a unit of work. The unit of work is distributed across more than one system and Recovery Manager will manage the syncpoint processing between systems.

Input Parameters

CLIENT_NAME

Name of the communications protocol used on the link.

RMC_TOKEN
A token to be passed to the client on all callback functions.

COORDINATOR
Optional Parameter

A parameter specifying whether the remote system is the coordinator of the distributed unit of work.

Values for the parameter are:
NO
YES

INITIATOR
Optional Parameter

A parameter specifying whether the remote system is the initiator of the syncpoint.

Values for the parameter are:
NO
YES

LAST
Optional Parameter

A parameter specifying whether the remote system supports the last agent optimization.

Values for the parameter are:
DESIRABLE
MAYBE
NO
YES

LINK_ID_BUFFER
Optional Parameter

A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LINK_ID_SOURCE
Optional Parameter

An optional parameter specifying whether the local or remote system allocated the session.

Values for the parameter are:
LOCAL
REMOTE

LOGNAME_BUFFER
Optional Parameter

An optional parameter specifying a buffer containing the logname of the remote system.

NO_RESYNC_OUTCOME
Optional Parameter

A binary value indicating that the link will not provide a resolution to the distributed unit-of-work during resynchronization.

Values for the parameter are:
NO
YES

OTS_HOSTNAME_BUFFER
Optional Parameter

A buffer in which the TCP/IP host name is supplied.

OTS_IORSTRING_BUFFER
Optional Parameter

A buffer containing the OTS IOR string.

PRELOGGING
Optional Parameter

A parameter specifying whether the client requires to be called with the PERFORM_PRELOGGING callback function.

Values for the parameter are:
NO
YES

PRESUMPTION
Optional Parameter

A parameter specifying whether the remote system assumes the presume abort or presume nothing protocols.

Values for the parameter are:
ABORT
NOTHING

RECOVERY_STATUS
Optional Parameter

A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system.

Values for the parameter are:
NECESSARY
SYNC_LEVEL_1
UNNECESSARY

REMOTE_ACCESS_ID_BUFFER
Optional Parameter

A buffer containing the netname of the remote system, or the name of the External Resource Manager.

SINGLE_UPDATER
Optional Parameter

A parameter specifying whether the remote system supports the single updater optimization.

Values for the parameter are:
NO
YES

UOW_ID
Optional Parameter

An optional parameter specifying the network UOWID to be given to the unit of work object. This parameter will be present if the unit of work being created is part of a distributed unit of work that originated on another system.

VOLATILE
Optional Parameter

A binary parameter indicating whether the link is volatile.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The values for the parameter are:

```
CLIENT_UNKNOWN
INVALID_SYNCPOINT_STATE
UOW_UNKNOWN
```

LINK_TOKEN

A token that identifies the Recovery Manager Link object.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, DELETE_LINK function

This function removes a link to a remote system from a unit of work. The remote system will not now be included in syncpoint processing for the current unit of work.

Input Parameters

LINK_TOKEN

A token identifying the Recovery Manager Link object.

Output Parameters

REASON

The values for the parameter are:

```
LINK_UNKNOWN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, END_LINK_BROWSE function

This function is used to terminate a browse of Recovery Manager Link objects.

Input Parameters

LINK_BROWSE_TOKEN

Optional Parameter

A token identifying a browse of all the Recovery Manager Link objects belonging to a particular Recovery Manager client.

UOW_BROWSE_TOKEN

Optional Parameter

A token identifying a browse of all the Recovery Manager Link objects belonging to a particular unit of work object.

Output Parameters

REASON

The values for the parameter are:

```
INVALID_BROWSE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, GET_NEXT_LINK function

This function returns information about the next Recovery Manager Link object in a browse.

Input Parameters

LINK_BROWSE_TOKEN
Optional Parameter

A token identifying a browse of all the Recovery Manager Link objects belonging to a particular Recovery Manager client.

LINK_ID_BUFFER
Optional Parameter

A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LOGNAME_BUFFER
Optional Parameter

An optional parameter specifying a buffer containing the logname of the remote system.

OTS_HOSTNAME_BUFFER
Optional Parameter

A buffer in which the TCP/IP host name is returned.

OTS_IORSTRING_BUFFER
Optional Parameter

A buffer containing the OTS IOR string.

REMOTE_ACCESS_ID_BUFFER
Optional Parameter

A buffer containing the netname of the remote system, or the name of the External Resource Manager.

UOW_BROWSE_TOKEN
Optional Parameter

A token identifying a browse of all the Recovery Manager Link objects belonging to a particular unit of work object.

Output Parameters

REASON
The values for the parameter are:

```
END_BROWSE
INVALID_BROWSE
UOW_UNKNOWN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESSIBLE
Optional Parameter

Whether the communications link to the remote system is active or not.

Values for the parameter are:

```
NO
SHUNTED
YES
```

CLIENT_NAME
Optional Parameter

The name of the Recovery Manager client that owns the resource that has caused the unit of work to shunt.

COORDINATOR
Optional Parameter

Whether the remote system is the coordinator of the distributed unit of work.

Values for the parameter are:
NO
YES

FORGET
Optional Parameter

Whether all obligations to the remote system with respect to recovery have been discharged.

Values for the parameter are:
NO
YES

HEURISM
Optional Parameter

Whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window.

Values for the parameter are:
NO
YES

INITIATOR
Optional Parameter

Whether the remote system is the initiator of the syncpoint of the distributed unit of work.

Values for the parameter are:
NO
YES

LAST
Optional Parameter

Whether the remote system supports the last agent optimization.

Values for the parameter are:
MAYBE
NO
YES

LINK_ID_SOURCE
Optional Parameter

Whether the local or remote system allocated the session.

Values for the parameter are:
LOCAL
REMOTE

LINK_TOKEN
Optional Parameter

A token identifying the new Recovery Manager Link object.

LOCAL_UOW_ID
Optional Parameter

An optional parameter to receive the local UOWID.

LOGICAL_SERVER
Optional Parameter

The logical server associated with the link.

MARK
Optional Parameter

Whether the Recovery Manager Link object has been marked during resynchronization.

Values for the parameter are:
NO
YES

PRESUMPTION
Optional Parameter

Whether the remote system assumes the presume abort or presume nothing protocols.

Values for the parameter are:
ABORT
NOTHING

PUBLIC_ID
Optional Parameter

The public identifier of the RequestStream associated with the link.

RECOVERY_STATUS
Optional Parameter

Whether recoverable work has taken place as part of the distributed unit of work on the remote system.

Values for the parameter are:
NECESSARY
SYNC_LEVEL_1
UNNECESSARY

RESYNC_SCHEDULED
Optional Parameter

Whether resynchronization activity has been scheduled.

Values for the parameter are:
NO
YES

RMC_TOKEN
Optional Parameter

A token to be passed to the client on all callback functions.

SINGLE_UPDATER
Optional Parameter

Whether the remote system supports the single updater optimization.

Values for the parameter are:
NO
YES

UNSHUNTED
Optional Parameter

Whether the unit of work is not currently shunted.

Values for the parameter are:
NO
YES

UOW_TOKEN
Optional Parameter

A token identifying the unit of work object.

RMLN gate, INBOUND_FLOW function

This function is used to notify Recovery Manager of the successful completion of syncpoint processing on the remote system, or a communications failure with the remote system.

Input Parameters

FLOW

A parameter specifying successful completion (DATA) or communication failure (UNBIND).

Values for the parameter are:

```
DATA
UNBIND
```

LINK_TOKEN

A token identifying the Recovery Manager Link object.

Output Parameters

REASON

The values for the parameter are:

```
LINK_INACCESSIBLE
LINK_UNKNOWN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, INITIATE_RECOVERY function

This function identifies a Recovery Manager Link object in an in doubt failed unit of work and marks it as being resynchronized.

Input Parameters

CLIENT_NAME

Name of the communications protocol used on the link.

DIRECTION

Parameter specifying whether to commit (FORWARD), backout (BACKWARD) or obey the ACTION attribute in the definition of the originating transaction.

Values for the parameter are:

```
INBOUND
OUTBOUND
```

COORDINATOR_LINK

Optional Parameter

A binary value indicating whether the remote system is the coordinator of the distributed unit of work.

Values for the parameter are:

```
YES
```

LINK_ID_BUFFER

Optional Parameter

A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LINK_ID_SOURCE

Optional Parameter

An optional parameter specifying whether the local or remote system allocated the session.

Values for the parameter are:

LOCAL
REMOTE

LOCAL_UOW_ID
Optional Parameter

The local UOWID of the required unit of work.

OTS_IORSTRING_BUFFER
Optional Parameter

A buffer containing the OTS IOR string.

REMOTE_ACCESS_ID_BUFFER
Optional Parameter

A buffer containing the netname of the remote system, or the name of the External Resource Manager.

UOW_ID
Optional Parameter

An optional parameter specifying the network UOWID to be given to the unit of work object. This parameter will be present if the unit of work being created is part of a distributed unit of work that originated on another system.

Output Parameters

REASON
The values for the parameter are:
LINK_ACTIVE
LINK_UNKNOWN
RECOVERY_ALREADY_IN_PROG

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

COORDINATOR
Optional Parameter

Whether the remote system is the coordinator of the distributed unit of work.

Values for the parameter are:
NO
YES

FAILURE_TIME
Optional Parameter

An 8 byte Store Clock representation of the in doubt failure time.

INITIATOR
Optional Parameter

Whether the remote system is the initiator of the syncpoint of the distributed unit of work.

Values for the parameter are:
NO
YES

LINK_TOKEN
Optional Parameter

A token identifying the new Recovery Manager Link object.

PRESUMPTION
Optional Parameter

Whether the remote system assumes the presume abort or presume nothing protocols.

Values for the parameter are:
ABORT
NOTHING

UOW_STATUS
Optional Parameter

The status of the unit of work.

Values for the parameter are:
BACKWARD
FORWARD
HEURISTIC_BACKWARD
HEURISTIC_FORWARD
INDOUBT

UOW_TOKEN
Optional Parameter

A token identifying the unit of work object.

RMLN gate, INQUIRE_LINK function

This function returns information about a given Recovery Manager Link object.

Input Parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object.

LINK_ID_BUFFER
Optional Parameter

A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LOGNAME_BUFFER
Optional Parameter

An optional parameter specifying a buffer containing the logname of the remote system.

OTS_HOSTNAME_BUFFER
Optional Parameter

A buffer in which the TCP/IP host name is returned.

OTS_IORSTRING_BUFFER
Optional Parameter

A buffer containing the OTS IOR string.

REMOTE_ACCESS_ID_BUFFER
Optional Parameter

A buffer containing the netname of the remote system, or the name of the External Resource Manager.

RESOLVE_TO_CURRENT_LINK
Optional Parameter

Up to two Recovery Manager Link objects may be associated with a token. This optional parameter specifies whether to return information about the most recent or not.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The values for the parameter are:

LINK_UNKNOWN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESSIBLE

Optional Parameter

Whether the communications link to the remote system is active or not.

Values for the parameter are:

NO
SHUNTED
YES

CLIENT_NAME

Optional Parameter

The name of the Recovery Manager client that owns the resource that has caused the unit of work to shunt.

COORDINATOR

Optional Parameter

Whether the remote system is the coordinator of the distributed unit of work.

Values for the parameter are:

NO
YES

CURRENT_TOKEN

Optional Parameter

The link token of the current link.

FORGET

Optional Parameter

Whether all obligations to the remote system with respect to recovery have been discharged.

Values for the parameter are:

NO
YES

HEURISM

Optional Parameter

Whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window.

Values for the parameter are:

NO
YES

INITIATOR

Optional Parameter

Whether the remote system is the initiator of the syncpoint of the distributed unit of work.

Values for the parameter are:

NO
YES

LAST

Optional Parameter

Whether the remote system supports the last agent optimization.

Values for the parameter are:
MAYBE
NO
YES

LINK_ID_SOURCE
Optional Parameter

Whether the local or remote system allocated the session.

Values for the parameter are:
LOCAL
REMOTE

LOCAL_UOW_ID
Optional Parameter

An optional parameter to receive the local UOWID.

LOGICAL_SERVER
Optional Parameter

The logical server associated with the link.

MARK
Optional Parameter

Whether the Recovery Manager Link object has been marked during resynchronization.

Values for the parameter are:
NO
YES

PRESUMPTION
Optional Parameter

Whether the remote system assumes the presume abort or presume nothing protocols.

Values for the parameter are:
ABORT
NOTHING

PUBLIC_ID
Optional Parameter

The public identifier of the RequestStream associated with the link.

RECOVERY_STATUS
Optional Parameter

Whether recoverable work has taken place as part of the distributed unit of work on the remote system.

Values for the parameter are:
NECESSARY
SYNC_LEVEL_1
UNNECESSARY

RESYNC_SCHEDULED
Optional Parameter

Whether resynchronization activity has been scheduled.

Values for the parameter are:
NO
YES

RMC_TOKEN
Optional Parameter

A token to be passed to the client on all callback functions.

SINGLE_UPDATER
Optional Parameter

Whether the remote system supports the single updater optimization.

Values for the parameter are:
NO
YES

UNSHUNTED
Optional Parameter

Whether the unit of work is not currently shunted.

Values for the parameter are:
NO
YES

UOW_TOKEN
Optional Parameter

A token identifying the unit of work object.

RMLN gate, INSERT_LINK function

Insert a link into the link-set of the current unit of work.

Input Parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object.

Output Parameters

REASON
The values for the parameter are:
COORDINATOR_ALREADY
LINK_UNKNOWN
NOT_REMOVED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, ISSUE_PREPARE function

This function performs phase 1 of syncpoint processing on the specified Recovery Manager Link object.

Input Parameters

CONTINUE
Is the task continuing into a following, new unit of work.

Values for the parameter are:
NO
YES

LINK_TOKEN
A token identifying the Recovery Manager Link object.

Output Parameters

REASON
The values for the parameter are:
COORDINATOR_ALREADY

INITIATOR_ALREADY
LINK_UNKNOWN
PREPARE_REJECTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

VOTE
The vote from the client owning the Recovery Manager Link object.

Values for the parameter are:
NO
NO_CONTINUE
READ_ONLY
YES

RMLN gate, RECORD_VOTE function

Record a link's vote in a distributed syncpoint.

Input Parameters

HEURISM
A binary value indicating whether the vote is heuristic.

Values for the parameter are:
NO
YES

LINK_TOKEN
A token identifying the Recovery Manager Link object.

VOTE
The link's vote.

Values for the parameter are:
NO
NO_CONTINUE
READ_ONLY
YES

Output Parameters

REASON
The values for the parameter are:
COORDINATOR_ALREADY
INITIATOR_ALREADY
LINK_UNKNOWN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, REMOVE_LINK function

This function remove a link to a remote system from a unit of work.

Input Parameters

LINK_TOKEN
A token that identifies the Recovery Manager Link object.

Output Parameters

REASON
The values for the parameter are:
ALREADY_REMOVED
LINK_UNKNOWN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, REPORT_RECOVERY_STATUS function

This function is similar to SET_RECOVERY_STATUS but is applicable in the case of Presumed Abort or Last Agent resynchronization where the coordinator has backed out and has no record of the UOW. The participant may have gone indoubt, and needs to resynchronize.

Input Parameters

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system, or the name of the External Resource Manager.

REMOTE_UOW_STATUS
The status of the unit of work in the remote system.

Values for the parameter are:
```
HEURISTIC_BACKWARD
HEURISTIC_FORWARD
HEURISTIC_MIXED
INDOUBT
```

UOW_ID
An optional parameter specifying the network UOWID to be given to the unit of work object. This parameter will be present if the unit of work being created is part of a distributed unit of work that originated on another system.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
ALREADY_REMOVED
ALREADY_SET
CLIENT_UNKNOWN
COORDINATOR_ALREADY
END_BROWSE
INITIATOR_ALREADY
INVALID_SYNCPOINT_STATE
LINK_ACTIVE
LINK_INACCESSIBLE
LINK_UNKNOWN
NO_FORGET_PENDING
NOT_REMOVED
PREPARE_REJECTED
RECOVERY_ALREADY_IN_PROG
RECOVERY_IN_PROGRESS
RECOVERY_NOT_IN_PROGRESS
SET_NOT_DONE
UOW_UNKNOWN
VOTED_ALREADY
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_BROWSE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, SET_LINK function

This function is used to set characteristics of a Recovery Manager Link object.

Input Parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object.

ACCESSIBLE
Optional Parameter

A parameter specifying that the communications link to the remote system has failed.

Values for the parameter are:
NO
SHUNTED

COORDINATOR
Optional Parameter

A parameter specifying whether the remote system is the coordinator of the distributed unit of work.

Values for the parameter are:
NO
YES

FORGET
Optional Parameter

A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged.

Values for the parameter are:
NO
YES

INITIATOR
Optional Parameter

A parameter specifying whether the remote system is the initiator of the syncpoint.

Values for the parameter are:
NO
YES

LINK_ID_BUFFER
Optional Parameter

A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LINK_ID_SOURCE
Optional Parameter

An optional parameter specifying whether the local or remote system allocated the session.

Values for the parameter are:
LOCAL
REMOTE

LOGNAME_BUFFER
Optional Parameter

An optional parameter specifying a buffer containing the logname of the remote system.

PRELOGGING
Optional Parameter

A parameter specifying whether the client requires to be called with the PERFORM_PRELOGGING callback function.

Values for the parameter are:
NO
YES

RECOVERY_STATUS
Optional Parameter

A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system.

Values for the parameter are:
NECESSARY
SYNC_LEVEL_1
UNNECESSARY

RESOLVE_TO_CURRENT_LINK
Optional Parameter

Up to two Recovery Manager Link objects may be associated with a token. This optional parameter specifies whether to return information about the most recent or not.

Values for the parameter are:
NO
YES

RESYNC_SCHEDULED
Optional Parameter

A parameter specifying whether resynchronization activity has been scheduled.

Values for the parameter are:
NO
YES

SINGLE_UPDATER
Optional Parameter

A parameter specifying whether the remote system supports the single updater optimization.

Values for the parameter are:
NO
YES

UNSHUNTED
Optional Parameter

A parameter specifying whether the unit of work is not currently shunted.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
COORDINATOR_ALREADY
INITIATOR_ALREADY
INVALID_SYNCPOINT_STATE
LINK_UNKNOWN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, SET_MARK function

This function marks a Recovery Manager Link object during recovery.

Input Parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object.

MARK
Optional Parameter

Binary parameter indicating whether the links should be marked.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
LINK_ACTIVE
LINK_UNKNOWN
RECOVERY_IN_PROGRESS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLN gate, SET_RECOVERY_STATUS function

This function is used to notify an Recovery Manager Link object of the outcome of a distributed unit of work which failed in the in doubt window. It results in the shunted unit of work the Recovery Manager Link object belongs to unshunting and committing or backing out its resource updates as appropriate.

Input Parameters

DIRECTION
Parameter specifying whether to commit (FORWARD), backout (BACKWARD) or obey the ACTION attribute in the definition of the originating transaction.

Values for the parameter are:
INBOUND
OUTBOUND

LINK_TOKEN
A token identifying the Recovery Manager Link object.

REMOTE_UOW_STATUS
Optional Parameter

The status of the unit of work in the remote system.

Values for the parameter are:
BACKWARD
COLD
FORWARD
HEURISTIC_BACKWARD
HEURISTIC_FORWARD
HEURISTIC_MIXED
INDOUBT
RESET

UNKNOWN

TOLERATE_VIOLATIONS
Optional Parameter

A parameter specifying the rules to be used to detect resynchronization protocol violations.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
ALREADY_SET
LINK_UNKNOWN
RECOVERY_NOT_IN_PROGRESS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

UOW_STATUS
Optional Parameter

The status of the unit of work.

Values for the parameter are:
BACKWARD
FORWARD
HEURISTIC_BACKWARD
HEURISTIC_FORWARD
INDOUBT

RMLN gate, START_LINK_BROWSE function

This function starts a browse of Recovery Manager Link objects.

Input Parameters

CLIENT_NAME
Optional Parameter

Name of the communications protocol used on the link.

REMOTE_ACCESS_ID_BUFFER
Optional Parameter

A buffer containing the netname of the remote system, or the name of the External Resource Manager.

Output Parameters

REASON
The values for the parameter are:
CLIENT_UNKNOWN
UOW_UNKNOWN

LINK_BROWSE_TOKEN
A token to be used during a browse of all Recovery Manager Link objects for a particular Recovery Manager client.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

UOW_BROWSE_TOKEN
A token to be used during a browse of all Recovery Manager Link objects for a particular unit of work object.

RMLN gate, TERMINATE_RECOVERY function

Input Parameters

DIRECTION
Parameter specifying whether to commit (FORWARD), backout (BACKWARD) or obey the ACTION attribute in the definition of the originating transaction.

Values for the parameter are:
INBOUND
OUTBOUND

FORGET
A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged.

Values for the parameter are:
NO
YES

LINK_TOKEN
A token identifying the Recovery Manager Link object.

OPERATOR_INITIATED
A parameter specifying whether the function is the result of an explicit user action.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
LINK_UNKNOWN
RECOVERY_NOT_IN_PROGRESS
SET_NOT_DONE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMNM gate, CLEAR_PENDING function

This function is used to remove Recovery Manager Link objects associated with a specified remote system. Affected indoubt units of work will take a unilateral decision to commit or backout their resource updates.

Input Parameters

CLIENT_NAME
Name of the communications protocol used on the link.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system, or the name of the External Resource Manager.

ALL
Optional Parameter

A parameter specifying whether only Recovery Manager Link objects with the same logname as that currently associated with the remote system should be removed or all Recovery Manager Link objects.

Values for the parameter are:
NO
YES

COLD
Optional Parameter

A parameter specifying whether the remote system has a new log and so has lost recovery information with respect to units of work in this system.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
CLEAR_PENDING_IN_PROGRESS
NOT_FOUND
UNKNOWN_CLIENT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMNM gate, INQUIRE_LOGNAME function

This function returns the logname and data associated with the specified remote system being communicated with via the specified Recovery Manager client.

Input Parameters

LOGNAME_BUFFER
An optional parameter specifying a buffer containing the logname of the remote system.

CLIENT_NAME
Optional Parameter

Name of the communications protocol used on the link.

REMOTE_ACCESS_ID_BUFFER
Optional Parameter

A buffer containing the netname of the remote system, or the name of the External Resource Manager.

RMC_DATA_BUFFER
Optional Parameter

A buffer to be used to return data owned by the Recovery Manager client.

Output Parameters

REASON
The values for the parameter are:
NOT_FOUND
UNKNOWN_CLIENT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

IN_USE
Optional Parameter

Whether there are any Recovery Manager Link object in the system associated with the logname.

Values for the parameter are:

NO
YES

RMNM gate, SET_LOGNAME function

This function is used to associate a logname and some data with the netname of a remote system for a specified Recovery Manager client.

Input Parameters

CLIENT_NAME
Name of the communications protocol used on the link.

LOGNAME_BUFFER
An optional parameter specifying a buffer containing the logname of the remote system.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system, or the name of the External Resource Manager.

RMC_DATA_BUFFER
Optional Parameter

A buffer to be used to return data owned by the Recovery Manager client.

Output Parameters

REASON
The values for the parameter are:
UNKNOWN_CLIENT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMOT gate, COMMIT function

Commit an Open Transaction Environment (OTE) transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
UOW_ROLLEDBACK

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMOT gate, PREPARE function

Prepare an Open Transaction Environment (OTE) transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
INVALID_VOTE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

VOTE
The vote from the OTE transaction.

Values for the parameter are:
HEURISTIC_MIXED
NO
READ_ONLY

YES

RMOT gate, ROLLBACK function

Roll back an Open Transaction Environment (OTE) transaction.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

UOW_COMMITTED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMOT gate, SET_OTS_UOW function

Set the properties of an Open Transaction Environment (OTE) transaction.

Input Parameters

BQUAL_LEN
FORMAT_ID
LOGICAL_SERVER
PUBLIC_ID
TID_BLOCK_IN

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMRE gate, APPEND function

This function writes data to the system log. The data written is associated with the current unit of work of the currently executing transaction if either FORWARD_DATA(YES) or BACKWARD_DATA(YES) is specified.

Input Parameters

BACKWARD_DATA

A parameter specifying whether the data is used for backward recovery purposes.

Values for the parameter are:

NO
YES

CLIENT_NAME

Name of the communications protocol used on the link.

Values for the parameter are:

APAL
APIC
APRD
APSP
APUS
BAM
BR
DH
EJ
FC
FCCF
IND

IRC
IRCO
LGGL
LT
LU61
LU62
NQ
OT
OTDM
PIRS
RMI
RMIO
RUNT
RZ
SH
TDTR
TS
XFFR

DATA

Address of an extended Iliffe vector. An extended Iliffe vector consists of a linked list of at least one element. Each element of the linked list consists of a variable length array of address length pairs. Each address and length field is four bytes long. The top bit of each address is off except for the last which may be on.

If an address is binary zero, then this terminates the element and the linked list.

If an address has the top bit on, then it terminates the element and points to the next element in the linked list.

An extended Iliffe vector represents the block of data formed by concatenating all the blocks which are pointed to by address length pairs in the vector which have the address top bit off. The order is from front to back of the linked list and from low to high index within each array.

FORCE_DATA

A parameter specifying whether the data is forced out on to the non-volatile log or can merely be written to the volatile log buffer.

Values for the parameter are:

NO
YES

FORWARD_DATA

A parameter specifying whether the data is used for forward recovery purposes.

Values for the parameter are:

NO
YES

LOG_BUFFER_SUSPEND

Optional Parameter

A binary value specifying whether the caller can tolerate the task suspending to wait for space in a log buffer.

Values for the parameter are:

NO
YES

RAISE_INV_DATA_LENGTH
Optional Parameter

An optional parameter specifying whether the caller wants to be informed of there being to much data to be logged.

Values for the parameter are:
NO
YES

REMARK
Optional Parameter

An optional parameter for the benefit of trace to describe the data being logged.

RESOURCE_ID
Optional Parameter

A parameter specifying the name of the resource with which the data to be logged is associated.

Output Parameters

REASON
The values for the parameter are:
INSUFFICIENT_BUFFER_SPACE
INVALID_CLIENT_NAME
INVALID_DATA_LENGTH
INVALID_RESOURCE_ID
NO_DATA

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FORCE_TOKEN
Optional Parameter

A token that can be used to force the data on to the non-volatile log with the FORCE function of the RMRE gate.

RMRE gate, AVAIL function

This function informs Recovery Manager that a local resource has become available. It is used when either a backout failure or a commit failure has previously occurred and the resource (or reason for the failure) has now cleared - or there is now reason to believe it may have cleared.

Input Parameters

CLIENT_NAME
Name of the communications protocol used on the link.

Values for the parameter are:
APAL
APIC
APRD
APSP
APUS
BAM
BR
DH
EJ
FC
FCCF

IND
IRC
IRCO
LGGL
LT
LU61
LU62
NQ
OT
OTDM
PIRS
RMI
RMIO
RUNT
RZ
SH
TDTR
TS
XFFR

LOCAL_ACCESS_ID
An optional parameter specifying a buffer in which the local access ID of the resource causing the unit of work to shunt will be returned.

GENERIC
Optional Parameter

A binary value indicating if the local access ID is generic.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
LOCAL_ACCESS_ID_UNKNOWN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMRE gate, FORCE function

This function forces data written previously to a log buffer to the non-volatile log.

Input Parameters

FORCE_TOKEN
A token returned on a previous call to the APPEND function of the RMRE gate.

Output Parameters

REASON
The values for the parameter are:
INSUFFICIENT_BUFFER_SPACE
INVALID_CLIENT_NAME
INVALID_DATA_LENGTH
INVALID_LOCAL_ACCESS_ID
INVALID_RESOURCE_ID
LOCAL_ACCESS_ID_UNKNOWN
NO_DATA

UOW_NOT_BACKWARDS
UOW_NOT_SHUNTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMRE gate, KEYPOINT_DATA function

Record keypoint data on the system log.

Input Parameters

CLIENT_NAME
Name of the communications protocol used on the link.

Values for the parameter are:
APAL
APIC
APRD
APSP
APUS
BAM
BR
DH
EJ
FC
FCCF
IND
IRC
IRCO
LGGL
LT
LU61
LU62
NQ
OT
OTDM
PIRS
RMI
RMIO
RUNT
RZ
SH
TDTR
TS
XFFR

DATA

Address of an extended Iliffe vector. An extended Iliffe vector consists of a linked list of at least one element. Each element of the linked list consists of a variable length array of address length pairs. Each address and length field is four bytes long. The top bit of each address is off except for the last which may be on.

If an address is binary zero, then this terminates the element and the linked list.

If an address has the top bit on, then it terminates the element and points to the next element in the linked list.

An extended Iliffe vector represents the block of data formed by concatenating all the blocks which are pointed to by address length pairs in the vector which have the address top bit off. The order is from front to back of the linked list and from low to high index within each array.

RAISE_INV_DATA_LENGTH
Optional Parameter

An optional parameter specifying whether the caller wants to be informed of there being to much data to be logged.

Values for the parameter are:
NO
YES

REMARK
Optional Parameter

An optional parameter for the benefit of trace to describe the data being logged.

Output Parameters

REASON
The values for the parameter are:
INVALID_CLIENT_NAME
INVALID_DATA_LENGTH
NO_DATA

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMRE gate, REMOVE function

This function removes data logged by a Recovery Manager client and associated with a particular local resource from a unit of work.

Input Parameters

CLIENT_NAME
Name of the communications protocol used on the link.

Values for the parameter are:
APAL
APIC
APRD
APSP
APUS
BAM
BR
DH
EJ
FC
FCCF
IND
IRC
IRCO
LGGL
LT
LU61
LU62
NQ
OT

OTDM
PIRS
RMI
RMIO
RUNT
RZ
SH
TDTR
TS
XFFR

LOCAL_ACCESS_ID
An optional parameter specifying a buffer in which the local access id of resource causing the unit of work to shunt will be returned.

LOCAL_UOW_ID
The local UOWID of the required unit of work.

UOW_ID
An optional parameter specifying the network UOWID to be given to the unit of work object. This parameter will be present if the unit of work being created is part of a distributed unit of work that originated on another system.

Output Parameters

REASON
The values for the parameter are:
INVALID_CLIENT_NAME
INVALID_LOCAL_ACCESS_ID
UOW_NOT_BACKWARDS
UOW_NOT_SHUNTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMRE gate, REQUEST_FORGET function

This function associates a Recovery Manager client and a named local resource with a requirement to engage in forget processing.

Input Parameters

CLIENT_NAME
Name of the communications protocol used on the link.

Values for the parameter are:
APAL
APIC
APRD
APSP
APUS
BAM
BR
DH
EJ
FC
FCCF
IND
IRC
IRCO
LGGL
LT
LU61

LU62
NQ
OT
OTDM
PIRS
RMI
RMIO
RUNT
RZ
SH
TDTR
TS
XFFR

LOCAL_ACCESS_ID
An optional parameter specifying a buffer in which the local access id of resource causing the unit of work to shunt will be returned.

LOG_NEEDED
Optional Parameter

Binary value that specifies whether the information is to be recorded in the system log, for recovery at emergency restart.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
INVALID_CLIENT_NAME
INVALID_LOCAL_ACCESS_ID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMSL gate, TAKE_ACTIVITY_KEYPOINT function

This function performs the activity associated with taking a keypoint.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMUW gate, BACKOUT_UOW function

This function causes the changes in a unit of work to be backed out.

Input Parameters

CONTINUE
Is the task continuing into a following, new unit of work.

Values for the parameter are:
NO
YES

RESTART
Optional Parameter

This parameter is only applicable when CONTINUE(NO) is specified and indicates whether or not transaction restart will be performed.

Values for the parameter are:

NO
YES

Output Parameters

REASON

The values for the parameter are:

BACKOUT_FAILURE
COMMIT_FAILURE
REMOTE_COMMIT_ABENDED
ROLLBACK_NOT_SUPPORTED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMUW gate, BIND_UOW_TO_TXN function

Make the specified unit of work the current unit of work for the current transaction.

Input Parameters

UOW_TOKEN

An optional parameter specifying a token used to identify the unit of work object being queried.

Output Parameters

REASON

The values for the parameter are:

BACKOUT_FAILURE
BROWSE_END
COMMIT_FAILURE
HEURISTIC_BACKOUT
HEURISTIC_COMMIT
HEURISTIC_READONLY_BACKOUT
HEURISTIC_READONLY_COMMIT
INDOUBT_FAILURE
INVALID_BROWSE_TOKEN
LINKS_INVALID
LOCAL_NO_MARKED
LOCAL_NO_VOTE
NOT_FOUND
NOT_SHUNTED
REMOTE_COMMIT_ABENDED
REMOTE_NO_DECISION
REMOTE_NO_VOTE
RESYNCH_IN_PROGRESS
ROLLBACK
ROLLBACK_NOT_SUPPORTED
UOW_NOT_INDOUBT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMUW gate, COMMIT_UOW function

This function attempts to commit the changes made in a unit of work.

Input Parameters

CONTINUE

Is the task continuing into a following, new unit of work.

Values for the parameter are:

NO
YES

Output Parameters

REASON

The values for the parameter are:

COMMIT_FAILURE
HEURISTIC_BACKOUT
HEURISTIC_COMMIT
HEURISTIC_READONLY_BACKOUT
HEURISTIC_READONLY_COMMIT
INDOUBT_FAILURE
LINKS_INVALID
LOCAL_NO_MARKED
LOCAL_NO_VOTE
REMOTE_COMMIT_ABENDED
REMOTE_NO_DECISION
REMOTE_NO_VOTE
ROLLBACK

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMUW gate, CREATE_NETWORK_UOWID function

Generate a unit-of-work ID (UOWID).

Input Parameters

UOW_ID

A block in which the generated UOWID is returned.

Output Parameters

REASON

The values for the parameter are:

BACKOUT_FAILURE
BROWSE_END
COMMIT_FAILURE
HEURISTIC_BACKOUT
HEURISTIC_COMMIT
HEURISTIC_READONLY_BACKOUT
HEURISTIC_READONLY_COMMIT
INDOUBT_FAILURE
INVALID_BROWSE_TOKEN
LINKS_INVALID
LOCAL_NO_MARKED
LOCAL_NO_VOTE
NOT_FOUND
NOT_SHUNTED
REMOTE_COMMIT_ABENDED
REMOTE_NO_DECISION
REMOTE_NO_VOTE
RESYNCH_IN_PROGRESS
ROLLBACK

ROLLBACK_NOT_SUPPORTED
UOW_NOT_INDOUBT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMUW gate, CREATE_UOW function

Create a unit of work object under the currently executing transaction.

Input Parameters

CHOICE
Optional Parameter

Specifies whether the unit of work should commit or backout if requested to take a unilateral decision.

Values for the parameter are:
BACKWARD
FORWARD

HEURISM
Optional Parameter

Specifies whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window.

Values for the parameter are:
NO
YES

INDOUBT_TIMEOUT_INTERVAL
Optional Parameter

The period of time that the unit of work should be prepared to wait in doubt.

UOW_ID
Optional Parameter

The network UOWID to be given to the unit of work object. This parameter will be present if the unit of work being created is part of a distributed unit of work that originated on another system.

USERID
Optional Parameter

The userid associated with the currently executing transaction.

Output Parameters

REASON
The values for the parameter are:
BACKOUT_FAILURE
BROWSE_END
COMMIT_FAILURE
HEURISTIC_BACKOUT
HEURISTIC_COMMIT
HEURISTIC_READONLY_BACKOUT
HEURISTIC_READONLY_COMMIT
INDOUBT_FAILURE
INVALID_BROWSE_TOKEN
LINKS_INVALID
LOCAL_NO_MARKED
LOCAL_NO_VOTE
NOT_FOUND
NOT_SHUNTED

REMOTE_COMMIT_ABENDED
REMOTE_NO_DECISION
REMOTE_NO_VOTE
RESYNCH_IN_PROGRESS
ROLLBACK
ROLLBACK_NOT_SUPPORTED
UOW_NOT_INDOUBT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMUW gate, END_UOW_BROWSE function

This function is used at the end of a browse of the unit of work objects in the system.

Input Parameters

BROWSE_TOKEN
A token obtained from a previous START_UOW_BROWSE call.

Output Parameters

REASON
The values for the parameter are:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMUW gate, FORCE_UOW function

This function forces an in doubt unit of work to unilaterally commit or backout its changes rather than continue waiting for resynchronization with the coordinating system.

Input Parameters

UOW_TOKEN
An optional parameter specifying a token used to identify the unit of work object being queried.

DIRECTION
Optional Parameter

Parameter specifying whether to commit (FORWARD), backout (BACKWARD) or obey the ACTION attribute in the definition of the originating transaction.

Values for the parameter are:
BACKWARD
FORWARD
HEURISTIC

HEURISTIC_CAUSE
Optional Parameter

An indication of the reason a unilateral decision must be taken.

Values for the parameter are:
OPERATOR
OTHER_CAUSE
TIMEOUT

Output Parameters

REASON

The values for the parameter are:

```
NOT_FOUND
NOT_SHUNTED
RESYNCH_IN_PROGRESS
UOW_NOT_INDOUBT
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMUW gate, GET_NEXT_UOW function

This function returns information about the next unit of work object in the browse.

Input Parameters

BROWSE_TOKEN

A token obtained from a previous START_UOW_BROWSE call.

LINK_ID

Optional Parameter

An optional parameter specifying a buffer in which the termid of the link to the coordinating system will be returned.

LOCAL_ACCESS_ID

Optional Parameter

An optional parameter specifying a buffer in which the local access id of resource causing the unit of work to shunt will be returned.

LOGNAME

Optional Parameter

An optional parameter specifying a buffer in which the log name of the coordinating system will be returned.

OTS_TID

Optional Parameter

REMOTE_ACCESS_ID

Optional Parameter

An optional parameter specifying a buffer in which the netname of coordinating system will be returned.

UOW_ID

Optional Parameter

An optional parameter specifying the network UOWID to be given to the unit of work object. This parameter will be present if the unit of work being created is part of a distributed unit of work that originated on another system.

Output Parameters

REASON

The values for the parameter are:

```
BROWSE_END
INVALID_BROWSE_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACCESS_ID_TYPE

Optional Parameter

The type of resource that has caused the unit of work to shunt.

Values for the parameter are:

LOCAL
REMOTE

AWAITING_FORGET
Optional Parameter

The unit of work might have completed syncpoint processing, and be merely waiting for confirmation that subordinates have completed theirs.

Values for the parameter are:
NO
YES

CHOICE
Optional Parameter

The choice of whether the unit of work should commit or backout if requested to take a unilateral decision.

Values for the parameter are:
BACKWARD
FORWARD

CLIENT_NAME
Optional Parameter

The name of the Recovery Manager client that owns the resource that has caused the unit of work to shunt.

CREATION_TIME
Optional Parameter

An 8 byte Store Clock representation of the time the unit of work was created.

DURATION
Optional Parameter

An 8 byte Store Clock representation of the time the unit of work changed state.

FIRST_UOW_FOR_TXN
Optional Parameter

A binary value that indicates whether this is the first unit of work in the CICS transaction.

Values for the parameter are:
NO
YES

HEURISM
Optional Parameter

Whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window.

Values for the parameter are:
NO
YES

LOCAL_UOW_ID
Optional Parameter

An optional parameter to receive the local UOWID.

OP_ID
Optional Parameter

The Operator Id associated with the task that created the unit of work.

OUT_UOW_TOKEN
Optional Parameter

The token used to identify the unit of work object.

SHUNTED

Optional Parameter

The unit of work may or may not be shunted.

Values for the parameter are:

```
NO
YES
```

TERMID

Optional Parameter

The termid associated with the task that created the unit of work object.

TERMINAL_LUNAME

Optional Parameter

The terminal LU name associated with the task that created the unit of work object.

TRANID

Optional Parameter

The tranid of the task that created the unit of work object.

TRANNUM

Optional Parameter

The task number of the task that created the unit of work.

UOW_STATUS

Optional Parameter

The status of the unit of work.

Values for the parameter are:

```
BACKWARD
FORWARD
HEURISTIC_BACKWARD
HEURISTIC_FORWARD
IN_DOUBT
IN_FLIGHT
```

USERID

Optional Parameter

The userid associated with the task that created the unit of work object.

RMUW gate, INQUIRE_UOW function

This function is used to query information about a particular unit of work.

Input Parameters

LINK_ID

Optional Parameter

A buffer in which the termid of the link to the coordinating system will be returned.

LOCAL_ACCESS_ID

Optional Parameter

A buffer in which the local access id of resource causing the unit of work to shunt will be returned.

LOG_CHAIN_TOKEN

Optional Parameter

A token that identifies the log chain whose unit of work object is to be queried.

LOGNAME
Optional Parameter

A buffer in which the log name of the coordinating system will be returned.

OTS_TID
Optional Parameter

The Open Transaction Environment (OTE) identifier of the unit of work.

REMOTE_ACCESS_ID
Optional Parameter

A buffer in which the netname of coordinating system will be returned.

TRANSACTION_TOKEN
Optional Parameter

A token that identifies the transaction whose unit of work object is to be queried.

UOW_ID
Optional Parameter

A buffer in which the network UOWID will be returned.

UOW_TOKEN
Optional Parameter

A token that identifies the unit of work object being queried.

Output Parameters

REASON
The values for the parameter are:

```
NOT_FOUND
```

RESPONSE
The domian's response to the call.

Values for the parameter are:

```
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED
```

ACCESS_ID_TYPE
Optional Parameter

The type of resource that has caused the unit of work to shunt.

Values for the parameter are:

```
LOCAL
REMOTE
```

AWAITING_FORGET
Optional Parameter

Indicates that the unit of work has completed syncpoint processing, and is just waiting for confirmation that subordinates have completed theirs.

Values for the parameter are:

```
NO
YES
```

CHOICE
Optional Parameter

The choice that has been made as to whether the unit of work should commit or backout if requested to take a unilateral decision.

Values for the parameter are:
BACKWARD
FORWARD

CLIENT_NAME
Optional Parameter

The name of the Recovery Manager client that owns the resource that has caused the unit of work to shunt.

CREATION_TIME
Optional Parameter

An 8 byte Store Clock representation of the time the unit of work was created.

DURATION
Optional Parameter

An 8 byte Store Clock representation of the time the unit of work changed state.

FIRST_UOW_FOR_TXN
Optional Parameter

A binary value indicating if this is the first unit of work for the transaction.

Values for the parameter are:
NO
YES

HEURISM
Optional Parameter

Binary value indicating whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window.

Values for the parameter are:
NO
YES

LOCAL_UOW_ID
Optional Parameter

The local unit of work id.

OP_ID
Optional Parameter

The Operator Id associated with the task that created the unit of work.

OUT_UOW_TOKEN
Optional Parameter

The token used to identify the unit of work object.

SHUNTED
Optional Parameter

A binary value indicating if the unit of work has been shunted.

Values for the parameter are:
NO
YES

TERMID
Optional Parameter

The termid associated with the task that created the unit of work object.

TERMINAL_LUNAME
Optional Parameter

The terminal LU name associated with the task that created the unit of work object.

TRANID
Optional Parameter

The tranid of the task that created the unit of work object.

TRANNUM
Optional Parameter

The transaction number of the task that created the unit of work.

UOW_STATUS
Optional Parameter

The status of the unit of work.

Values for the parameter are:

```
BACKWARD
FORWARD
HEURISTIC_BACKWARD
HEURISTIC_FORWARD
IN_DOUBT
IN_FLIGHT
```

USERID
Optional Parameter

The userid associated with the task that created the unit of work object.

RMUW gate, INQUIRE_UOW_ID function

Return the network and local UOWIDs of the unit of work of the currently executing transaction.

Input Parameters

UOW_ID
Optional Parameter

An optional parameter specifying the network UOWID to be given to the unit of work object. This parameter will be present if the unit of work being created is part of a distributed unit of work that originated on another system.

Output Parameters

REASON
The values for the parameter are:

```
BACKOUT_FAILURE
BROWSE_END
COMMIT_FAILURE
HEURISTIC_BACKOUT
HEURISTIC_COMMIT
HEURISTIC_READONLY_BACKOUT
HEURISTIC_READONLY_COMMIT
INDOUBT_FAILURE
INVALID_BROWSE_TOKEN
LINKS_INVALID
LOCAL_NO_MARKED
LOCAL_NO_VOTE
NOT_FOUND
NOT_SHUNTED
REMOTE_COMMIT_ABENDED
REMOTE_NO_DECISION
REMOTE_NO_VOTE
RESYNCH_IN_PROGRESS
ROLLBACK
```

ROLLBACK_NOT_SUPPORTED
UOW_NOT_INDOUBT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LOCAL_UOW_ID
Optional Parameter

An optional parameter to receive the local UOWID.

RMUW gate, INQUIRE_UOW_TOKEN function

Return the token identifying the unit of work object with the specified local UOWID.

Input Parameters

LOCAL_UOW_ID
The local UOWID of the required unit of work.

Output Parameters

REASON
The values for the parameter are:
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

UOW_TOKEN
A token identifying the unit of work object.

RMUW gate, REATTACH_REPLY function

This function gives control to Recovery Manager to do its unshunt processing under a re-attached transaction.

Input Parameters

UOW_TOKEN
An optional parameter specifying a token used to identify the unit of work object being queried.

Output Parameters

REASON
The values for the parameter are:
BACKOUT_FAILURE
BROWSE_END
COMMIT_FAILURE
HEURISTIC_BACKOUT
HEURISTIC_COMMIT
HEURISTIC_READONLY_BACKOUT
HEURISTIC_READONLY_COMMIT
INDOUBT_FAILURE
INVALID_BROWSE_TOKEN
LINKS_INVALID
LOCAL_NO_MARKED
LOCAL_NO_VOTE
NOT_FOUND
NOT_SHUNTED
REMOTE_COMMIT_ABENDED
REMOTE_NO_DECISION
REMOTE_NO_VOTE

RESYNCH_IN_PROGRESS
ROLLBACK
ROLLBACK_NOT_SUPPORTED
UOW_NOT_INDOUBT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMUW gate, SET_UOW function

This function is used to set characteristics of the currently executing unit of work.

Input Parameters

HEURISM
Optional Parameter

An optional parameter specifying whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window.

Values for the parameter are:
YES

HEURISTIC_CAUSE
Optional Parameter

An indication of the reason a unilateral decision must be taken.

Values for the parameter are:
LU61_CLIENT
MRO_CLIENT
OTHER_CLIENT
RMI_CLIENT
TD_CLIENT

Output Parameters

REASON
The values for the parameter are:
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMUW gate, START_UOW_BROWSE function

This function is used to start a browse of unit of work objects in the system.

Input Parameters

SHUNTED
Optional Parameter

The browse can be of only shunted units of work, only non-shunted units of work or all units of work.

Values for the parameter are:
BOTH
NO
YES

Output Parameters

REASON
The values for the parameter are:
NOT_FOUND

BROWSE_TOKEN
A token to be used on subsequent GET_NEXT_UOW calls.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMWT gate, END_WORK_TOKEN_BROWSE function

This function is used at the end of a browse of the work token objects in the system.

Input Parameters

BROWSE_TOKEN
A token obtained from a previous START_WORK_TOKEN_BROWSE call.

Output Parameters

REASON
The values for the parameter are:
`INVALID_BROWSE_TOKEN`
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMWT gate, GET_NEXT_WORK_TOKEN function

This function returns information about the next work token object in the browse.

Input Parameters

BROWSE_TOKEN
A token obtained from a previous START_WORK_TOKEN_BROWSE call.

Output Parameters

REASON
The values for the parameter are:
`BROWSE_END`
`INVALID_BROWSE_TOKEN`
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
WORK_TOKEN | WORK_TOKEN_64
WORK_TOKEN
The work token returned by the browse operation.
WORK_TOKEN_64
The 64-bit work token returned by the browse operation.
LOCAL_UOW_ID
Optional Parameter

The local unit of work identifier for the unit of work associated with the work token.
UOW_TOKEN
Optional Parameter

The token for the unit of work associated with the work token.

RMWT gate, INQUIRE_WORK_TOKEN function

Retrieve the work token that is associated with a client in a unit of work.

Input Parameters

CLIENT_NAME

The name of the client that is associated with the work token.

Values for the parameter are:

```
APAL
APIC
APRD
APSP
APUS
BAM
BR
DH
EJ
FC
FCCF
IND
IRC
IRCO
LGGL
LT
LU61
LU62
NQ
OT
OTDM
PIRS
RMI
RMIO
RUNT
RZ
SH
TDTR
TS
XFFR
```

UOW_TOKEN

Optional Parameter

A token that identifies the unit of work. If this parameter is omitted, the request is made against the current unit of work.

Output Parameters

REASON

The values for the parameter are:

```
NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WORK_TOKEN | WORK_TOKEN_64

WORK_TOKEN

The work token.

WORK_TOKEN_64

The 64-bit work token.

RMWT gate, SET_WORK_TOKEN function

Pass a work token to recovery manager, denoting the interest of a client in the current unit of work.

Input Parameters

CLIENT_NAME

The name of the client that is associated with the work token.

Values for the parameter are:

APAL
APIC
APRD
APSP
APUS
BAM
BR
DH
EJ
FC
FCCF
IND
IRC
IRCO
LGGL
LT
LU61
LU62
NQ
OT
OTDM
PIRS
RMI
RMIO
RUNT
RZ
SH
TDTR
TS
XFFR

WORK_TOKEN | WORK_TOKEN_64

WORK_TOKEN

The work token.

WORK_TOKEN_64

The 64-bit work token.

RMWT gate, START_WORK_TOKEN_BROWSE function

Start a browse operation on the work tokens associated with a client.

Input Parameters

CLIENT_NAME

The name of the client that is associated with the work token.

Values for the parameter are:

APAL
APIC
APRD
APSP
APUS
BAM
BR
DH

EJ
FC
FCCF
IND
IRC
IRCO
LGGL
LT
LU61
LU62
NQ
OT
OTDM
PIRS
RMI
RMIO
RUNT
RZ
SH
TDTR
TS
XFFR

Output Parameters

REASON
The values for the parameter are:
NOT_FOUND

BROWSE_TOKEN
A token that identifies the browse operation.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Recovery manager domain's callback formats

Table 41 describes the callback formats owned by the domain and shows the functions performed on the calls.

Table 41. Recovery manager domain's callback formats

Format	Calling module	Function
RMRO	DFHRMUO	PERFORM_COMMIT
	DFHRMUP	
	DFHRMUQ	
	DFHRMUW	
	DFHRMUO	PERFORM_PREPARE
	DFHRMRO2	START_BACKOUT
	DFHRMRO3	DELIVER_BACKOUT_DATA
	DFHRMRO4	END_BACKOUT
	DFHRMROS	PERFORM_SHUNT
	DFHRMROU	PERFORM_UNSHUNT
RMDE	DFHRMR1S	START_DELIVERY
	DFHRMR1D	DELIVER_RECOVERY
	DFHRMR1E	END_DELIVERY
	DFHRMR1D	DELIVER_FORGET
RMKP	DFHRMR1K	TAKE_KEYPOINT

Table 41. Recovery manager domain's callback formats (continued)

Format	Calling module	Function
RMLK	DFHRMLSP	PERFORM_PRELOGGING
	DFHRMLSP	PERFORM_PREPARE
	DFHRMLSD	REPLY_DO_COMMIT
	DFHRMLSD	SEND_DO_COMMIT
	DFHRMLSO	PERFORM_COMMIT
	DFHRMLSS	PERFORM_SHUNT
	DFHRMLSU	PERFORM_UNSHUNT

Note: In the descriptions of the formats, the input parameters are input not to the Recovery manager domain, but to the domain being called by the Recovery manager domain. Similarly, the output parameters are output by the domain that was called by the Recovery manager domain, in response to the call.

RMRO gate, DELIVER_BACKOUT_DATA function

This function requires the Recovery Manager client process backout data from the system log for the unit of work.

Input Parameters

WORK_TOKEN
The Recovery Manager client's work token for the syncpointing unit of work.

DATA
A buffer containing the data previously logged with BACKWARD_DATA(YES) via the APPEND function of the RMRE gate.

RESOURCE_ID
Optional parameter.

The name of the resource with which the logged data is associated.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:
NO
YES

FORWARD_DATA
A parameter specifying whether or not the data was originally logged as FORWARD_DATA.

The values for the parameter are:
NO
YES

REMOVE
A parameter specifying whether or not the backout is due to an invocation of the REMOVE function of the RMRE gate.

The values for the parameter are:
NO
YES

CLUSTER_ID
A buffer to receive a symbolic name identifying the resource.

LOCAL_ACCESS_ID
A buffer to receive the specific name of the resource

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

KEEP
A value specifying whether the backout action failed, implying the record should be kept and not forgotten.

The values for the parameter are:
NO
YES

RMRO gate, END_BACKOUT function

This function notifies the Recovery Manager client that backout processing has completed for the unit of work.

Input Parameters

WORK_TOKEN
The Recovery Manager client's work token for the syncpointing unit of work.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:
NO
YES

REMOVE
A parameter specifying whether or not the backout is due to an invocation of the REMOVE function of the RMRE gate.

The values for the parameter are:
NO
YES

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMRO gate, PERFORM_COMMIT function

This function requires the Recovery Manager client to perform phase two of syncpoint processing.

Input Parameters

WORK_TOKEN
The Recovery Manager client's work token for the syncpointing unit of work.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:
NO
YES

UOW_STATUS
The status of the current unit of work.

The values for the parameter are:
BACKWARD
FORWARD

RESTART
Optional parameter

Specifies whether a backing out transaction will be restarted.

The values for the parameter are:
NO
YES

Output Parameters
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
FORGET_RECORD
A value specifying whether all obligations to this Recovery Manager client have been discharged.

The values for the parameter are:
NO
YES

RMRO gate, PERFORM_PREPARE function
This function requires the Recovery Manager client to perform phase one of syncpoint processing.

Input Parameters
WORK_TOKEN
The Recovery Manager client's work token for the syncpointing unit of work.
CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:
NO
YES

Output Parameters
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
VOTE
A value specifying the Recovery Manager client's vote on the outcome of the syncpointing unit of work.

It can have any one of these values: YES | NO | NO_CONTINUE | READ_ONLY

The values for the parameter are:
NO
NO_CONTINUE
READ_ONLY
YES

RMRO gate, PERFORM_SHUNT function
This function notifies the Recovery Manager client that the unit of work is about to shunt.

Input Parameters
WORK_TOKEN
The Recovery Manager client's work token for the syncpointing unit of work.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:
NO
YES

Output Parameters
NEXT_WORK_TOKEN
A value for the Recovery Manager client's work token in the following unit of work.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMRO gate, PERFORM_UNSHUNT function
This function notifies the Recovery Manager client that the unit of work is unshunting.

Input Parameters
WORK_TOKEN
The Recovery Manager client's work token for the syncpointing unit of work.

Output Parameters
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMRO gate, START_BACKOUT function
This function notifies the Recovery Manager client that backout processing is about to be performed for the unit of work.

Input Parameters
WORK_TOKEN
The Recovery Manager client's work token for the syncpointing unit of work.
CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:
NO
YES
REMOVE
A parameter specifying whether or not the backout is due to an invocation of the REMOVE function of the RMRE gate.

The values for the parameter are:
NO
YES

Output Parameters
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMDE gate, DELIVER_FORGET function

This function notifies the Recovery Manager client that FORGET processing is required for some resource in a unit of work.

Input Parameters

LOCAL_ACCESS_ID

A parameter specifying the name of the resource associated with the forget processing.

UOW

A parameter with the fixed value YES.

UOW_STATUS

The status of the unit of work.

The values for the parameter are:

FORWARD
BACKWARD
IN_DOUBT
IN_FLIGHT

LOCAL_UOW_ID

The local unit of work identifier.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMDE gate, DELIVER_RECOVERY function

This function requires the Recovery Manager client to process recovery data from the system log.

Input Parameters

RESOURCE_ID

Optional parameter

The name of the resource with which the logged data is associated.

DATA

A buffer containing the data previously logged with BACKWARD_DATA(YES) via the APPEND function of the RMRE gate.

FORWARD_DATA

A parameter specifying whether or not the data was originally logged as FORWARD_DATA.

The values for the parameter are:

NO
YES

BACKWARD_DATA

A parameter specifying whether or not the data was originally logged as BACKWARD_DATA.

The values for the parameter are:

NO
YES

KEYPOINT

A parameter specifying whether or not the data was logged as part of a keypoint.

The values for the parameter are:

NO

YES

BACKED_OUT
A parameter specifying whether or not the data is associated with backed out.

UOW
A parameter specifying whether the data is related to a particular unit of work.

The values for the parameter are:
NO
YES

UOW_STATUS
Optional parameter

Specifies the status of unit of work the data belongs to (if any).

The values for the parameter are:
FORWARD
BACKWARD
IN_DOUBT
IN_FLIGHT

LOCAL_UOW_ID
Optional parameter

Specifies the local UOWID of the unit of work the data belongs to (if any).

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMDE gate, END_DELIVERY function

This function notifies the Recovery Manager client that all recovery information from the system log has been processed.

Input Parameters

None

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMDE gate, START_DELIVERY function

This function notifies the Recovery Manager client that system recovery processing is about to be performed.

Input Parameters

None

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMKP gate, TAKE_KEYPOINT function

This function requires the Recovery Manager client to perform keypoint processing.

Input Parameters

SHUTDOWN

Specifies whether the keypoint is the warm keypoint taken during shutdown or an activity keypoint.

The values for the parameter are:

NO
YES

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RMLK gate, PERFORM_COMMIT function

This function requires the Recovery Manager client perform phase two of syncpoint processing.

Input Parameters

RMC_TOKEN

The Recovery Manager client's token associated with the Recovery Manager Link object.

CONTINUE

A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:

NO
YES

SINGLE_UPDATER

A parameter specifying whether the single updater optimization is being performed.

The values for the parameter are:

NO
YES

UOW_STATUS

The status of the syncpointing unit of work.

The values for the parameter are:

BACKWARD
FORWARD

RESTART

Optional parameter

Specifies whether a backing out transaction will be restarted.

The values for the parameter are:

NO
YES

COORDINATOR

A parameter specifying whether the remote system is the coordinator of the distributed unit of work.

The values for the parameter are:

NO
YES

INITIATOR

A parameter specifying whether the remote system is the initiator of the syncpoint.

The values for the parameter are:
NO
YES

PRESUMPTION
A parameter specifying whether the remote system assumes the presume abort or presume nothing protocols.

The values for the parameter are:
ABORT
NOTHING

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system.

The values for the parameter are:
NECESSARY
UNNECESSARY
SYNC_LEVEL_1

Output Parameters

RESPONSE
is the Recovery Manager domain's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ACCESSIBLE
A parameter specifying that the communications link to the remote system has failed.

The values for the parameter are:
NO
SHUNTED
YES

FORGET
A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged.

The values for the parameter are:
NO
YES

PASS
A parameter specifying whether an equivalent Recovery Manager Link object should be created in the following unit of work.

The values for the parameter are:
NO
YES

ABEND
A parameter specifying whether an abend occurred during the PERFORM_COMMIT call-back.

The values for the parameter are:
NO
YES

NEXT_RECOVERY_STATUS
A parameter specifying the initial RECOVERY_STATUS of the Recovery Manager Link object created in the following unit of work as a result of PASS(YES).

The values for the parameter are:
DEFAULT
NECESSARY
SYNC_LEVEL_1
UNNECESSARY

RMLK gate, PERFORM_PRELOGGING function

This function notifies the Recovery Manager client that phase one of syncpoint processing is about to occur.

Input Parameters

RMC_TOKEN
The Recovery Manager client's token associated with the Recovery Manager Link object.

INITIATOR
A parameter specifying whether the remote system is the initiator of the syncpoint.

The values for the parameter are:
NO
YES

COORDINATOR
A parameter specifying whether the remote system is the coordinator of the distributed unit of work.

The values for the parameter are:
NO
YES

Output Parameters

RESPONSE
is the domain's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

RMLK gate, PERFORM_PREPARE function

This function requires the Recovery Manager client perform phase one of syncpoint processing.

Input Parameters

RMC_TOKEN
The Recovery Manager client's token associated with the Recovery Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:

NO
YES

SYSTEM
A parameter specifying whether the PERFORM_PREPARE call is part of a syncpoint or the result of an **EXEC CICS ISSUE PREPARE** command.

The values for the parameter are:
NO
YES

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system.

The values for the parameter are:
NECESSARY
UNNECESSARY
SYNC_LEVEL_1

Output Parameters

RESPONSE
is the Recovery Manager domain's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

VOTE
A value specifying the Recovery Manager client's vote on the outcome of the syncpointing unit of work.

The values for the parameter are:
HEURISTIC_MIXED
NO
NO_CONTINUE
READ_ONLY
YES

RMLK gate, PERFORM_SHUNT function

This function notifies the Recovery Manager client that the unit of work is shunting. Input parameters

Input Parameters

RMC_TOKEN
The Recovery Manager client's token associated with the Recovery Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:
NO
YES

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system.

The values for the parameter are:
NECESSARY
UNNECESSARY
SYNC_LEVEL_1

Output Parameters

RESPONSE
is the Recovery Manager domain's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

FORGET
A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged.

The values for the parameter are:
NO
YES

RMLK gate, PERFORM_UNSHUNT function

This function notifies the Recovery Manager client that the unit of work is unshunting.

Input Parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object to be unshunted.

LOGNAME_BUFFER
A buffer containing the logname of the remote system.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system, or the name of the External Resource Manager.

LINK_ID_BUFFER
A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LINK_ID_SOURCE
An optional parameter specifying whether the local or remote system allocated the session.

The values for the parameter are:
LOCAL
REMOTE

Output Parameters

RESPONSE
is the Recovery Manager domain's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

RMLK gate, REPLY_DO_COMMIT function

This function requires the Recovery Manager client communicate the result of this systems phase one syncpoint processing to the coordinating system, and obtain the outcome of the distributed unit of work.

Input Parameters

RMC_TOKEN
The Recovery Manager client's token associated with the Recovery Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:
NO
YES

SINGLE_UPDATER
A parameter specifying whether the single updater optimization is being performed.

The values for the parameter are:
NO
YES

Output Parameters

RESPONSE
is the Recovery Manager domain's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ACCESSIBLE
A value specifying whether communication with the remote system failed.

The values for the parameter are:
NO
SUNTED
YES

VOTE
A value specifying the Recovery Manager client's vote on the outcome of the syncpointing unit of work.

The values for the parameter are:
HEURISTIC_MIXED
NO
NO_CONTINUE
READ_ONLY
YES

RMLK gate, SEND_DO_COMMIT function

This function requires the Recovery Manager client communicate the result of this system's phase one syncpoint processing to the last agent system, and obtain the outcome of the distributed unit of work.

Input Parameters

RMC_TOKEN
The Recovery Manager client's token associated with the Recovery Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work.

The values for the parameter are:
NO
YES

SINGLE_UPDATER
A parameter specifying whether the single updater optimization is being performed.

The values for the parameter are:
NO
YES

Output Parameters

RESPONSE
is the Recovery Manager domain's response to the call.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ACCESSIBLE
A value specifying whether communication with the remote system failed.

The values for the parameter are:
NO
SUNTED
YES

VOTE
A value specifying the Recovery Manager client's vote on the outcome of the syncpointing unit of work.

The values for the parameter are:
HEURISTIC_MIXED
NO
NO_CONTINUE
READ_ONLY
YES

Modules

Module	Function
DFHRMCD	Handles the functions of the RMCD gate.
DFHRMCD1	Initialises the Client Directory Class.
DFHRMCD2	Quiesces the Client Directory Class.
DFHRMCI2	Sets the callback gate of a Recovery Manager client.
DFHRMCI3	Waits for a registered Recovery Manager client to set its callback gate.

Module	Function
DFHRMCI4	Waits for a registered Recovery Manager client to set its callback gate and calls it with a given parameter list.
DFHRMDM	Recovery Manager domain initialization and termination. Handles the DMDM and RMDM gate functions.
DFHRMDU0	Formats the Recovery Manager control blocks.
DFHRMDU2	Starts a browse of all Recovery Manager client work tokens during dump formatting.
DFHRMDU3	Gets the next Recovery Manager client work token during dump formatting.
DFHRMDU4	Ends a browse of all Recovery Manager client work tokens during dump formatting.
DFHRMLKQ	Quiesces the Recovery Manager Link Class.
DFHRMLK1	Initialises the Recovery Manager Link Class.
DFHRMLK2	Handles the INITIATE_RECOVERY function of the RMLN gate.
DFHRMLK3	Inquires whether a Logname is in-use by any Recovery Manager Link.
DFHRMLK4	Handles the CLEAR_PENDING function for a particular Recovery Manager Link.
DFHRMLK5	Collects statistics from the Recovery Manager Link Class.
DFHRMLN	Handles the functions of the RMLN gate.
DFHRMLSD	Asks the coordinator Recovery Manager Link to decide the outcome of the unit of work.
DFHRMLSF	Determines the reason for a unit of work being in doubt.
DFHRMLSO	Commits the Recovery Manager Links for a unit of work.
DFHRMLSP	Prepares the Recovery Manager Links for a unit of work.
DFHRMLSS	Shunts the Recovery Manager Links for a unit of work.
DFHRMLSU	Unshunts the Recovery Manager Links for a unit of work.
DFHRML1D	Reconstructs Recovery Manager Links from log records.
DFHRMNM	Handles the functions of the RMNM gate.
DFHRMNM1	Initialises the Recovery Manager Lognames Class.
DFHRMNS1	Initialises the Recovery Manager Logname Set Class.
DFHRMNS2	Quiesces the Recovery Manager Logname Set Class.
DFHRMOFI	Initialises a Recovery Manager Object Factory.
DFHRMRO	Handles the functions of the RMRO gate.
DFHRMROO	Handles FORGET processing for Recovery Manager Resource Owners.
DFHRMROS	Shunts a Recovery Manager Resource Owner.
DFHRMROU	Unshunts a Recovery Manager Resource Owner.
DFHRMROV	Handles AVAIL processing for Recovery Manager Resource Owners.
DFHRMRO1	Initialises the Recovery Manager Resource Owner Class.
DFHRMRO2	Signals start_backout to a Recovery Manager Resource Owner.
DFHRMRO3	Delivers backout data to a Recovery Manager Resource Owner.
DFHRMRO4	Signals end_backout to a Recovery Manager Resource Owner.
DFHRMR1D	Delivers recovery data to a Recovery Manager Resource Owner.
DFHRMR1E	Signals end of recovery to a Recovery Manager Resource Owner.
DFHRMR1K	Signals a keypoint to a Recovery Manager Resource Owner.
DFHRMR1S	Signals start of recovery to a Recovery Manager Resource Owner.
DFHRMSL	Handles the functions of the RMSL gate.

Module	Function
DFHRMSLF	Forces the System Log.
DFHRMSLJ	Checks for Chain independence during recovery.
DFHRMSLL	Closes a Chain on the System Log.
DFHRMSLO	Opens a Chain on the System Log.
DFHRMSLV	Moves a Chain on the System Log.
DFHRMSLW	Writes a record to a Chain on the System Log.
DFHRMSL1	Initialises the Recovery Manager System Log Class.
DFHRMSL2	Starts a browse of a Chain on the System Log.
DFHRMSL3	Reads a Record from a Chain on the System Log.
DFHRMSL4	Ends a browse of a Chain on the System Log.
DFHRMSL5	Performs restart processing for Recovery Manager System Log Class.
DFHRMSL6	Schedules keypoint activity.
DFHRMSL7	Performs keypoint processing.
DFHRMST	Handles STST functions for Recovery Manager.
DFHRMST1	Initializes the Recovery Manager Statistics Class.
DFHRMTRI	Formats Recovery Manager trace entries.
DFHRMUC	Creates a RMUW (unit of work) object.
DFHRMUO	Commits a unit of work.
DFHRMUTL	Recovery Manager batch utility.
DFHRMUW	Handles the functions of the RMUW gate.
DFHRMUWB	Handles data during backout of a unit of work.
DFHRMUWE	Handles activities when a unit of work is unshunted.
DFHRMUWF	Forces log records for a unit of work.
DFHRMUWH	Holds an RMUW object.
DFHRMUWJ	Forces a unit of work to take a unilateral decision.
DFHRMUWL	Handles notification that all remote resources have finished processing.
DFHRMUWN	Schedules a unit of work to be unshunted.
DFHRMUWP	Handles notification that a local resource has become available.
DFHRMUWQ	Handles commit or backout of an unshunted, in doubt unit of work.
DFHRMUWS	Records the outcome of a unit of work during resynchronization.
DFHRMUWU	Records the local LU name.
DFHRMUWV	Handles notification that a local resource has become available.
DFHRMUWW	Writes a record belonging to a unit of work to the System Log.
DFHRMUW0	Releases an RMUW object.
DFHRMUW1	Initializes the Recovery Manager Unit of Work Class.
DFHRMUW2	Collects the Recovery Manager Unit of Work Class Statistics.
DFHRMUW3	Handles the INQUIRE_UOW_TOKEN function.
DFHRMU1C	Sets the Chain token for a unit of work.
DFHRMU1D	Handles log records of units of work during recovery.
DFHRMU1E	Signals that all records have been recovered from the System Log during recovery.
DFHRMU1F	Handles an in doubt wait timeout.

Module	Function
DFHRMU1J	Inquires whether all unit of work chains are disjoint.
DFHRMU1K	Keypoints a unit of work.
DFHRMU1L	Handle XMPP_FORCE_PURGE_INHIBIT_QUERY.
DFHRMU1N	Handle XMPP_FORCE_PURGE_INHIBIT_QUERY.
DFHRMU1Q	Handle the NOTIFY function of the TISR gate.
DFHRMU1R	Performs restart processing for Recovery Manager Unit of Work Class.
DFHRMU1S	Signals that recovery of log records is about to be performed.
DFHRMU1U	Process a unit of work after recovery.
DFHRMU1V	Requests time out interval notification for a unit of work.
DFHRMU1W	Cancels wait time out notification for a unit of work.
DFHRMVP1	Initializes the Recovery Manager Variable Length Subpool Class.
DFHRMXNE	Reattaches a transaction to process an unshunted unit of work.
DFHRMXN2	Schedules a keypoint.
DFHRMXN3	The keypoint program
DFHRMXN4	Restarts the Recovery Manager Transaction Class.
DFHRMXN5	Increments Recovery Manager statistics for a Transaction.

Region status domain (RS)

The region status (RS) domain captures information about the status of a region and records the status in a coupling facility data table (CFDT). Using RS domain services, other CICS regions can enquire on this status, by reading the CFDT record.

Region status domains specific gates

The specific gates provide access for other domains to functions that are provided by the RS domain.

RSDU gate, END_SYSTEM_DUMP function

The END_SYSTEM_DUMP function is called from the dump domain to record the end of a system dump (SDUMP).

Output parameters

REASON

One of the following values is returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

One of the following values is returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following value is returned when RESPONSE is PURGED:

```
TASK_CANCELLED
```

RESPONSE

Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSDU gate, END_TRANSACTION_DUMP function

The END_TRANSACTION_DUMP function is called from the dump domain to record the end of a transaction dump.

Output parameters

REASON

One of the following values is returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

One of the following values is returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following value is returned when RESPONSE is PURGED:

```
TASK_CANCELLED
```

RESPONSE

Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSDU gate, START_SYSTEM_DUMP function

The START_SYSTEM_DUMP function is called from the dump domain to record the start of a system dump (SDUMP).

Output parameters

REASON

One of the following values is returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

One of the following values is returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following value is returned when RESPONSE is PURGED:

```
TASK_CANCELLED
```

RESPONSE

Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSDU gate, START_TRANSACTION_DUMP function

The START_TRANSACTION_DUMP function is called from the dump domain to record the start of a transaction dump.

Output parameters

REASON

One of the following values is returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

One of the following values is returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following value is returned when RESPONSE is PURGED:

```
TASK_CANCELLED
```

RESPONSE
Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSSR gate, DEREGISTER_INTEREST function

DEREGISTER_INTEREST deregisters interest in a target region.

Input parameters

FILE_NAME
Specifies a 16-character file name.

POOL_NAME
Specifies an 8-character pool name.

REGION_NAME
Specifies an 8-character region name.

Output parameters

REASON
One of the following values is returned when RESPONSE is DISASTER:

```
ABEND
LOOP
LOCK_FAILURE
```

One of the following values is returned when RESPONSE is EXCEPTION:

```
RECORDING_NOT_ACTIVE
INCORRECT_POOL_NAME
TARGET_NOT_KNOWN
SERVER_FAILED
```

One of the following values is returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
RECORDING_ACTIVE
```

The following value is returned when RESPONSE is PURGED:

```
TASK_CANCELLED
```

RESPONSE
Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSSR gate, INQUIRE_TARGET_STATUS function

INQUIRE_TARGET_STATUS retrieves information about target region status.

Input parameters

FILE_NAME
Specifies a 16-character file name.

POOL_NAME
Specifies an 8-character pool name.

REGION_NAME
Specifies an 8-character region name.

<STATUS_BLOCK_TOKEN>
The token that identifies a status block where the region status is to be copied.

Output parameters

<SOS>
Specifies whether a target region is short-on-storage in the CICS environment.

The values of this parameter are:

NO
YES

<MAXTASK>

Specifies whether a target region is at maxtask within the CICS environment.

The values of this parameter are:

NO
YES

<SDUMPACTIVE>

Specifies whether a system dump is active in the target region.

The values of this parameter are:

NO
YES

<TDUMPACTIVE>

Specifies whether a transaction dump is active in the target region.

The values of this parameter are:

NO
YES

<CURRENT_TASK_COUNT>

The number of running tasks used to evaluate the load on the current routing target.

<MAX_TASK_COUNT>

The defined maximum number of active tasks that can concurrently run in the routing target.

<THRESHOLD_PERCENTAGE>

The threshold percentage of the target region, as a halfword binary value.

REASON

One of the following values is returned when RESPONSE is DISASTER:

ABEND
LOOP
LOCK_FAILURE

One of the following values is returned when RESPONSE is EXCEPTION:

RECORDING_NOT_ACTIVE
INCORRECT_POOL_NAME
TARGET_NOT_KNOWN
SERVER_FAILED

One of the following values is returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
RECORDING_ACTIVE

The following value is returned when RESPONSE is PURGED:

TASK_CANCELLED

RESPONSE

Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSSR gate, SET_THRESHOLD_PERCENTAGE function

SET_THRESHOLD_PERCENTAGE sets the threshold percentage, upper-tier percentage, and the lower-tier percentage value.

Input parameters

FILE_NAME
Specifies a 16-character file name.

POOL_NAME
Specifies an 8-character pool name.

REGION_NAME
Specifies an 8-character region name.

THRESHOLD_PERCENTAGE
Specifies the threshold percentage, as a halfword binary value. The value specified must be in the range 0 - 31.

<UPPER_TIER_PERCENTAGE>
Specifies the upper-tier percentage, as a halfword binary value. The value specified must be in the range 0 - 31.

<LOWER_TIER_PERCENTAGE>
Specifies the lower-tier percentage, as a halfword binary value. The value specified must be in the range 0 - 31.

<STATUS_BLOCK_TOKEN>
The token that identifies a status block where the region status is to be copied.

Output parameters

REASON
One of the following values is returned when RESPONSE is DISASTER:

```
ABEND
LOOP
LOCK_FAILURE
```

One of the following values is returned when RESPONSE is EXCEPTION:

```
RECORDING_NOT_ACTIVE
INCORRECT_POOL_NAME
TARGET_NOT_KNOWN
SERVER_FAILED
```

One of the following values is returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
RECORDING_ACTIVE
```

The following value is returned when RESPONSE is PURGED:

```
TASK_CANCELLED
```

RESPONSE
Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSSR gate, START_RECORDING function

START_RECORDING starts the recording of region status data into a coupling facility data table (CFDT).

Input parameters

FILE_NAME
Specifies a 16-character file name.

POOL_NAME
Specifies an 8-character pool name.

REGION_NAME
Specifies an 8-character region name.

THRESHOLD_PERCENTAGE
Specifies the threshold percentage, as a halfword binary value. The value specified must be in the range 0 - 31.

<UPPER_TIER_PERCENTAGE>
Specifies the upper-tier percentage, as a halfword binary value. The value specified must be in the range 0 - 31.

<LOWER_TIER_PERCENTAGE>
Specifies the lower-tier percentage, as a halfword binary value. The value specified must be in the range 0 - 31.

<STATUS_BLOCK_TOKEN>
The token that identifies a status block where the region status is to be copied.

<FAILURE_ECB_PTR>
The token that identifies the address of an ECB to be posted when connection to the CFDT server is lost.

Output parameters

REASON
One of the following values is returned when RESPONSE is DISASTER:

```
ABEND
LOOP
LOCK_FAILURE
```

One of the following values is returned when RESPONSE is EXCEPTION:

```
RECORDING_NOT_ACTIVE
INCORRECT_POOL_NAME
TARGET_NOT_KNOWN
SERVER_FAILED
```

One of the following values is returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
RECORDING_ACTIVE
```

The following value is returned when RESPONSE is PURGED:

```
TASK_CANCELLED
```

RESPONSE
Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSSR gate, STOP_RECORDING function

STOP_RECORDING stops the recording of region status data into a coupling facility data table (CFDT).

Input parameters

FILE_NAME
Specifies a 16-character file name.

POOL_NAME
Specifies an 8-character pool name.

REGION_NAME
Specifies an 8-character region name.

Output parameters

REASON
One of the following values is returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

LOCK_FAILURE

One of the following values is returned when RESPONSE is EXCEPTION:
RECORDING_NOT_ACTIVE
INCORRECT_POOL_NAME
TARGET_NOT_KNOWN
SERVER_FAILED

One of the following values is returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
RECORDING_ACTIVE

The following value is returned when RESPONSE is PURGED:
TASK_CANCELLED

RESPONSE
Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSSR gate, TEST_CONNECTION function

TEST_CONNECTION tests the status of the region status (RS) domain connection by attempting a read from the Coupling Facility (CF) for the pool name.

Input parameters

FILE_NAME
Specifies a 16-character file name.

POOL_NAME
Specifies an 8-character pool name.

Output parameters

REASON
One of the following values is returned when RESPONSE is DISASTER:
ABEND
LOOP
LOCK_FAILURE

One of the following values is returned when RESPONSE is EXCEPTION:
RECORDING_NOT_ACTIVE
INCORRECT_POOL_NAME
TARGET_NOT_KNOWN
SERVER_FAILED

One of the following values is returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
RECORDING_ACTIVE

The following value is returned when RESPONSE is PURGED:
TASK_CANCELLED

RESPONSE
Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSXM gate, END_TRANSACTION function

The END_TRANSACTION function is called at the end of each transaction to update the number of active and queued transactions in the region.

Input parameters

ACTIVE_TXN_COUNT
Specifies the number of started transactions in the region.

QUEUED_TXN_COUNT
Specifies the number of transactions that are queued in the region because a MAXTASK limit has been exceeded.

Output parameters

REASON
One of the following values is returned when RESPONSE is DISASTER:
ABEND
LOOP

One of the following values is returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following value is returned when RESPONSE is PURGED:
TASK_CANCELLED

RESPONSE
Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RSXM gate, START_TRANSACTION function

The START_TRANSACTION function is called at the start of each transaction to update the number of active and queued transactions in the region.

Input parameters

ACTIVE_TXN_COUNT
Specifies the number of started transactions in the region.

QUEUED_TXN_COUNT
Specifies the number of transactions that are queued in the region because a MAXTASK limit has been exceeded.

Output parameters

REASON
One of the following values is returned when RESPONSE is DISASTER:
ABEND
LOOP

One of the following values is returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following value is returned when RESPONSE is PURGED:
TASK_CANCELLED

RESPONSE
Indicates whether the domain call was successful.

For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Region status domains generic gates

The generic gates provide access for other domains to functions that are provided by the RS domain.

Table 42 summarizes the region status domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 42. Region status domain's generic gates

Gate	Trace	Function	Format
RSDM	RS 0101 RS 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
RSSM	RS 0300 RS 0301	STORAGE_NOTIFY	SMNT
RSXM	RS 0400 RS 0401	MXT_CHANGE_NOTIFY MXT_NOTIFY	XMNT

Modules

The RS domain modules handle requests to process, format or broadcast RS domain data.

Module	Function
DFHMERSx	RS domain messages
DFHRSDM	RS domain initialization and termination program
DFHRSDU	RS domain dump domain interface
DFHRSDUF	RS domain dump formatting
DFHRSFD	RS domain Create Region Status CFDT File
DFHRSSM	RS domain storage notification handler
DFHRSSR	RS domain request handler
DFHRSXM	RS domain transaction manager interface and transaction manager notification handler
DFHRSXRI	RS domain trace formatting

RRMS domain (RX)

The RRMS domain is responsible for managing interaction with z/OS Recoverable Resource Management Services (RRMS) and in particular, Resource Recovery Services (RRS) which is a component of RRMS.

RRMS domain's specific gates

The specific gates provide access for other domains to functions that are provided by the RX domain.

RXDM gate, INQUIRE_RRS function

The INQUIRE_RRS function of the RXDM gate is used to determine the status of the interface with Recoverable Resource Management Services (RRMS).

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

OPEN

A binary value indicating if the interface is open.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RESTART_STATE
Optional Parameter

The restart state of RRS

Values for the parameter are:
COLD
NOT_STARTED
STARTING
WARM

RXDM gate, SET_PARAMETERS function

The SET_PARAMETERS function of the RXDM gate is used to pass the values of relevant System Initialization parameters to the domain.

Input Parameters

RRMS
A binary value that specifies whether CICS is to register as a resource manager with recoverable resource management services (RRMS).

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RXUW gate, GET_CLIENT_REQUEST function

The GET_CLIENT_REQUEST function of the RXDM gate is used to suspend a transaction until the PUT_CLIENT_REQUEST is issued for the same Unit of Recovery.

Input Parameters

UR_TOKEN
is the token by which the UR associated with the request is known by the RX domain.

TIMEOUT
Optional Parameter

The time (in seconds) for which the transaction should be suspended. If this parameter is omitted, the transaction will be suspended indefinitely.

Output Parameters

REASON
The values for the parameter are:
BACKOUT
RACE

SYNCPOINT
TASK_CANCELLED
TIMED_OUT

CLIENT_TOKEN
A token representing the client of the UR.

CLIENT_TYPE
Indicates the type of client of the transaction.

Values for the parameter are:
TERMINAL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RXUW gate, INQUIRE function

The INQUIRE function requests attributes of a Unit of Recovery

Input Parameters

UR_TOKEN
is the token by which the UR associated with the request is known by the RX domain.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BACKOUT
RACE
RRS_UNAVAILABLE
SYNCPOINT

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

URID
Optional Parameter

The identifier of the Unit of Recovery used by RRMS.

RXUW gate, PUT_CLIENT_REQUEST function

The PUT_CLIENT_REQUEST function of the RXDM gate is used to associate a request from a client with an RRS Unit of Recovery (UR).

Input Parameters

CLIENT_TOKEN
A token representing the client of the UR.

CLIENT_TYPE
Indicates the type of client of the transaction.

Values for the parameter are:
TERMINAL

CONNECTION
The connection on which the client request was received. This parameter is used to identify the source of the request in any messages that are issued.

CONTEXT_TOKEN
The token representing the RRMS context for which the request is issued.
PASS_TOKEN
A token used to protect against unauthorised use of the context token and URID.
TRANSACTION_ID
The transaction id associated with the request. This parmeter is used to correlate succesive requests for the same transaction instance.
URID
The identifier of the RRS Unit of Recovery associated with the context.
USERID
The userid associated with the request. This parmeter is used to correlate succesive requests for the same transaction instance.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
BACKOUT
RACE
RRS_UNAVAILABLE
SYNCPOINT
```
The following values are returned when RESPONSE is INVALID:
```
INVALID_FUNCTION
```
The following values are returned when RESPONSE is PURGED:
```
TASK_CANCELLED
TIMED_OUT
```
NEW_UR
Indicates whether a new UR has been created for this request.

Values for the parameter are:
```
YES
NO_AND_READY
NO_AND_NOT_READY
NO_AND_NOTASK
```
YES
Indicates that a new UR has been created
NO_AND_READY
Indicates that the request was associated with an existing UR and that task is ready to receive the request.
NO_AND_NOT_READY
Indicates that the request was associated with an existing UR but that task is not ready to receive the request. This typically occurs when the original request has timed out and another transactional EXCI request in the same RU has been sent by the EXCI job.
NO_AND_NOTASK
Indicates that the request was associated with an existing UR but that task has not yet expressed an interest in that UR. This can occur when the original request has been held by MAXTASK or TRANCLASS (TCLASS) limits and has timed out, and another transactional EXCI request in the same RU has been sent by the EXCI job.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
TRANSACTION_NUMBER
The transaction number of the transaction associated with the request.

UR_TOKEN
is the token by which the UR associated with the request is known by the RX domain.

RRMS domain's call-back gates

Table 43 summarizes the domain's call-back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 43. RRMS domain's call-back gates

Gate	Trace	Function	Format
RXXM	RX 0401 RX 0402	INIT_XM_CLIENT BIND_XM_CLIENT	XMAC

Modules

Module	Function
DFHRXDM	RX domain management and global functions.
DFHRXUW	RX domain unit-of-work related functions. .
DFHRXSVC	RX domain SVC code for RRMS authorized interface.
DFHRXXRG	RX domain Registration Services exits.
DFHRXXRM	RX domain Resource Manager exits.
DFHRXDUF	RX domain dump formatting.
DFHRXTRI	RX domain trace interpretation

Request streams domain (RZ)

The request streams domain provides context switching facilities. The domain allows a running task to block while a new task is started, potentially under a different user ID or transaction. When the new task ends, control returns to the original task. For example, the RZ domain is used by web services when a provider mode application runs under a different user ID from the pipeline that attached it.

Request streams domain's specific gates

The specific gates provide access for other domains to functions that are provided by the RZ domain.

RZRJ gate, PERFORM_JOIN function

This function reduces the calls necessary from the join task (in remote join capability) to the RZ domain. It initiates the procedures necessary to pass an attached RequestStream to a local processor.

Output Parameters

REASON
The values for the parameter are:

```
JOIN_NOT_POSSIBLE
TRANSPORT_FAILURE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RZRT gate, SET_EXIT_PROGRAM function

The following defines the syntax of the SET_EXIT_PROGRAM function.

Input Parameters

PROGRAM_NAME

The name of the user-replaceable program for the Distributed Dynamic Routing program.

LOCAL_SYSID

Optional Parameter

The SYSID for the local CICS region to recognize it in routing user-replaceable program responses.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RZSO gate, CREATE function

Create a RequestStream and return a (local region) source RequestStream token for it.

The target process(or) is identified either by USERID and TRANID or by HOST_IP_ADDRESS and PORT_NUMBER. Precisely one of these groups must be provided. (The HOST_IP_ADDRESS is a character string as expected by the internal sockets domain interfaces.)

The SERVER_DATA may be retrieved at the target (RZTA) interface and is copied (and fixed) on this call.

The response is (exception, service_not_available) if it is not possible to resolve the target, or to set up a connection to the target. (Success does not guarantee that this exception will not occur on the SEND function.)

The response is (exception, target_unknown) if the HOST_IP_ADDRESS character string is malformed (as detected by the sockets domain interfaces). The response is invalid when the parameters are badly formed, in particular if there is not the correct combination of target identification parameters.

Input Parameters

CERTIFICATE_LABEL

Optional Parameter

The label of an X.509 certificate that is used during the SSL handshake

CIPHER_COUNT

Optional Parameter

The number of cipher suites encoded in the **CIPHER_SUITES** parameter.

CIPHER_SUITES

Optional Parameter

A binary representation of the cipher suites used to encrypt data.

DEBUG_BLOCK

Optional Parameter

A block used to return debugging information.

HOST_IP_ADDRESS
Optional Parameter

Identification of the target which is to process the requests.

PORT_NUMBER
Optional Parameter

Further identification of the target.

PRIVACY
Optional Parameter

Specifies the level of SSL encryption required.

Values for the parameter are:
```
NOTSUPPORTED
REQUIRED
SUPPORTED
```

SERVER_BLOCK
Optional Parameter

Data associated with the RequestStream available at the target end by the server using the RZTA interface.

SSL_REQUIRED
Optional Parameter

Whether to use SSL on a socket transport. Otherwise ignored.

Values for the parameter are:
```
NO
YES
```

TRANID
Optional Parameter

TranId of the transaction which runs the target processor.

USER_KEY_VERSION
Optional Parameter

USERID
Optional Parameter

Userid under which the requests are to be processed.

Output Parameters

REASON
The values for the parameter are:
```
SERVICE_NOT_AVAILABLE
TARGET_UNKNOWN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RS_TOKEN
Token by which RequestStream is identified on all subsequent requests from this task on this region.

APPLID
Optional Parameter

The application ID of the target processor.

RZSO gate, JOIN function

Join a RequestStream identified by a public_id.

If the required transport mechanism is not available, or fails in use, the appropriate exception is returned as for "create". If the RequestStream, identified by the "public_id", does not exist (because the target end does not exist) then this call does not detect this. Instead a new request processor will be created implicitly just as for "create". The "userid" (if supplied) must match that used on the "create", otherwise an error may occur later in (Request Processor) processing. This is not detected at this call. The "tranid" and the "server_data" is supplied in case the RequestStream is re-created on this call, otherwise they are ignored. They may be omitted as in *create*. If the "public_id" is not valid, or cannot be interpreted then the response "(exception, public_id_invalid)" will be returned. The "rs_token" for the local source RequestStream is returned as result.

Input Parameters

PUBLIC_ID
The public RequestStream identifier, valid for all participating regions in the logical server, of the target RequestStream, which may be in a separate region.

TRANID
The transaction identifier of the transaction which runs the target processor.

DEBUG_BLOCK
Optional Parameter

A block used to return debugging information.

SERVER_BLOCK
Optional Parameter

Data associated with the RequestStream available at the target end by the server using the RZTA interface.

USERID
Optional Parameter

Userid under which the requests are to be processed.

Output Parameters

REASON
The values for the parameter are:

```
INVALID_USERID
PUBLIC_ID_INVALID
SERVICE_NOT_AVAILABLE
TRANSPORT_FAILURE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see “**RESPONSE** parameter on domain interfaces” on page 7.

RS_TOKEN
A token by which the RequestStream is identified on all subsequent requests from this task on this region.

RZSO gate, LEAVE function

Remove this source from its RequestStream. The RequestStream is modified so that the "rs_token" (which must denote a source end of the RequestStream) is no longer valid. (A token value may or may not be reissued by "RZ" on another "create" or "join" request - however the caller must not rely on its value after "leave".)

Input Parameters

RS_TOKEN
Token returned on CREATE by which RequestStream is identified.

Output Parameters

REASON

The values for the parameter are:

```
RS_TOKEN_NOT_SOURCE
RS_TOKEN_UNKNOWN
TRANSPORT_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RZSO gate, RECEIVE_REPLY function

A reply is returned (blocks until one is available).

Input Parameters

RS_TOKEN

Token returned on CREATE by which RequestStream is identified.

MINIMUM_DATA_LENGTH

Optional Parameter

Minimum amount of data to accept (multiple transfers may occur until this amount is received).

REPLY_BUFFER

Optional Parameter

Buffer in which reply bytes are assembled.

Output Parameters

REASON

The values for the parameter are:

```
INVALID_BUFFER
REQUEST_PROCESSOR_FAILURE
RS_TOKEN_UNKNOWN
SERVICE_NOT_AVAILABLE
TRANSPORT_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REPLY_DATA_LENGTH

Optional Parameter

Total length of reply (even if not all received in one call).

RZSO gate, SEND_REQUEST function

The source RequestStream token and the request (coded as a *RUEI* or as a contiguous data block) is passed as input. Either a RUEI or a block must be used, not both. If this is not so then an invalid response is returned.

The request is deemed to be entire and may be presented to the target. Data may be transported across the transport mechanism during this call. The request may be of zero length, this does not imply that nothing is transported.

If the source RequestStream token does not exist (in the local region) the response (exception, rs_token_unknown) is returned.

If a transport mechanism fails to respond, or is not functional, then the response (exception, service_not_available) is returned. If it fails during transmission then

(exception, transport_failure) is returned. The distinction is that in the former case there is no transport mechanism and in the latter there is still one (albeit inoperational).

Input Parameters

RS_TOKEN
Token returned on CREATE by which RequestStream is identified.

LAST
Optional Parameter

A binary value indicating if this is the last request.

Values for the parameter are:
NO
YES

REQUEST_BLOCK
Optional Parameter

Request data to send described as a single block. Exclusive with REQUEST_RUEI.

REQUEST_RUEI
Optional Parameter

Reusable-extended-Iliffe Vector which describes contiguous bytes to send as a request, supplied in possibly discontiguous blocks. Exclusive with REQUST_BLOCK.

TARGET_PROGRAM
Optional Parameter

The name of the program in the target that will receive the request.

Output Parameters

REASON
The values for the parameter are:
RS_TOKEN_UNKNOWN
SERVICE_NOT_AVAILABLE
TRANSPORT_FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RZSO gate, WEAK_JOIN function

Join a RequestStream when there is no public_id.

Input Parameters

APPLID
The application ID of the target.

TRANID
The transaction identifier of the transaction which runs the target processor.

USERID
The user identifier associated with the current task.

SERVER_BLOCK
Optional Parameter

Data associated with the RequestStream available at the target end by the server using the RZTA interface.

Output Parameters

REASON

The values for the parameter are:

INVALID_USERID
SERVICE_NOT_AVAILABLE
TRANSPORT_FAILURE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RS_TOKEN

A token by which the RequestStream is identified on all subsequent requests from this task on this region.

RZTA gate, GET_CURRENT function

The token for the RequestStream for the current transaction is returned. If the "XM" token is not set, or is set to an invalid value, then the response "(exception, RequestStream_not_current)" is returned.

Output Parameters

REASON

The values for the parameter are:

REQUESTSTREAM_NOT_CURRENT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RS_TOKEN

Token by which RequestStream is identified on all subsequent requests from this task on this region.

RZTA gate, GET_DEBUG_DATA function

The GET_DEBUG_DATA function returns debugging information about the current request stream for use in end-to-end debugging.

Input Parameters

DEBUG_BLOCK

A block containing the debugging information returned by the domain.

Output Parameters

REASON

The values for the parameter are:

REQUESTSTREAM_NOT_CURRENT
SERVER_BLOCK_TOO_SMALL

DEBUG_DATA_LENGTH

The length of the debugging information returned.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RZTA gate, GET_JOIN_DATA function

This is a utility function used by the join task which can thereby reduce the number of domain calls to RZ when acting as intermediary to another task on remote join.

Output Parameters

REASON

The values for the parameter are:

REQUESTSTREAM_NOT_CURRENT

PUBLIC_ID
Public RequestStream Identifier -- valid for all participating regions in the logical server -- of the current target RequestStream which must be attached to this task/transaction.

REQUEST_DATA_LENGTH
The data length of the request to be passed to the processor to be joined.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANID
The transid of the request processor to be joined.

RZTA gate, GET_PUBLIC_ID function

The public identifier of the RequestStream for the current transaction is returned. (If the target of the RequestStream is not internal to the plex there may not be a public identifier, for example in the case of outbound RequestStreams. In this case the response is "(exception, public_id_unknown)". However, this should never happen on this interface, since such a RequestStream will never be set in the "RZ" transaction manager token for a transaction instance.)

Output Parameters

REASON
The values for the parameter are:
PUBLIC_ID_UNKNOWN
REQUESTSTREAM_NOT_CURRENT

PUBLIC_ID
Public RequestStream Identifier -- valid for all participating regions in the logical server -- of the current target RequestStream which must be attached to this task/transaction.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RZTA gate, GET_SERVER_DATA function

Return the server data for the current RequestStream.

Input Parameters

SERVER_BLOCK
Data associated with the RequestStream available at the target end by the server using the RZTA interface.

Output Parameters

REASON
The values for the parameter are:
REQUESTSTREAM_NOT_CURRENT
SERVER_BLOCK_TOO_SMALL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SERVER_DATA_LENGTH
The number of bytes of the server data, even if not all were returned.

RZTA gate, RECEIVE_REQUEST function

Get the next request. This call blocks if there is no request ready, and returns when a request becomes available or if the RequestStream is destroyed while waiting ("terminated"). This call will be satisfied without undue waiting if a "notify" callback has been invoked.

Input Parameters

REQUEST_BUFFER

Buffer into which the request is received.

MINIMUM_DATA_LENGTH

Optional Parameter

Minimum amount of data to accept (multiple transfers may occur until this amount is received).

Output Parameters

REASON

The values for the parameter are:

```
INVALID_BUFFER
REQUESTSTREAM_NOT_CURRENT
SERVICE_NOT_AVAILABLE
TRANSPORT_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CORRELATION_ID

Optional Parameter

The identifier of the requester using this RequestStream. It is used when replying to this request (using SEND_REPLY on this RequestStream) so as to identify the source from which the request was issued. It is valid only while this RequestStream is available to this transaction.

RZTA gate, SEND_REPLY function

Send a reply to a source identified by "correlation_id".

The "correlation_id" must be one returned by the "receive_request" function for the current RequestStream, or else the exception "correlation_id_unknown" may be returned. A reply may consist of the empty sequence of bytes in which case an empty reply is sent. The usual exceptions are returned for transportation failures.

Input Parameters

CORRELATION_ID

The correlation id received on RECEIVE_REQUEST for the request to which this is the reply.

REPLY_BLOCK

A block containing the complete contiguous reply.

LAST

Optional Parameter

Indicates if this is the last request.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON

The values for the parameter are:

```
CORRELATION_ID_UNKNOWN
REQUESTSTREAM_NOT_CURRENT
SERVICE_NOT_AVAILABLE
TRANSPORT_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RZTA gate, TERMINATE function

Terminate the current (target) RequestStream either normally or abnormally. After this call the "XM" token in the transaction instance is cleared and no longer denotes a RequestStream.

Output Parameters

REASON

The values for the parameter are:

```
CANNOT_TERMINATE_NORMALLY
REQUESTSTREAM_NOT_CURRENT
RS_TOKEN_UNKNOWN
SERVICE_NOT_AVAILABLE
TRANSPORT_FAILURE
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHRZDUF	Dump Formatting program
DFHRZIX	XM Attach Client for InStore transports
DFHRZJN	Join task program
DFHRZLN	Listen and Notify calls
DFHRZNR2	Initialize `rsnr` class (notification object)
DFHRZOFI	Initialize object factory class
DFHRZRG2	Initialize `rsrg` registration class
DFHRZRJ	Perform join
DFHRZRM	RM Resource Owner for RZ
DFHRZRS1	RM Resource Owner for RZ
DFHRZRT	Set Routing Exit program name
DFHRZRT1	Initialize routing user-replaceable program class (`rzrt`)
DFHRZRT2	Invoke Routing user-replaceable program
DFHRZSO	Source commands on RequestStreams (not Create/Join)
DFHRZSO1	Create and Join commands on Source RequestStreams
DFHRZTA	Target commands on RequestStreams
DFHRZTCX	XM Attach Client for MRO transports
DFHRZTRI	Trace interpretation

Module	Function
DFHRZTR1	Initialize rztr class
DFHRZVP1	Initialize rzvp class
DFHRZXM	XM Attach Client for RequestStreams

Scheduler services domain (SH)

The scheduler services domain is used to harden schedule requests between the end of one unit of work and the start of the next, and to route schedule requests to a target region identified by the distributed routing exit program. A schedule request is a request to undertake a piece of work, or execute a named transaction. The domain is used by CICS business transaction services.

Scheduler services domain's specific gates

The specific gates provide access for other domains to functions that are provided by the SH domain.

SHPR gate, ADD_PENDING_REQUEST function

The ADD_PENDING_REQUEST function of the SHPR gate is used to add a pending schedule request to the scheduler services queue associated with this UOW. The pending schedule requests are hardened to the scheduler services local request queue (LRQ) as part of syncpoint processing.

Input Parameters

ACTIVITY_REQUEST_BLOCK
Is a block containing the BAM domain activity request block.

BALANCE
Indicates whether this schedule request is eligible for workload routing.

Values for the parameter are:
NO
YES

TOKEN
A string of length 4, used to identify the pending queue.

TRANID
A 4-character transaction ID.

USERID
An 8-character user ID.

ACTIVITY_ID
Optional Parameter

A block containing the activity ID.

ACTIVITY_NAME
Optional Parameter

The name of the activity.

PNAME
Optional Parameter

The 36-character process name.

PROCESS_ID
Optional Parameter

is a block containing the process ID.

PTYPE
Optional Parameter

The 8-character process type.

TIME

Optional Parameter

A string of length 8, used when a request is delayed for a period time.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHPR gate, DELETE_PENDING_REQUEST function

The DELETE_PENDING_REQUEST of the SHPR gate is used to delete a pending request queue.

Input Parameters

TOKEN

is a string of length 4, used to identify the pending queue.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

REQUEST_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHPR gate, SET_BOUND_REQUEST function

The SET_BOUND_REQUEST function of the SHPR gate is used to update the schedule request to indicate that a process or activity has completed.

Input Parameters

ACTIVITY_COMPLETE

indicates whether the activity associated with this UOW has completed.

Values for the parameter are:

NO

YES

PROCESS_COMPLETE

indicates whether the process associated with this UOW has completed.

Values for the parameter are:

NO

YES

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

REQUEST_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHRQ gate, PERFORM_REGULAR_DREDGE function

The PERFORM_REGULAR_DREDGE function of the SHRQ gate initiates the periodic dredging of expired schedule requests on the local request queue (LRQ).

Output Parameters

QUIESCE

A binary value indicating that whether the system is quuiescing.

Values for the parameter are:

NO
YES

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHRQ gate, PERFORM_RESTART_DREDGE function

The PERFORM_RESTART_DREDGE of the SHRQ gate is used to initiate the dredging of expired schedule requests on the local request queue (LRQ) after a CICS system restart.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHRQ gate, PERFORM_SHUTDOWN function

The PERFORM_SHUTDOWN function of the SHRQ gate is used to stop dredging of schedule requests on the local request queue (LRQ), preventing any further CICS BTS work from being initiated.

Input Parameters

IMMEDIATE

Optional Parameter

A binary value indicating if this is an immediate shutdowm.

Values for the parameter are:

NO
YES

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHRR gate, RECEIVE_REQUEST function

The RECIEVE_REQUEST function of the SHRR gate is used to receive a schedule request once it has been routed to the target region.

Input Parameters

REQUEST_BLOCK

A block into which the request is received.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

INVALID_REQUEST_RECEIVED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHRR gate, RETRY_REQUEST function

The RETRY_REQUEST function of the SHRR gate is used obtain another target region if the initial attempt at routing the schedule request fails.

Input Parameters

REQUEST_BUFFER
is a buffer used to hold the schedule request which is to be routed.

ROUTE_ERROR
indicates the reason why the routing of the schedule request failed.

Values for the parameter are:
- ALLOCATE_REJECTED
- FUNC_NOT_SUPPORTED
- INVREQ
- LENGERR
- NO_SESSIONS
- NOTAUTH
- PGMIDERR
- QUEUE_PURGED
- SYSID_NOT_FOUND
- SYSID_OUT_SERVICE
- TERMERR

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
- NO_REQUEST_FOUND
- NO_SYSTEM
- REQUEST_BUFFER_TOO_SMALL

ABEND_CODE
is the 4-character abend code.

LOCAL
indicates whether we should retry the schedule request on the local region.

Values for the parameter are:
- NO
- YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SYSID
is the 4-character sysid of the region to which the schedule request should be routed.

SHRR gate, ROUTE_REQUEST function

The ROUTE_REQUEST function of the SHRR gate is used to identify a target region to which a schedule request should be routed.

Input Parameters

REQUEST_BUFFER
is a buffer used to hold the schedule request which is to be routed.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
- NO_REQUEST_FOUND
- NO_SYSTEM

REQUEST_BUFFER_TOO_SMALL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SYSID
is the 4-character sysid of the region to which the schedule request should be routed.

SHRT gate, INQUIRE_EXIT_PROGRAM function

The INQUIRE_EXIT_PROGRAM function returns the name of the distributed routing exit program, initially named on the **DSRTPGM** system initialization parameter.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

PROGRAM_NAME
The name of the distributed routing exit program.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SHRT gate, SET_EXIT_PROGRAM function

The SET_EXIT_PROGRAM function alters the distributed routing exit program, initially named on the **DSRTPGM** system initialization parameter. The SYSID of the local system is passed during CICS initialization.

Input Parameters

PROGRAM_NAME
The the 8-character exit program name.

LOCAL_SYSID
Optional Parameter

The 4-character local SYSID.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Scheduler services domain's generic gates

Table 44 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 44. Scheduler domain's generic gates

Gate	Trace	Function	Format
DMDM	SH 0101 SH 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

Table 44. Scheduler domain's generic gates (continued)

Gate	Trace	Function	Format
XMAC	SH 0121 SH 0122	INIT_XM_CLIENT BIND_XM_CLIENT RELEASE_XM_CLIENT	XMAC
TISR	SH 0701 SH 0702	NOTIFY	TISR
KETI	SH 0701 SH 0702	NOTIFY_RESET	KETI

When invoked for the DMDM INITIALIZE_DOMAIN function scheduler services obtains its anchor block and initializes its various classes. This would include starting the scheduler services system task , CSHY and obtaining the name of the distributed routing exit program named on the DSRTPGM system initialization parameter.

When invoked by transaction manager via the XMAC generic gate, for INIT_XM_CLIENT SH domain obtains a user token in order to set up the correct transaction environment. For BIND_XM_CLIENT SH domain initializes recoverable resources, which includes setting the RM work token and logging a backout request for this UOW. SH domain also determines the name of the program to be invoked on the initial program link.

When invoked for the RMRO PERFORM_PREPARE function SH domain prepares to commit the pending request for the UOW by adding them to the local request queue (LRQ). On receipt of the RMRO PERFORM_COMMIT the schedule requests for this UOW are committed or destroyed, depending upon whether we are committing forwards or backwards.

When invoked for the RMDE DELIVER_RECOVERY function SH domain re-creates the pending request queues and in the case of inflight UOWs attempts to retry the associated BTS activation.

Scheduler services makes use of the TISR functions, REQUEST_ NOTIFY_INTERVAL and NOTIFY to deal with delayed schedule requests i.e. EXEC CICS(R) DEFINE TIMER calls.

The KETI interface is used when the time is adjusted, causing the time at which delayed schedule requests are to expire to be recalculated.

Scheduler domain's call-back gates

Table 45 summarizes the domain's call-back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 45. Scheduler domain's call-back gates

Gate	Trace	Function	Format
RMDE	SH 0131 SH 0132	START_DELIVERY DELIVER_RECOVERY END_DELIVERY	RMDE
RMKP	SH 0131 SH 0132	TAKE_KEYPOINT	RMKP

Table 45. Scheduler domain's call-back gates (continued)

Gate	Trace	Function	Format
RMRO	SH 0131 SH 0132	PERFORM_PREPARE PERFORM_COMMIT START_BACKOUT DELIVER_BACKOUT_DATA END_BACKOUT PERFORM_SHUNT PERFORM_UNSHUNT	RMRO

When invoked for the RMRO PERFORM_PREPARE function SH domain prepares to commit the pending request for the UOW by adding them to the local request queue (LRQ). On receipt of the RMRO PERFORM_COMMIT the schedule requests for this UOW are committed or destroyed, depending upon whether we are committing forwards or backwards.

When invoked for the RMDE DELIVER_RECOVERY function SH domain re-creates the pending request queues and in the case of inflight UOWs attempts to retry the associated BTS activation.

Modules

Module	Function
DFHSHDM	Handles the following requests: PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHSHDUF	Formats the SH domain control blocks
DFHSHOFI	Initializes the SH domain object factory class.
DFHSHPR	Handles the following requests: ADD_PENDING_REQUEST DELETE_PENDING_REQUEST SET_BOUND_REQUEST
DFHSHRE1	Initializes the SH domain request class.
DFHSHRM	Handles the following requests: PERFORM_PREPARE PERFORM_COMMIT START_BACKOUT DELIVER_BACKOUT END_BACKOUT PERFORM_SHUNT PERFORM_UNSHUNT TAKE_KEYPOINT START_DELIVERY DELIVER_RECOVERY END_DELIVERY
DFHSHRQ	Handles the following requests: PERFORM_RESTART_DREDGE PERFORM_REGULAR_DREDGE PERFORM_SHUTDOWN
DFHSHRQ1	Initializes the SH domain request queue class.

Module	Function
DFHSHRR	Handles the following requests: ROUTE_REQUEST RECEIVE_REUEST RETRY_REQUEST
DFHSHRRP	The SH domain request receiving program, the back-end to SH domain DPL requests.
DFHSHRSP	The SH domain request sending program, the front-end to SH domain DPL requests.
DFHSHRT	Handles the following requests: SET_EXIT_PROGRAM INQUIRE_EXIT_PROGRAM
DFHSHRT1	Initializes the SH domain request routing class.
DFHSHRT2	Invokes the distributed routing exit program, named on the DSRTPGM system initialization parameter.
DFHSHSY	Implements the SH domain system task, CSHY.
DFHSHTI	Handles the following requests: NOTIFY NOTIFY_RESET
DFHSHTRI	Interprets SH domain trace entries
DFHSHVP1	Initializes the SH domain variable length storage class.
DFHSHXM	Handles the following requests: INIT_XM_CLIENT BIND_XM_CLIENT RELEASE_XM_CLIENT

JVM domain (SJ)

The JVM domain provides services that are used by Java virtual machines in the CICS environment.

JVM domain's specific gates

The specific gates provide access for other domains to functions that are provided by the SJ domain.

SJBD gate, END_BROWSE_OSGIBUNDLE function

The END_BROWSE_OSGIBUNDLE function of the SJBD gate ends the browse of OSGi bundles in a JVM server.

Input Parameters

BROWSE_TOKEN

The browse token from the START_BROWSE_OSGIBUNDLE function.

Output Parameters

REASON

The following values are returned:

`INVALID_BROWSE_TOKEN`

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJBD gate, END_BROWSE_OSGISERVICE function

The END_BROWSE_OSGIBUNDLE function of the SJBD gate ends the browse of OSGi services in a JVM server.

Input Parameters

BROWSE_TOKEN
The browse token from the START_BROWSE_OSGISERVICE function.

Output Parameters

REASON
The following values are returned:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJBD gate, GET_NEXT_OSGIBUNDLE function

The GET_NEXT_OSGIBUNDLE function of the SJBD gate returns the next installed OSGi bundle in a JVM server

Input Parameters

BROWSE_TOKEN
The browse token from the START_BROWSE_OSGIBUNDLE function.

BUNDLEPART
Optional parameter

Buffer to optionally receive the bundle part information.

OSGIBUNDLE_BUFFER
Optional parameter

Buffer to optionally receive the OSGi bundle symbolic name.

OSGIVERSION_BUFFER
Optional parameter

Buffer to optionally receive the OSGi bundle version.

RESOURCE_SIGNATURE
Optional parameter

Buffer to optionally receive the resource signature.

Output Parameters

BUNDLE
The name of the BUNDLE resource.

OSGIBUNDLEID
The OSGi bundle ID.

OSGISTATUS
The status of the OSGi bundle.

REASON
The following values are returned:
BROWSE_END
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJBD gate, GET_NEXT_OSGISERVICE function

The GET_NEXT_OSGISERVICE function of the SJBD gate returns the next installed OSGi service in the OSGi framework of a JVM server

Input Parameters

BROWSE_TOKEN
The browse token from the START_BROWSE_OSGISERVICE function.

BUNDLEPART
Optional parameter

Buffer to optionally receive the bundle part information.
OSGIBUNDLE_BUFFER
Optional parameter

Buffer to optionally receive the OSGi bundle symbolic name.
OSGIVERSION_BUFFER
Optional parameter

Buffer to optionally receive the OSGi bundle version.
OSGISERVICE
Optional parameter

Buffer to optionally receive the name of the OSGi service.

Output Parameters
BUNDLE
The name of the BUNDLE resource.
OSGISERVICEID
The ID of the OSGi service.
SERVICE_STATUS
The status of the OSGi service.
REASON
The following values are returned:
```
BROWSE_END
INVALID_BROWSE_TOKEN
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJBD gate, INQUIRE_OSGIBUNDLE function
The INQUIRE_OSGIBUNDLE function of the SJBD gate returns information about an OSGi bundle that is installed in a JVM server.

Input Parameters
BUNDLEPART
Optional parameter

Buffer to optionally receive the bundle part information.
JVMSERVER
The name of the JVMSERVER resource.
OSGIBUNDLE_BLOCK
The OSGi bundle symbolic name.
OSGIVERSION_BLOCK
The OSGi bundle version.
RESOURCE_SIGNATURE
Optional parameter

Buffer to optionally receive the resource signature.

Output Parameters
BUNDLE
The name of the BUNDLE resource.
OSGIBUNDLEID
The OSGi bundle ID.
OSGISTATUS
The status of the OSGi bundle.

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
OSGIBUNDLE_NOT_FOUND
```
The following values are returned when RESPONSE IS DISASTER:
```
ABEND
DIRECTORY_ERROR
INTERNAL_ERROR
SEVERE_ERROR
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJBD gate, INQUIRE_OSGISERVICE function

The INQUIRE_OSGISERVICE function of the SJBD gate inquires on installed OSGi services in the OSGi framework of a JVM server.

Input Parameters

BUNDLEPART
Optional parameter

Buffer to optionally receive the bundle part information.
JVMSERVER
The name of the JVMSERVER resource.
OSGIBUNDLE_BUFFER
Optional parameter

Buffer to optionally receive the name of the OSGi bundle.
OSGISERVICEID
The ID of the OSGi service.
OSGIVERSION_BUFFER
Optional parameter

Buffer to optionally receive the OSGi bundle version.

Output Parameters

BUNDLE
The name of the BUNDLE resource.
SERVICE_STATUS
The status of the OSGi service.
REASON
The following values are returned when RESPONSE is EXCEPTION:
```
JVMSERVER_NOT_FOUND
OSGISERVICE_NOT_FOUND
```
The following values are returned when RESPONSE is DISASTER:
```
ABEND
DIRECTORY_ERROR
INTERNAL_ERROR
SEVERE_ERROR
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJBD gate, START_BROWSE_OSGIBUNDLE function

The START_BROWSE_OSGIBUNDLE function of the SJBD gate starts a browse of a JVM server.

Input Parameters

JVMSERVER
The name of the JVMSERVER resource.

Output Parameters

BROWSE_TOKEN
A browse token that is used as input to browsing installed OSGi bundles in the JVM server.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJBD gate, START_BROWSE_OSGISERVICE function

The START_BROWSE_OSGISERVICE function of the SJBD gate starts a browse of OSGi services in the OSGi framework of a JVM server.

Input Parameters

JVMSERVER
The name of the JVMSERVER resource.

Output Parameters

BROWSE_TOKEN
A browse token that is used as input to browsing installed OSGi services in the JVM server.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJBD gate, INQUIRE_OSGIBUNDLE_BY_ID function

The INQUIRE_OSGIBUNDLE_BY_ID function of the SJBD gate takes the ID of an OSGi bundle and the name of the JVMSERVER resource as inputs and returns the details of the OSGi bundle.

Input Parameters

JVMSERVER
The name of the JVMSERVER resource.

OSGIBUNDLEID
The ID of the OSGi bundle.

Output Parameters

BUNDLE
The CICS bundle.

BUNDLEPART
CICS bundle part of the CICS bundle used to install the OSGi bundle.

OSGIBUNDLE_BUFFER
OSGi bundle symbolic name

OSGISTATUS
The OSGi status.

OSGIVERSION_BUFFER
OSGi bundle version

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
DIRECTORY_ERROR
INTERNAL_ERROR
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
OSGIBUNDLE_NOT_FOUND
OSGIBUNDLEID_NOT_VALID

RESOURCE_SIGNATURE
The resource signature of the OSGi bundle.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJDS gate, DELETE_THREADED_TCB function

The DELETE_THREADED_TCB function deletes a T8 TCB from the THRD pool.

Input parameters

TCB_TOKEN
A token that represents the TCB.

MODENAME
The mode of the TCB.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

ERROR_TERMINATING_ENCLAVE
TRANSACTION_ABENDED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJJS gate, CREATE_JVMSERVER function

The CREATE_JVMSERVER function creates a JVMSERVER resource.

Input parameters

ENABLESTATUS
Optional parameter

The status of the JVMSERVER resource.

JVMPROFILE
The JVM profile that the JVM server uses during initialization.

JVMSERVER
The name of the JVMSERVER resource.

LERUNOPTS
The program that defines the runtime options for the Language Environment enclave.

RESOURCE_SIGNATURE
The resource signature of the JVMSERVER resource.

THREADLIMIT
Optional parameter

The maximum number of threads that are allowed in the Language Environment enclave.

WARM_RESTART
Optional parameter

Indicates whether the JVMSERVER resource is to be recovered from the catalog during a warm restart of CICS.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
DIRECTORY_ERROR
DUPLICATE
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INTERNAL_ERROR
SEVERE_ERROR
THREADS_LIMITED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJJS gate, COMPLETE_JVMSERVER function

The COMPLETE_JVMSERVER function completes the installation of the JVMSERVER resource.

Input parameters

ENABLESTATUS
Optional parameter

The status of the JVMSERVER resource.

JVMPROFILE
Optional parameter

The JVM profile that the JVM server uses during initialization.

JVMSERVER
The name of the JVMSERVER resource.

LERUNOPTS
Optional parameter

The program that defines the runtime options for the Language Environment enclave.

RESOURCE_SIGNATURE
Optional parameter

The resource signature of the JVMSERVER resource.

THREADLIMIT
Optional parameter

The maximum number of threads that are allowed in the Language Environment enclave.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
ACTIVATE_TP_FAILED

CREATE_ENCLAVE_FAILED
DIRECTORY_ERROR
INSUFFICIENT_STORAGE
INTERNAL_ERROR
LE_RUNOPTS_LOAD_ERROR
LE_RUNOPTS_TOO_LONG
SEVERE_ERROR

The following values are returned when RESPONSE is EXCEPTION:
JVMSERVER_NOT_FOUND
NOT_AUTHORIZED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJJS gate, DISCARD_JVMSERVER function

The DISCARD_JVMSERVER function discards a JVMSERVER resource.

Input parameters

ENABLESTATUS
Optional parameter

The status of the JVMSERVER resource.

JVMPROFILE
Optional parameter

The JVM profile that the JVM server uses during initialization.

JVMSERVER
The name of the JVMSERVER resource.

LERUNOPTS
Optional parameter

The program that defines the runtime options for the Language Environment enclave.

RESOURCE_SIGNATURE
Optional parameter

The resource signature of the JVMSERVER resource.

THREADLIMIT
Optional parameter

The maximum number of threads that are allowed in the Language Environment enclave.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DIRECTORY_ERROR
INTERNAL_ERROR
NOT_DISABLED
SEVERE_ERROR

The following values are returned when RESPONSE is EXCEPTION:
JVMSERVER_NOT_FOUND
NOT_AUTHORIZED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJJS gate, END_BROWSE_JVMSERVER function

The END_BROWSE_JVMSERVER function ends the browse operation for JVMSERVER resources.

Input parameters

BROWSE_TOKEN
The token for the browse operation.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_BROWSE_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJJS gate, GET_NEXT_JVMSERVER function

The GET_NEXT_JVMSERVER function returns the next name in the browse specified by the browse token and returns the attributes associated with the JVMSERVER resource.

Input parameters

BROWSE_TOKEN
The token that identifies the requested browse of JVMSERVER resources.

RESET
Optional parameter

Reset the browse operation.

RESOURCE_SIGNATURE
Optional parameter

The resource signature of a JVMSERVER resource.

Output parameters

ENABLESTATUS
Optional parameter

The status of the JVMSERVER resource.

JVMPROFILE
Optional parameter

The JVM profile that the JVM server uses during initialization.

JVMSERVER
The name of a JVMSERVER resource.

LERUNOPTS
Optional parameter

The program that defines the runtime options for the Language Environment enclave.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
INVALID_BROWSE_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

THREADLIMIT
Optional parameter

The maximum number of threads that are allowed in the Language Environment enclave.

SJJS gate, INQUIRE_JVMSERVER function

The INQUIRE_JVMSERVER function inquires on a JVMSERVER resource.

Input parameters

JVMSERVER
The name of the JVMSERVER resource.

RESOURCE_SIGNATURE
Optional parameter

The resource signature of the JVMSERVER resource.

Output parameters

ENABLESTATUS
Optional parameter

The status of the JVMSERVER resource.

JVMPROFILE
Optional parameter

The JVM profile that the JVM server uses during initialization.

LERUNOPTS
Optional parameter

The program that defines the runtime options for the Language Environment enclave.

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
DIRECTORY_ERROR
INTERNAL_ERROR
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
JVMSERVER_NOT_FOUND
NOT_AUTHORIZED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

THREADLIMIT
Optional parameter

The maximum number of threads that are allowed in the Language Environment enclave.

SJJS gate, MARK_THREAD_DELETED function

The MARK_THREAD_DELETED function deletes a thread when the CICS dispatcher deletes the associated T8 TCB.

Input parameters

TCB_TOKEN
A token that represents the T8 TCB.

Output parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJJS gate, RESOLVE_ALL_JVMSERVERS function

The RESOLVE_ALL_JVMSERVERS function runs the CJSR transaction for all JVMSERVER resources that are in the enabling state.

Input parameters

JVMSERVER
Optional parameter

The name of the JVMSERVER resource.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND

CREATE_ENCLAVE_FAILED

DIRECTORY_ERROR

INTERNAL_ERROR

SEVERE_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJJS gate, SET_JVMSERVER function

The SET_JVMSERVER function sets the attributes of a JVMSERVER resource.

Input parameters

ENABLESTATUS
Optional parameter

The status of the JVMSERVER resource.

JVMSERVER
The name of the JVMSERVER resource.

THREADLIMIT
Optional parameter

The maximum number of threads that are allowed in the Language Environment enclave.

Output parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND

DIRECTORY_ERROR

INTERNAL_ERROR

SEVERE_ERROR

The following values are returned when RESPONSE is EXCEPTION:

JVMSERVER_NOT_FOUND
NOT_AUTHORIZED
WRONG_STATE
INSUFFICIENT_THREADS
THREADS_LIMITED
CREATE_ENCLAVE_FAILED
INVALID_THREADLIMIT
JVMSERVER_IN_USE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJJS gate, START_BROWSE_JVMSERVER function

The START_BROWSE_JVMSERVER function starts to browse installed JVMSERVER resources.

Input parameters

None.

Output parameters

BROWSE_TOKEN
The browse token for the browse operation.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, DISABLE_OSGIBUNDLE function

The DISABLE_OSGIBUNDLE function disables an OSGi bundle in the OSGi framework of a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

OSGIBUNDLE_ID
The ID of the OSGi bundle.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

OSGIBUNDLE_STATUS
The status of the OSGi bundle.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND
DETACH_THREAD_FAILED
DISABLE_OSGIBUNDLE_FAILED
INTERNAL_ERROR
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND

METHOD_NOT_FOUND

NO_PTHREAD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, ENABLE_OSGIBUNDLE function

The ENABLE_OSGIBUNDLE function installs an OSGi bundle in the OSGi framework of a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

OSGIBUNDLE_ID
The ID of the OSGi bundle.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

OSGIBUNDLE_STATUS
The status of the OSGi bundle.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED

CLASS_NOT_FOUND

DETACH_THREAD_FAILED

ENABLE_OSGIBUNDLE_FAILED

INTERNAL_ERROR

JVMSERVER_NOT_ENABLED

JVMSERVER_NOT_FOUND

METHOD_NOT_FOUND

NO_PTHREAD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, END_BROWSE_OSGISERVICE function

The END_BROWSE_OSGISERVICE function ends the browse of OSGi services in the OSGi framework of a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

BROWSE_TOKEN
A browse token for browsing the JVM server.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND
DETACH_THREAD_FAILED
END_BROWSE_OSGISRV_FAILED
INTERNAL_ERROR
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
METHOD_NOT_FOUND
NO_PTHREAD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, FIND_HIGHEST_VERSION function

The FIND_HIGHEST_VERSION function determines the highest semantic version of an OSGi bundle on zFS.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.
BUNDLE_DIR
The root directory on zFS for the OSGi bundle.
BUNDLE_NAME
The name of the OSGi bundle.
BUNDLE_VERSION_RANGE
The version range string for the OSGi bundle.
HIGHEST_VERSION
The buffer to contain returned highest semantic version string.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND
DETACH_THREAD_FAILED
FIND_HIGHEST_VER_FAILED
INTERNAL_ERROR
INVALID_VERSION_RANGE
JVMSERVER_ENABLING
JVMSERVER_EXITING
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
METHOD_NOT_FOUND
NO_BUNDLE_IN_RANGE
NO_PTHREAD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate,GET_NEXT_OSGISERVICE function

The GET_NEXT_OSGISERVICE function gets the next OSGi service when browsing the OSGi framework of a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

BROWSE_TOKEN
A browse token for browsing the JVM server.

SERVICE_BUFFER
Optional parameter

A buffer to return the name of the OSGi service.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

OSGIBUNDLE_ID
The ID of the OSGi bundle.

SERVICE_ID
The ID of the OSGi service.

SERVICE_STATUS
The status of the OSGi service.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND
DETACH_THREAD_FAILED
END_OF_SERVICES
GET_NEXT_OSGISRV_FAILED
INTERNAL_ERROR
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
METHOD_NOT_FOUND
NO_PTHREAD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, INQUIRE_OSGIBUNDLE function

The INQUIRE_OSGIBUNDLE function returns information about an OSGi bundle in the OSGi framework of a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

OSGIBUNDLE_ID
The ID of the OSGi bundle.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

OSGIBUNDLE_STATUS
The status of the OSGi bundle.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED

CLASS_NOT_FOUND

DETACH_THREAD_FAILED

INQUIRE_OSGIBUNDLE_FAILED

INTERNAL_ERROR

JVMSERVER_NOT_ENABLED

JVMSERVER_NOT_FOUND

METHOD_NOT_FOUND

NO_PTHREAD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, INQUIRE_OSGISERVICE function

The INQUIRE_OSGISERVICE function returns information about an OSGi service in the OSGi framework of a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

SERVICE_ID
The ID of the OSGi service.

SERVICE_BUFFER
Optional parameter

A buffer to return the name of the OSGi service.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

OSGIBUNDLE_ID
The ID of the OSGi bundle.

SERVICE_STATUS
The status of the OSGi service.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED

CLASS_NOT_FOUND

DETACH_THREAD_FAILED

INQUIRE_OSGISERVICE_FAILED
INTERNAL_ERROR
JVM_THREW_EXCEPTION
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
METHOD_NOT_FOUND
NO_PTHREAD
SERVICE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, INSTALL_OSGIBUNDLE function

The INSTALL_OSGIBUNDLE function installs an OSGi bundle in the OSGi framework of a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

JVM_TOKEN
Optional parameter

A token that represents the JVM.

USER_CLASS
Optional parameter

A user Java class.

USER_METHOD_NAME
Optional parameter

The name of the user method.

USER_METHOD_SIG
Optional parameter

WRAPPER_CLASS
Optional parameter

The wrapper class

WRAPPER_METH_NAME
Optional parameter

The name of the method in the wrapper.

WRAPPER_METH_SIG
Optional parameter

SYSTEM_THREAD
Optional parameter

This parameter value is either YES or NO.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND

DETACH_THREAD_FAILED
INTERNAL_ERROR
JVM_THREW_EXCEPTION
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
METHOD_NOT_FOUND
NO_PTHREAD
SYSTEM_THREAD_IN_USE
THREAD_KILLED
TRANSACTION_ABENDED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, INVOKE_JAVA_PROGRAM function

The INVOKE_JAVA_PROGRAM function changes TCB mode to a T8 TCB and calls the specified user Java class on a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

JVM_TOKEN
Optional parameter

A token that represents the JVM.

USER_CLASS
Optional parameter

A user Java class.

USER_METHOD_NAME
Optional parameter

The name of the user method.

USER_METHOD_SIG
Optional parameter

WRAPPER_CLASS
Optional parameter

The wrapper class

WRAPPER_METH_NAME
Optional parameter

The name of the method in the wrapper.

WRAPPER_METH_SIG
Optional parameter

SYSTEM_THREAD
Optional parameter

This parameter value is either YES or NO.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND
DETACH_THREAD_FAILED
DRIVE_JVM_STATS_FAILED
INTERNAL_ERROR
JVM_THREW_EXCEPTION
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
METHOD_NOT_FOUND
NO_PTHREAD
SYSTEM_THREAD_IN_USE
THREAD_KILLED
TRANSACTION_ABENDED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, JVMSERVER_STATISTICS function

The JVMSERVER_STATISTICS function collects statistics about a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND
DETACH_THREAD_FAILED
DRIVE_JVM_STATS_FAILED
INTERNAL_ERROR
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
METHOD_NOT_FOUND
NO_PTHREAD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, KILL_JVMSERVER function

The KILL_JVMSERVER function removes a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND
DETACH_THREAD_FAILED
INTERNAL_ERROR
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
KILL_JVMSERVER_FAILED
METHOD_NOT_FOUND
NO_PTHREAD

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, RUN_SERVICE function

The RUN_SERVICE function runs an OSGi service in the OSGi framework of a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

SERVICE_NAME
The name of the OSGi service.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND
DETACH_THREAD_FAILED
INTERNAL_ERROR
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
METHOD_NOT_FOUND
NO_PTHREAD
RUN_SERVICE_FAILED
THREAD_KILLED
TRANSACTION_ABENDED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, SET_JVMSERVER_DISABLING function

The SET_JVMSERVER_DISABLING function moves the JVM server into a disabling state.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND
DETACH_THREAD_FAILED
INTERNAL_ERROR
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
METHOD_NOT_FOUND
NO_PTHREAD
SET_JVMS_DISAB_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, START_BROWSE_OSGISERVICE function

The START_BROWSE_OSGISERVICE function starts the browse of OSGi services in the OSGi framework of a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

BROWSE_TOKEN
A token for browsing the JVM server.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED
CLASS_NOT_FOUND
DETACH_THREAD_FAILED
INTERNAL_ERROR
JVMSERVER_NOT_ENABLED
JVMSERVER_NOT_FOUND
METHOD_NOT_FOUND
NO_PTHREAD

START_BROWSE_OSGISRV_FAIL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SJTH gate, UNINSTALL_OSGIBUNDLE function

The UNINSTALL_OSGIBUNDLE function removes an OSGi bundle from the OSGi framework of a JVM server.

Input parameters

JVMSERVER
The name of a JVMSERVER resource.

OSGIBUNDLE_ID
The ID of the OSGi bundle.

Output parameters

ABEND_CODE
The CICS abend code that is returned if an abend occurs.

REASON
The following values are returned when RESPONSE is EXCEPTION:

ATTACH_THREAD_FAILED

CLASS_NOT_FOUND

DETACH_THREAD_FAILED

INTERNAL_ERROR

JVMSERVER_NOT_ENABLED

JVMSERVER_NOT_FOUND

METHOD_NOT_FOUND

NO_PTHREAD

UNINSTALL_OSGIBUNDLE_FAIL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

JVM domain's generic gates

Table 46 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 46. JVM domain's generic gates

Gate	Trace	Functions	Format
SJDM	SJ 0000 SJ 0001	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
SJIN	SJ 0200 SJ 0201	NOTIFY_DELETE_TCB	DSAT

Table 46. JVM domain's generic gates (continued)

Gate	Trace	Functions	Format
SJRL		CREATE DISCARD INQUIRE INQUIRE_BY_NAME RECREATE RECREATE_COMPLETE SET	RLCB
SJSM	SJ 0900 SJ 0901	MVS_STORAGE_NOTIFY	SMNT
SJST	SJ 0400 SJ 0401	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

Modules

Module	Function
DFHSJBD	Handles the bundle directory and its bundles for a JVM server.

DFHSJCS	An internal module which handles the following C subroutines called by SJIN:
	`sjcsbld (sjcs_build_jvm)`
	`sjcsdes (sjcs_destroy_jvm)`
	`sjcscall (sjcs_call_java_method)`
	`sjcsgc (sjcs_invoke_gc)`
	`sjcsInitCache (sjcs_init_cache)`
	`sjcsDestroyCache (sjcs_destroy_cache)`

DFHSJDM	Handles requests associated with the DMDM generic gate.
DFHSJDS	Handles the dispatcher callback to delete TCBs and pthreads.
DFHSJIN	Handles requests associated with the SJIN gate.
DFHSJIS	Handles requests associated with the SJIS gate.
DFHSJJS	Handles JVMSERVER resources.
DFHSJNT	Handles the following requests: MXT_NOTIFY
DFHSJPJP	An internal module that handles the following C subroutine called by `sjcsbld`: `sjpjp_process_jvm_profile`
DFHSJSM	Handles the following requests: STORAGE_NOTIFY
DFHSJTH	Handles JVM server threads.

Storage manager domain (SM)

The storage manager domain manages virtual storage requests.

Storage manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the SM domain.

SMAD gate, ADD_SUBPOOL function

The ADD_SUBPOOL function of the SMAD gate is used to create a new subpool with given attributes.

Input Parameters

BOUNDARY
is the boundary on which all elements within the subpool must be aligned. The boundary must be a power of two in the range 8 through 4096.

ELEMENT_CHAIN
indicates whether a chain of the addresses and lengths of the elements is to be kept.

Values for the parameter are:
NO
YES

ELEMENT_TYPE
indicates whether the subpool elements are of fixed or variable length.

Values for the parameter are:
FIXED
VARIABLE

INITIAL_FREE
is the size of the initial free storage area for the subpool.

LOCATION
specifies whether all elements within the subpool must be allocated below the maximum 24-bit address, or may be allocated anywhere.

Values for the parameter are:
ANY
BELOW

SUBPOOL_NAME
is the 8-character name by which the subpool is known.

USAGE
indicates whether the subpool is for task or domain use.

Values for the parameter are:
DOMAIN
TASK

ACCESS
Optional Parameter

The type of storage access required.

Values for the parameter are:
CICS
READ_ONLY
USER

FIXED_LENGTH
Optional Parameter

is the element length for a fixed-length subpool.

LOCK_POOL
Optional Parameter

Indicates if access to the subpool is to be controlled by a lock.

Values for the parameter are:
NO
YES

STORAGE_CHECK
Optional Parameter

indicates whether storage zone checking is to be enabled for this subpool.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE

The following values are returned when RESPONSE is INVALID:
DUPLICATE_SUBPOOL_NAME
INVALID_BOUNDARY
INVALID_FIXED_LENGTH
INVALID_INITIAL_FREE
INVALID_SUBPOOL_NAME
LOCK_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SUBPOOL_TOKEN
is the token identifying the newly created subpool.

DSA_NAME
Optional Parameter

is the name of the CICS dynamic storage area (DSA) in which the subpool resides.

Values for the parameter are:
CDSA
ECDSA
ERDSA
ESDSA
EUDSA
RDSA
SDSA
UDSA

SMAD gate, DELETE_SUBPOOL function

The DELETE_SUBPOOL function of the SMAD gate is used to delete a subpool.

Input Parameters

SUBPOOL_TOKEN
is the token identifying the subpool to be deleted.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_SUBPOOL_TOKEN
NOT_SUBPOOL_OWNER
SUBPOOL_NOT_EMPTY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMAD gate, END_SUBPOOL_BROWSE function

The END_SUBPOOL_BROWSE function of the SMAD gate is used to end a browse of the storage manager domain subpools.

Input Parameters

BROWSE_TOKEN
is the token identifying the browse operation.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMAD gate, GET_NEXT_SUBPOOL function

The GET_NEXT_SUBPOOL function of the SMAD gate is used in a storage manager domain subpool browse to get the next subpool.

Input Parameters

BROWSE_TOKEN
is the token identifying the browse operation.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SUBPOOL_NAME
is name of the subpool returned by the browse.

DSA_NAME
Optional Parameter

is the name of the CICS dynamic storage area (DSA) in which the subpool resides.

Values for the parameter are:
CDSA
ECDSA
ERDSA
ESDSA
EUDSA
RDSA
SDSA
UDSA

SMAD gate, INQUIRE_SUBPOOL function

The INQUIRE_SUBPOOL function of the SMAD gate is used to inquire about a storage manager domain subpool.

Input Parameters

SUBPOOL_NAME
is the 8-character name by which the subpool is known.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

SUBPOOL_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSA_NAME
Optional Parameter

is the name of the CICS dynamic storage area (DSA) in which the subpool resides.

Values for the parameter are:
CDSA
ECDSA
ERDSA
ESDSA
EUDSA
RDSA
SDSA
UDSA

SMAD gate, START_SUBPOOL_BROWSE function

The START_SUBPOOL_BROWSE function of the SMAD gate is used to start a browse of the storage manager domain subpools.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE

BROWSE_TOKEN
is the token identifying the browse operation.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMAR gate, ALLOCATE_TRANSACTION_STG function

The ALLOCATE_TRANSACTION_STG function of the SMAR gate is used at task initialization to add the task-lifetime storage subpools.

Input Parameters

ISOLATE
Indicates whether CICS isolates the user-key task-lifetime storage of the transaction to provide application-to-application protection, as specified by the ISOLATE attribute on the associated TRANSACTION resource definition.

Values for the parameter are as follows:
NO
YES

STORAGE_CLEAR
Indicates whether to clear task-lifetime storage to zeros when it is released by a FREEMAIN request.

Values for the parameter are as follows:
NO
YES

STORAGE_FREEZE
Indicates whether to delay task-lifetime storage FREEMAIN requests until task termination.

Values for the parameter are as follows:

NO
YES

TASK_DATAKEY
Indicates the storage key for the task-lifetime storage and program-related storage (for all programs that run under the transaction) for the transaction, as specified by the TASKDATAKEY attribute on the associated TRANSACTION resource definition.

Values for the parameter are as follows:
CICS
USER

TASK_DATALOC
Indicates the location of task data for the transaction, as specified by the TASKDATALOC attribute on the associated TRANSACTION resource definition.

Values for the parameter are as follows:
ANY
BELOW

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
INSUFFICIENT_STORAGE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMAR gate, RELEASE_TRANSACTION_STG function

The RELEASE_TRANSACTION_STG function of the SMAR gate is used at task termination to release all remaining task-lifetime storage by using a FREEMAIN request, and delete the task-lifetime storage subpools.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DEACTIVATE_FAILURE
INSUFFICIENT_STORAGE

The following values are returned when RESPONSE is EXCEPTION:
STORAGE_VIOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMCK gate, CHECK_STORAGE function

The CHECK_STORAGE function of the SMCK gate is used to check the storage check zones of task-lifetime storage, and the storage accounting areas (SAAs) of terminal storage, for consistency.

Input Parameters

TASK_STORAGE
Specifies whether the storage check zones of task-lifetime storage are checked for the current task.

Values for the parameter are as follows:

CURRENT_TASK
NO

TP_STORAGE
Specifies whether the SAAs of terminal storage are checked for the current terminal.

Values for the parameter are as follows:
CURRENT_TERMINAL
NO

STORAGE_LOC
Optional Parameter

Specifies whether storage is located above the bar (that is, 64-bit storage) or below 2 GB (below the bar; that is, 24-bit and 31-bit storage).

Values for the parameter are as follows:
ABOVEBAR
BELOWBAR

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
INVALID_FUNCTION
LOOP
STORAGE_VIOLATION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMCK gate, RECOVER_STORAGE function

The RECOVER_STORAGE function of the SMCK gate is used to recover task-lifetime storage and the storage accounting areas (SAAs) of terminal storage.

Input Parameters

TASK_STORAGE
Specifies whether to recover task-lifetime storage for the current task.

Values for the parameter are as follows:
CURRENT_TASK
NO

TP_STORAGE
Specifies whether to recover the SAAs of terminal storage for the current terminal.

Values for the parameter are as follows:
CURRENT_TERMINAL
NO

STORAGE_LOC
Optional Parameter

Specifies whether storage is located above the bar (that is, 64-bit storage) or below 2 GB (below the bar; that is, 24-bit and 31-bit storage).

Values for the parameter are as follows:
ABOVEBAR
BELOWBAR

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INVALID_FUNCTION
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
STORAGE_NOT_RECOVERED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMGF gate, FREEMAIN function

The FREEMAIN function of the SMGF gate is used to release an element of storage within a subpool.

Input Parameters

ADDRESS | ADDRESS_64

ADDRESS

The 31-bit address of the element to be released.

ADDRESS_64

The 64-bit address of the element to be released.

STORAGE_CLASS

Optional Parameter

Identifies the class of storage that is being released.

The parameter can take the following values:

```
CICS
CICS24
TASK
TASK24
TASK31
USER
USER24
```

SUBPOOL_TOKEN

Optional Parameter

A token that identifies the subpool within which the element is to be allocated.

FREE_LENGTH

Optional Parameter

The length of the element to be released.

LOCK_POOL

Optional Parameter

Indicates whether access to the subpool is controlled by a lock.

The parameter can take the following values:

```
NO
YES
```

REMARK

Optional Parameter

An optional 8-character field that is used to identify the FREEMAIN operation for problem determination. This field is highlighted when the FREEMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being obtained.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DEACTIVATE_FAILURE
LOCK_ERROR
LOOP
UNLOCK_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_ADDRESS
INVALID_FREE_LENGTH
INVALID_STORAGE_CLASS
INVALID_SUBPOOL_TOKEN
NO_FREE_LENGTH
NOT_SUBPOOL_OWNER
SUBPOOL_EMPTY
SUBPOOL_LOCK_FAILED
SUBPOOL_UNLOCK_FAILED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMGF gate, GETMAIN function

The GETMAIN function of the SMGF gate is used to allocate an element of storage from a subpool.

Input Parameters

STORAGE_CLASS

Optional Parameter

Identifies the class of storage to be allocated.

The parameter can take the following values:

```
CICS
CICS24
TASK
TASK24
TASK31
USER
USER24
```

SUBPOOL_TOKEN

Optional Parameter

A token that identifies the subpool within which the element is to be allocated.

SUSPEND

If there is insufficient storage to satisfy the request, SUSPEND(YES) suspends the caller until the request can be satisfied, and SUSPEND(NO) sets REASON to INSUFFICIENT_STORAGE.

The parameter can take the following values:

```
NO
YES
```

GET_LENGTH

Optional Parameter

The length of the storage requested.

INITIAL_IMAGE

Optional Parameter

An optional byte value to which every byte in the new element is set.

LOCK_POOL
Optional Parameter

Indicates whether access to the subpool is to be controlled by a lock.

The parameter can take the following values:
NO
YES

REMARK
Optional Parameter

An optional 8-character field that identifies the GETMAIN operation for problem determination. This field is highlighted when the GETMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being obtained.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
ACTIVATE_FAILURE
LOCK_ERROR
LOOP
UNLOCK_ERROR

The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE

The following values are returned when RESPONSE is INVALID:
INVALID_GET_LENGTH
INVALID_INITIAL_IMAGE
INVALID_STORAGE_CLASS
INVALID_SUBPOOL_TOKEN
NO_GET_LENGTH
NOT_SUBPOOL_OWNER
SUBPOOL_LOCK_FAILED
SUBPOOL_UNLOCK_FAILED

ADDRESS | ADDRESS_64
ADDRESS
The 31-bit address of the new element.
ADDRESS_64
The 64-bit address of the new element.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ELEMENT_LENGTH
Optional Parameter

The actual length of the new element (after it is rounded up to a multiple of the boundary for the subpool).

SMGF gate, INQUIRE_ELEMENT_LENGTH function

The INQUIRE_ELEMENT_LENGTH function of the SMGF gate is used to return the length of an element of storage whose address is known.

Input Parameters

ADDRESS
is the address of the element under inquiry.

STORAGE_CLASS
Optional Parameter

identifies the class of storage that is under inquiry.

Values for the parameter are:
CICS
CICS24
TASK
TASK24
TASK31
USER
USER24

SUBPOOL_TOKEN
Optional Parameter

is a token identifying the subpool within which the element is allocated.

LOCK_POOL
Optional Parameter

Indicates if access to the subpool is to be controlled by a lock.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_ERROR
LOOP
UNLOCK_ERROR

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_STORAGE_CLASS
INVALID_SUBPOOL_TOKEN
SUBPOOL_LOCK_FAILED
SUBPOOL_UNLOCK_FAILED

ELEMENT_LENGTH
is the length of the element.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMK7 gate, ADD_TRUSTED_EXTENSION function

The ADD_TRUSTED_EXTENSION function of the SMK7 gate is used to add a secure storage extension to a parent control block.

Input Parameters

PARENT_BLOCK
Address of the parent control block, and the offset to the secure extension pointer.

BLOCK_TYPE
Name of the parent control block.

SUBPOOL_TOKEN
Subpool token identifying the secure extension.

CONTENT
Data to be added to the secure extension.

Output Parameters

EXTENSION
The address and length of the secure extension.

REASON
The following value is returned when RESPONSE is EXCEPTION:
```
NOT_AUTHORIZED
```
The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
INSUFFICIENT_STORAGE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMK7 gate, DELETE_TRUSTED_EXTENSION function

The DELETE_TRUSTED_EXTENSION function of the SMK7 gate is used to delete a secure storage extension from a parent control block.

Input Parameters

BLOCK_TYPE
Name of the parent control block.

EXTENSION
Address and length of the parent control block.

PARENT_BLOCK
Address of the parent control block, and the offset to the secure extension pointer.

SUBPOOL_TOKEN
Subpool token identifying the secure extension.

Output Parameters

EXTENSION
Address and length of the secure extension.

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
INSUFFICIENT_STORAGE
LOOP
```
The following value is returned when RESPONSE is EXCEPTION:
```
NOT_AUTHORIZED
```
The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMMC gate, FREEMAIN function

The FREEMAIN function of the SMMC gate is used to release an element of storage.

Input Parameters

ADDRESS | ADDRESS_64

ADDRESS
The address of the element to be released, for an element in 24-bit or 31-bit storage.

ADDRESS_64
The address of the element to be released, for an element in 64-bit storage.

CALLER
Optional Parameter

Indicates the caller of the function.

Values for the parameter are as follows:
EXEC
MACRO
SYSTEM

EXEC_KEY
Optional Parameter

The execution key of the program that issued the EXEC FREEMAIN request.

Values for the parameter are as follows:
CICS
USER

REMARK
Optional Parameter

An optional 8-character field that identifies the FREEMAIN operation for problem determination. This field is highlighted when the FREEMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being released.

STORAGE_CLASS
Optional Parameter

Identifies the class of storage that is released.

Values for the parameter are as follows:
CICS
CICS24
CICS24_SAA
CONTROL
LINE
SHARED_CICS
SHARED_CICS24
SHARED_CICS24_SAA
SHARED_USER
SHARED_USER24
TACLE
TASK
TASK24
TASK31
TEMPSTG
TERMINAL
TERMINAL24
TRANSDATA

USER
USER24

TCTTE_ADDRESS
Optional Parameter

An optional field that must be specified if the FREEMAIN request is for storage of class LINE, TERMINAL, or TERMINAL24.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
DEACTIVATE_FAILURE

The following values are returned when RESPONSE is EXCEPTION:
INVALID_EXEC_KEY

The following values are returned when RESPONSE is INVALID:
INVALID_ADDRESS
INVALID_CLASS_ADDRESS
NO_TCTTE_ADDRESS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMMC gate, FREEMAIN_ALL_TERMINAL function

The FREEMAIN_ALL_TERMINAL function of the SMMC gate is used to release all terminal storage for a terminal.

Input Parameters

TCTTE_ADDRESS
The address of the terminal control table terminal entry (TCTTE) to release the storage for.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
ACTIVATE_FAILURE
DEACTIVATE_FAILURE
LOOP
STORAGE_VIOLATION

The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_DSA_NAME
NO_TRANSACTION_ENVIRONMENT

The following values are returned when RESPONSE is EXCEPTION or INVALID:
INVALID_ADDRESS
INVALID_CLASS_ADDRESS
INVALID_EXEC_KEY
INVALID_FUNCTION
INVALID_GET_LENGTH
INVALID_STORAGE_CLASS
NO_TCTTE_ADDRESS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMMC gate, GETMAIN function

The GETMAIN function of the SMMC gate is used to allocate an element of storage.

Input Parameters

GET_LENGTH

The length of the storage requested.

STORAGE_CLASS

Identifies the class of storage that is being allocated.

Values for the parameter are as follows:

```
CICS
CICS24
CICS24_SAA
CONTROL
LINE
SHARED_CICS
SHARED_CICS24
SHARED_CICS24_SAA
SHARED_USER
SHARED_USER24
TACLE
TASK
TASK24
TASK31
TEMPSTG
TERMINAL
TERMINAL24
TRANSDATA
USER
USER24
```

SUSPEND

If there is insufficient storage to satisfy the request, SUSPEND(YES) suspends the caller until the request can be satisfied, and SUSPEND(NO) sets REASON to INSUFFICIENT_STORAGE.

Values for the parameter are as follows:

```
NO
YES
```

CALLER

Optional Parameter

Indicates the caller of the function.

Values for the parameter are as follows:

```
EXEC
MACRO
SYSTEM
```

INITIAL_IMAGE

Optional Parameter

An optional byte value to which every byte in the new element is set.

REMARK

Optional Parameter

An optional 8-character field that identifies the GETMAIN operation for problem determination. This field is highlighted when the GETMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being obtained.

TCTTE_ADDRESS
Optional Parameter

An optional field that must be specified for GETMAIN requests for the TERMINAL storage class.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ACTIVATE_FAILURE
```

The following values are returned when RESPONSE is EXCEPTION:
```
INSUFFICIENT_STORAGE
INVALID_GET_LENGTH
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_STORAGE_CLASS
INVALID_CLASS_ADDRESS
NO_TCTTE_ADDRESS
```

ADDRESS | ADDRESS_64

ADDRESS
The address of the new element if it is in 24-bit or 31-bit storage.

ADDRESS_64
The address of the new element if it is in 64-bit storage.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMMC gate, INITIALISE function

The INITIALISE function of the SMMC gate is used to perform macro-compatibility interface initialization.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
ACTIVATE_FAILURE
DEACTIVATE_FAILURE
LOOP
STORAGE_VIOLATION
```

The following values are returned when RESPONSE is EXCEPTION:
```
INSUFFICIENT_STORAGE
```

The following values are returned when RESPONSE is EXCEPTION:
```
INVALID_ADDRESS
INVALID_EXEC_KEY
INVALID_FUNCTION
INVALID_GET_LENGTH
INVALID_STORAGE_CLASS
NO_TCTTE_ADDRESS
```

The following values are returned when RESPONSE is EXCEPTION:
```
INVALID_DSA_NAME
NO_TRANSACTION_ENVIRONMENT
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMMC gate, INQUIRE_ELEMENT_LENGTH function

The INQUIRE_ELEMENT_LENGTH function of the SMMC gate is used to obtain the start address and length of the storage element that contains the address that was specified on the input to the call. This function searches only the task-lifetime storage of the current task for the required storage element.

Input Parameters

ADDRESS | ADDRESS_64

ADDRESS
The storage address to inquire on. The address must be in an element of task-lifetime storage of the current task.

ADDRESS_64
The 64-bit storage address to inquire on. The address must be in an element of task-lifetime storage of the current task.

Output Parameters

ELEMENT_LENGTH
The length of the new element (after it is rounded up to a multiple of the boundary for the subpool).

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_ADDRESS
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_ADDRESS_ELEMENT
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ELEMENT_ADDRESS | ELEMENT_ADDRESS_64
Optional Parameter

ELEMENT_ADDRESS
The start address of the element that contains the input address.

ELEMENT_ADDRESS_64
The 64-bit start address of the element that contains the input address.

SMMC gate, INQUIRE_TASK_STORAGE function

The INQUIRE_TASK_STORAGE function of the SMMC gate is used to obtain details of all the task-lifetime storage that is associated with the specified task or the current task (depending whether the input parameter TRANSACTION_NUMBER is supplied).

Input Parameters

ELEMENT_BUFFER
A buffer into which the storage manager returns the start addresses of all the task-lifetime storage of the specified task.

LENGTH_BUFFER
A buffer into which the storage manager returns the lengths of all the task-lifetime storage of the specified task.

DSA_NAME
Optional Parameter

The DSA that contains the storage elements under inquiry.

TRANSACTION_NUMBER
Optional Parameter

The transaction to obtain storage details about. If this parameter is omitted, this function returns storage details of the current task.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INSUFFICIENT_STORAGE
INVALID_DSA_NAME
NO_TRANSACTION_ENVIRONMENT
```

NUMBER_OF_ELEMENTS

The number of storage elements in each buffer.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, INQ_TRANSACTION_ISOLATION function

The INQUIRE_TRANSACTION_ISOLATION function of the SMSR gate is used to inquire whether transaction isolation is active in the CICS region. This value is initially set by the **TRANISO** system initialization parameter.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANSACTION_ISOLATION

Indicates whether CICS operates with transaction isolation.

Values for the parameter are:

```
ACTIVE
INACTIVE
```

SMSR gate, INQUIRE_ACCESS function

The INQUIRE_ACCESS function of the SMSR gate is used to return the access key of an element of storage, and optionally, the dynamic storage area (DSA) that contains the storage element.

Input Parameters

ELEMENT_ADDRESS

The start address of the storage element.

ELEMENT_LENGTH

The length of the storage element.

ACCESS_TOKEN

Optional Parameter

The access token for the element of storage (returned by the INQUIRE_ACCESS_TOKEN function).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

INVALID_ELEMENT

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

ACCESS
The type of access for the storage element.

Values for the parameter are as follows:
CICS
READ_ONLY
USER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSA_EXTENT_END
Optional Parameter

The end address of the DSA extent that contains the input address.

DSA_EXTENT_START
Optional Parameter

The start address of the DSA extent that contains the input address.

DSA_NAME
Optional Parameter

The name of the dynamic storage area (DSA) that contains the storage element.

Values for the parameter are as follows:
CDSA
ECDSA
ERDSA
ESDSA
ETDSA
EUDSA
RDSA
SDSA
UDSA

SMSR gate, INQUIRE_ACCESS_TOKEN function

The INQUIRE_ACCESS_TOKEN function of the SMSR gate is used to return the access token for a storage element in 24-bit or 31-bit storage, for use with the SMSR INQUIRE_ACCESS function.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

ACCESS_TOKEN
The access token for the storage element. This token can be used with SMSR INQUIRE_ACCESS calls.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, INQUIRE_DSA_LIMIT function

The INQUIRE_DSA_LIMIT function of the SMSR gate is used to return the total storage limit for the DSAs that reside in 24-bit storage, and the total storage limit for the extended dynamic storage area (EDSA); that is, the DSAs that reside in 31-bit storage.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSA_LIMIT

Optional Parameter

The total storage limit for the DSAs that reside in 24-bit storage (below 16 MB, also known as below the line).

EDSA_LIMIT

Optional Parameter

The total storage limit for the EDSA; that is, the total storage for the DSAs that reside in 31-bit storage (above 16 MB but below 2 GB, also known as above the line).

SMSR gate, INQUIRE_DSA_SIZE function

The INQUIRE_DSA_SIZE function of the SMSR gate is used to return the size of the specified CICS dynamic storage area (DSA).

Input Parameters

DSA_NAME

The name of the DSA in 24-bit or 31-bit storage whose size is being inquired on.

Values for the parameter are as follows:

```
CDSA
ECDSA
ERDSA
ESDSA
ETDSA
EUDSA
RDSA
SDSA
UDSA
```

GDSA_NAME

Optional parameter

The name of the DSA in 64-bit storage whose size is being inquired on.

Values for the parameter are as follows:

```
GCDSA
GSDSA
GUDSA
```

Output Parameters

DSA_SIZE

The current size of the specified DSA.

GDSA_SIZE

The current size of the specified DSA in 64-bit storage.

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, INQUIRE_ISOLATION_TOKEN function

The INQUIRE_ISOLATION_TOKEN function of the SMSR gate is used to return an isolation token that can be used on SWITCH_SUBSPACE calls.

Output Parameters

ISOLATION_TOKEN
An isolation token that can be used on SWITCH_SUBSPACE calls.

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, INQUIRE_REENTRANT_PROGRAM function

The INQUIRE_REENTRANT_PROGRAM function of the SMSR gate is used to return whether the read-only DSAs, RDSA and ERDSA, are allocated from read-only key-0 protected storage, or CICS-key storage. This value is initially set by the **RENTPGM** system initialization parameter.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

REENTRANT_PROGRAM
Indicates whether the dynamic storage read-only DSAs are allocated from read-only key-0 protected storage

Values for the parameter are as follows:
NOPROTECT
PROTECT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, INQUIRE_SHORT_ON_STORAGE function

The INQUIRE_SHORT_ON_STORAGE function of the SMSR gate is used to return values that indicate whether CICS is currently short on storage below 16 MB, above 16 MB but below 2 GB, or above the bar.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOS_ABOVE_THE_BAR
Optional parameter

Indicates whether CICS is short on 64-bit (above-the-bar) storage.

Values for the parameter are as follows:
NO
YES

SOS_ABOVE_THE_LINE
Indicates whether CICS is short on storage above 16 MB but below 2 GB (above the line).

Values for the parameter are as follows:
NO
YES

SOS_BELOW_THE_LINE
Indicates whether CICS is short on storage below 16 MB (below the line).

Values for the parameter are as follows:
NO
YES

SMSR gate, INQUIRE_STORAGE_PROTECT function

The INQUIRE_STORAGE_PROTECT function of the SMSR gate is used to return the current value of the storage protection option.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STORAGE_PROTECT
The current setting of the storage protection option.

Values for the parameter are as follows:
NO
YES

SMSR gate, SET_DSA_LIMIT function

The SET_DSA_LIMIT function of the SMSR gate is used to set the total storage limit for the DSAs that reside in 24-bit storage (below 16 MB, also known as below the line) and the total storage limit for the EDSA, that is, the DSAs that reside in 31-bit storage (above 16 MB but below 2 GB, also known as above the line).

Input Parameters

DSA_LIMIT
Optional Parameter

The DSA storage limit required.

EDSA_LIMIT
Optional Parameter

The EDSA storage limit required.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_DSA_LIMIT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, SET_DSA_SIZE function

The SET_DSA_SIZE function of the SMSR gate is used to set the size of the CICS dynamic storage areas (DSAs) that reside in 24-bit storage (below 16 MB, also known as below the line) and the DSAs that reside in 31-bit storage (above 16 MB but below 2 GB, also known as above the line).

Input Parameters

DSA_NAME
The name of the DSA to set the size for.

Values for the parameter are as follows:
CDSA
ECDSA
ERDSA
ESDSA
EUDSA
RDSA
SDSA
UDSA

DSA_SIZE
The size of the DSA.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_DSA_SIZE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, SET_REENTRANT_PROGRAM function

The SET_REENTRANT_PROGRAM function of the SMSR gate is used to set whether the read-only DSAs, RDSA and ERDSA, are allocated from read-only key-0 protected storage or CICS-key storage. This value is initially set by the **RENTPGM** system initialization parameter.

Input Parameters

REENTRANT_PROGRAM
The reentrant program option for the RDSA and the ERDSA.

Values for the parameter are as follows:
NOPROTECT
PROTECT

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, SET_STORAGE_PROTECT function

The SET_STORAGE_PROTECT function of the SMSR gate is used to set the storage protection option.

Input Parameters

STORAGE_PROTECT

A value that indicates whether storage protection is required.

Values for the parameter are as follows:

NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, SET_STORAGE_RECOVERY function

The SET_STORAGE_RECOVERY function of the SMSR gate is used to set the storage recovery option.

Input Parameters

RECOVERY

The value to which the storage recovery option is to be set.

Values for the parameter are as follows:

NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, SET_TRANSACTION_ISOLATION function

The SET_TRANSACTION_ISOLATION function of the SMSR gate is used to set whether or not you want transaction isolation in your CICS region. This value is initially set by the **TRANISO** system initialization parameter.

Input Parameters

TRANSACTION_ISOLATION

Indicates whether or not transaction isolation is active in your CICS region.

Values for the parameter are as follows:

```
ACTIVE
INACTIVE
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, SWITCH_SUBSPACE function

The SWITCH_SUBSPACE function of the SMSR gate is used to change the subspace of a task.

Input Parameters

SPACE

The type of subspace you want this task to execute in.

Values for the parameter are as follows:

```
BASESPACE
RESET_SPACE
SUBSPACE
```

ISOLATION_TOKEN

Optional Parameter

An isolation token that can be returned from an INQUIRE_ISOLATION_TOKEN call.

TRANSACTION_TOKEN

Optional Parameter

A transaction manager token (which can be returned from an XMIQ INQUIRE_TRANSACTION_TOKEN call) that represents the task whose subspace you want to change.

WORK_REGISTER

Optional Parameter

A work register.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMSR gate, UPDATE_SUBSPACE_TCB_INFO function

The UPDATE_SUBSPACE_TCB_INFO function informs the storage manager of the deletion of open task control blocks (TCBs) that are associated with subspaces.

Input Parameters

OPEN_TCBS_DELETED
A 32-bit string that indicates each type of open TCB that has been deleted and that therefore is no longer associated with the subspace.

SUBSPACE_TOKEN
Indicates the subspace that is associated with the deleted TCBs.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

S2AD gate, ADD_SUBPOOL function

The ADD_SUBPOOL function of the S2AD gate is used to create a new subpool with given attributes.

Input Parameters

BOUNDARY
The boundary on which all elements within the subpool must be aligned. The boundary must be a power of two, in the range 8 through 4096.

ELEMENT_CHAIN
Indicates whether a chain of the addresses and lengths of the elements is kept.

Values for the parameter are as follows:

```
NO
YES
```

ELEMENT_TYPE
Indicates whether the subpool elements are of fixed or variable length.

Values for the parameter are as follows:

```
FIXED
VARIABLE
```

INITIAL_FREE
The size of the initial free storage area for the subpool.

LOCATION
Specifies whether all elements within the subpool are private or shared.

Values for the parameter are as follows:

```
PRIVATE
SHARED
```

SUBPOOL_NAME
The 8-character name by which the subpool is known.

USAGE
Indicates whether the subpool is for task or domain use.

Values for the parameter are as follows:
DOMAIN
TASK

ACCESS
Optional Parameter

The type of storage access required.

Values for the parameter are as follows:
CICS
USER

FIXED_LENGTH
Optional Parameter

The element length for a fixed-length subpool.

LOCK_POOL
Optional Parameter

Indicates whether access to the subpool is controlled by a lock.

Values for the parameter are:
NO
YES

STORAGE_CHECK
Optional Parameter

Indicates whether storage zone checking is enabled for this subpool.

Values for the parameter are as follows:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE

The following values are returned when RESPONSE is INVALID:
DUPLICATE_SUBPOOL_NAME
INVALID_FIXED_LENGTH
INVALID_INITIAL_FREE
INVALID_SUBPOOL_NAME
LOCK_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SUBPOOL_TOKEN
The token that identifies the newly created subpool.

DSA_NAME
Optional Parameter

The name of the CICS dynamic storage area (DSA) in which the subpool resides.

Values for the parameter are as follows:
GCDSA

S2AD gate, DELETE_SUBPOOL function

The DELETE_SUBPOOL function of the S2AD gate is used to delete a subpool.

Input Parameters

SUBPOOL_TOKEN
is the token identifying the subpool to be deleted.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_SUBPOOL_TOKEN
NOT_SUBPOOL_OWNER
SUBPOOL_NOT_EMPTY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

S2AD gate, END_SUBPOOL_BROWSE function

The END_SUBPOOL_BROWSE function of the S2AD gate is used to end a browse of the storage manager domain subpools.

Input Parameters

BROWSE_TOKEN
is the token identifying the browse operation.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

S2AD gate, GET_NEXT_SUBPOOL function

The GET_NEXT_SUBPOOL function of the S2AD gate is used in a storage manager domain subpool browse to get the next subpool.

Input Parameters

BROWSE_TOKEN
is the token identifying the browse operation.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SUBPOOL_NAME
is name of the subpool returned by the browse.

DSA_NAME
Optional Parameter

is the name of the CICS dynamic storage area (DSA) in which the subpool resides.

Values for the parameter are:
GCDSA

S2AD gate, INQUIRE_SUBPOOL function

The INQUIRE_SUBPOOL function of the S2AD gate is used to inquire about a storage manager domain subpool.

Input Parameters

SUBPOOL_NAME
is the 8-character name by which the subpool is known.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
SUBPOOL_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DSA_NAME
Optional Parameter

is the name of the CICS dynamic storage area (DSA) in which the subpool resides.

Values for the parameter are:
GCDSA

S2AD gate, START_SUBPOOL_BROWSE function

The START_SUBPOOL_BROWSE function of the S2AD gate is used to start a browse of the storage manager domain subpools.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE

BROWSE_TOKEN
is the token identifying the browse operation.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

S2GF gate, FREEMAIN function

The FREEMAIN function of the S2GF gate is used to release an element of storage within a subpool.

Input Parameters

ADDRESS
The 64-bit address of the element to release.

SUBPOOL_TOKEN
Optional Parameter

A token that identifies the subpool that contains the element to release.

FREE_LENGTH
Optional Parameter

The length of the element to release.

LOCK_POOL
Optional Parameter

Indicates whether access to the subpool is controlled by a lock.

Values for the parameter are as follows:
NO
YES

REMARK
Optional Parameter

An optional 8-character field that identifies the FREEMAIN operation for problem determination. This field is highlighted when the FREEMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being released.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
DEACTIVATE_FAILURE
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_ADDRESS
INVALID_FREE_LENGTH
INVALID_STORAGE_CLASS
INVALID_SUBPOOL_TOKEN
NO_FREE_LENGTH
NOT_SUBPOOL_OWNER
SUBPOOL_EMPTY
SUBPOOL_LOCK_FAILED
SUBPOOL_UNLOCK_FAILED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

S2GF gate, GETMAIN function

The GETMAIN function of the S2GF gate is used to allocate an element of storage from a subpool.

Input Parameters

SUBPOOL_TOKEN

Optional Parameter

A token that identifies the subpool within which the element is allocated.

SUSPEND

Specifies the action when there is insufficient storage to satisfy the request. When the value is YES, the caller is suspended until the request can be satisfied. When the value is NO, REASON is set to INSUFFICIENT_STORAGE.

Values for the parameter are as follows:

```
NO
YES
```

GET_LENGTH

Optional Parameter

The length of the storage requested.

INITIAL_IMAGE

Optional Parameter

An optional byte value to which each byte in the new element is set, up to a maximum of 2 GB.
Consider carefully whether to use this parameter with 64-bit storage, because system resources to support the allocated storage are required immediately, and auxiliary storage is required if storage needs to be paged out. To avoid using too much real memory, a maximum of 2 GB of storage is initialized, even if more than 2 GB of storage is requested.
If you do not specify an INITIAL_IMAGE parameter, minimal system resources are required until the requested 64-bit storage is used.

LOCK_POOL
Optional Parameter

Indicates whether access to the subpool is controlled by a lock.

Values for the parameter are as follows:
NO
YES

REMARK
Optional Parameter

An optional 8-character field that identifies the GETMAIN operation for problem determination. This field is highlighted when the GETMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being obtained.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
ACTIVATE_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE

The following values are returned when RESPONSE is INVALID:
INVALID_GET_LENGTH
INVALID_INITIAL_IMAGE
INVALID_STORAGE_CLASS
INVALID_SUBPOOL_TOKEN
NO_GET_LENGTH
NOT_SUBPOOL_OWNER
SUBPOOL_LOCK_FAILED
SUBPOOL_UNLOCK_FAILED

ADDRESS
The 64-bit address of the new element.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ELEMENT_LENGTH
Optional Parameter

The length of the new element.

S2GF gate, INQUIRE_ELEMENT_LENGTH function

The INQUIRE_ELEMENT_LENGTH function of the S2GF gate is used to return the length of an element of storage whose address is known. This function is for use with subpools that have element chaining.

Input Parameters

ADDRESS
The 64-bit address of the element under inquiry.

SUBPOOL_TOKEN
Optional Parameter

A token that identifies the subpool within which the element is allocated.

LOCK_POOL
Optional Parameter

Indicates whether access to the subpool is controlled by a lock.

Values for the parameter are as follows:
NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_STORAGE_CLASS
INVALID_SUBPOOL_TOKEN
SUBPOOL_LOCK_FAILED
SUBPOOL_UNLOCK_FAILED

ELEMENT_LENGTH

The length of the element.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

S2SR gate, INQUIRE_ACCESS function

The INQUIRE_ACCESS function of the S2SR gate is used to return the access key of a storage element in 64-bit storage and, optionally, the dynamic storage area (DSA) that contains the storage element.

Input Parameters

ELEMENT_ADDRESS

The start address of the storage element in 64-bit storage.

ELEMENT_LENGTH

The length of the storage element in 64-bit storage.

ACCESS_TOKEN

Optional Parameter

The access token for the storage element (returned by the S2SR INQUIRE_ACCESS_TOKEN function).

Output Parameters

ACCESS

The type of access for the storage element.

Values for the parameter are as follows:
CICS

DSA_EXTENT_END

Optional Parameter

The end address of the DSA extent that contains the input address.

DSA_EXTENT_START

Optional Parameter

The start address of the DSA extent that contains the input address.

DSA_NAME

Optional Parameter

The name of the dynamic storage area (DSA) that contains the storage element.

Values for the parameter are as follows:

GCDSA

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

INVALID_ELEMENT

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

S2SR gate, INQUIRE_ACCESS_TOKEN function

The INQUIRE_ACCESS_TOKEN function of the S2SR gate is used to return the access token for a storage element in 64-bit storage, for use with the S2SR INQUIRE_ACCESS function.

Output Parameters

ACCESS_TOKEN

The access token for the element in 64-bit storage. This token can be used with S2SR INQUIRE_ACCESS calls.

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is INVALID:

INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

S2SR gate, INQUIRE_MEMLIMIT function

The INQUIRE_MEMLIMIT function of the S2SR gate is used to return the value of the z/OS **MEMLIMIT** parameter; that is, the total amount of 64-bit (above-the-bar) storage that the CICS address space can use. This storage includes the above-the-bar DSAs (GDSA) and MVS storage in the CICS region outside the GDSA.

Output Parameters

MEMLIMIT

The total amount of 64-bit (above-the-bar) storage that the CICS address space can use. This storage includes the CICS dynamic storage areas above the bar (collectively called the GDSA) and MVS storage in the CICS region outside the GDSA.

MEMLIMIT_SOURCE

The origin of the z/OS **MEMLIMIT** value. Values for the parameter are as follows:

SMF
JCL
REG0
USI
AUTH
URG

For more information about these parameters, see Limiting the use of memory objects in the z/OS MVS Programming: Extended Addressability Guide.

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Storage manager domain's generic gates

Table 47 summarizes the generic gates of the domaqin. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 47. Storage manager domain's generic gates

Gate	Trace	Functions	Format
DMDM	SM 0101 SM 0102	PRE_INITIALISE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
SMVN	SM 1401 SM 1402	DSAT_TASK_REPLY DSAT_PURGE_INHIBIT_QUERY	DSAT
STST	SM 0A01 SM 0A02	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

In preinitialization processing, the storage manager domain sets the initial storage options:

- The amount of storage to be allocated to the dynamic storage area
- The amount of storage to be allocated to the extended dynamic storage area
- The storage recovery option
- The state of the storage protect, transaction isolation and the reentrant program option.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

Storage manager domain also issues console messages during preinitialization to report the amount of storage allocated above and below the line for DSA use.

In initialization, quiesce, and termination processing, the storage manager domain performs only internal routines.

Storage manager domain's generic formats

Table 48 on page 1097 describes the generic formats owned by the application domain and shows the functions performed on the calls.

Table 48. Storage manager domain's generic formats

Format	Calling module	Function
SMNT	DFHSMSY	STORAGE_NOTIFY
	DFHSMVN	MVS_STORAGE_NOTIFY

In the descriptions of the formats, the input parameters are input not to the Storage manager domain, but to the domain being called by the Storage manager domain. Similarly, the output parameters are output by the domain that was called by the Storage manager domain, in response to the call.

SMNT gate, MVS_STORAGE_NOTIFY function

The MVS_STORAGE_NOTIFY function of SMNT format is used to notify a domain when use of MVS storage becomes excessive, or MVS storage is no longer constrained.

If use of MVS storage becomes excessive, the target domain can take action to release MVS storage or to limit its future MVS storage requirements. There is a notification for a breach of the threshold value for MVS storage, and a different notification for a breach of the reserved MVS storage cushion, which is a more serious condition.

If MVS storage is no longer constrained, the domain can return to normal operation.

Input Parameters

CUSHION

The status of the reserved MVS storage cushion since the last time the SM domain issued an MVS_STORAGE_NOTIFY call.

Values for the parameter are as follows:

NEWLY_BREACHED. The storage cushion has been partially released to satisfy requirements for MVS storage.

NEWLY_RESTORED. CICS has reallocated the reserved storage cushion.

UNCHANGED. The state of the storage cushion has not changed. The storage cushion is still partially released, or still intact.

THRESHOLD

The MVS storage requirements in relation to the threshold value for MVS storage since the last time the SM domain issued an MVS_STORAGE_NOTIFY call.

Values for the parameter are as follows:

NEWLY_BREACHED. MVS storage requirements have increased above the threshold value.

NEWLY_RESTORED. MVS storage requirements have decreased below the threshold value.

UNCHANGED. MVS storage requirements have not changed in relations to the threshold value. The MVS storage requirements are still above the threshold or still below the threshold.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND

LOCK_FAILED

LOOP

RESUME_FAILURE

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SMNT gate, STORAGE_NOTIFY function

The STORAGE_NOTIFY function of SMNT format is used to notify free storage in each dynamic storage area (DSA).

Input Parameters

ALMOST_SOS_ABOVE
Indicates whether a DSA above 16 MB but below 2 GB (above the line) could go short on storage (SOS) imminently; that is, a single GETMAIN request might cause an SOS condition.

Values for the parameter are as follows:
NO
YES

ALMOST_SOS_ABOVE_BAR
Indicates whether a DSA above the bar could go short on storage (SOS) imminently; that is, a single GETMAIN request might cause an SOS condition.

Values for the parameter are as follows:
NO
YES

DSAS_CONSTRAINED
Indicates whether any DSA below the 16 MB line, or any DSA above 16 MB but below 2 GB (above the line), is currently constrained because of lack of free storage.

Values for the parameter are as follows:
NO
YES

FREE_BYTES_CDSA
The largest free area available (in bytes) in the CICS DSA below the 16 MB line (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

FREE_BYTES_ECDSA
The largest free area available (in bytes) in the extended CICS DSA above 16 MB but below 2 GB (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

FREE_BYTES_ERDSA
The largest free area available (in bytes) in the extended read-only DSA above 16 MB but below 2 GB (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

FREE_BYTES_ESDSA
The largest free area available (in bytes) in the extended shared user-key DSA above 16 MB but below 2 GB (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

FREE_BYTES_EUDSA
The largest free area available (in bytes) in the extended user-key DSA above 16 MB but below 2 GB (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

FREE_BYTES_RDSA
The largest free area available (in bytes) in the read-only DSA below the 16 MB

line (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

FREE_BYTES_SDSA
The largest free area available (in bytes) in the shared user-key DSA below the 16 MB line (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

FREE_BYTES_UDSA
The largest free area available (in bytes) in the user-key DSA below the 16 MB line (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

FREE_MBYTES_GCDSA
The largest free area available (in MB) in the above-the-bar CICS DSA (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

FREE_MBYTES_GUDSA
The largest free area available (in MB) in the above-the-bar user-key DSA (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

FREE_MBYTES_GSDSA
The largest free are available (in MB) in the above-the-bar shared user-key DSA (not including the cushion). If the cushion is in use, or a task in this DSA is suspended because of a storage condition, this value is 0.

GDSAS_CONSTRAINED
Indicates whether any DSA above the bar is currently constrained because of lack of free storage.

Values for the parameter are as follows:
NO
YES

CDSA_FIXED
Optional Parameter

Indicates whether the CICS DSA below the 16 MB line is fixed in size.

Values for the parameter are as follows:
NO
YES

ECDSA_FIXED
Optional Parameter

Indicates whether the extended CICS DSA (above 16 MB but below 2 GB) is fixed in size.

Values for the parameter are as follows:
NO
YES

ERDSA_FIXED
Optional Parameter

Indicates whether the extended read-only CICS DSA (above 16 MB but below 2 GB) is fixed in size.

Values for the parameter are as follows:
NO
YES

ESDSA_FIXED
Optional Parameter

Indicates whether extended shared user-key DSA (above 16 MB but below 2 GB) is fixed in size.

Values for the parameter are as follows:
NO
YES

EUDSA_FIXED
Optional Parameter

Indicates whether the extended user-key DSA (above 16 MB but below 2 GB) is fixed in size.

Values for the parameter are as follows:
NO
YES

GCDSA_FIXED
Optional Parameter

Indicates whether the above-the-bar CICS DSA is fixed in size.

Values for the parameter are as follows:
NO
YES

GSDSA_FIXED
Optional Parameter

Indicates whether the above-the-bar shared user-key DSA is fixed in size

Values for the parameter are as follows:
NO
YES

GUDSA_FIXED
Optional Parameter

Indicates whether the above-the-bar user-key DSA is fixed in size.

Values for the parameter are as follows:
NO
YES

RDSA_FIXED
Optional Parameter

Indicates whether the read-only CICS DSA below the 16 MB line is fixed in size.

Values for the parameter are as follows:
NO
YES

SDSA_FIXED
Optional Parameter

Indicates whether the shared user-key DSA below the 16 MB line is fixed in size.

Values for the parameter are as follows:
NO
YES

UDSA_FIXED
Optional Parameter

Indicates whether the user-key DSA below the 16 MB line is fixed in size.

Values for the parameter are as follows:
NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHSMAD	Handles the following requests: ADD_SUBPOOL DELETE_SUBPOOL END_SUBPOOL_BROWSE GET_NEXT_SUBPOOL INQUIRE_SUBPOOL START_SUBPOOL_BROWSE
DFHSMAR	Handles the following requests: ALLOCATE_TRANSACTION_STG RELEASE_TRANSACTION_STG Service routines for add subpool requests.
DFHSMCK	Handles the following requests: CHECK_STORAGE RECOVER_STORAGE
DFHSMDM	Handles the following requests: INITIALIZE_DOMAIN PRE_INITIALIZE QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHSMDUF	SM domain offline dump formatting routine
DFHSMGF	Handles the following requests: FREEMAIN GETMAIN INQUIRE_ELEMENT_LENGTH
DFHSMMCI	SM domain macro-compatibility interface INITIALISE function
DFHSMMC2	SM domain macro-compatibility interface that handles the following requests: FREEMAIN_ALL_TERMINAL INQUIRE_ELEMENT_LENGTH INQUIRE_TASK_STORAGE
DFHSMMF	SM domain macro-compatibility interface FREEMAIN function
DFHSMMG	SM domain macro-compatibility interface GETMAIN function
DFHSMNT	Handles the following request: STORAGE_NOTIFY

Module	Function
DFHSMSR	Handles the following requests: INQUIRE_ACCESS INQUIRE_ACCESS_TOKEN INQUIRE_DSA_LIMIT INQUIRE_DSA_SIZE INQUIRE_ISOLATION_TOKEN INQUIRE_REENTRANT_PROGRAM INQUIRE_SHORT_ON_STORAGE INQUIRE_STORAGE_PROTECT INQUIRE_TRANSACTION_ISOLATION SET_DSA_LIMIT SET_DSA_SIZE SET_REENTRANT_PROGRAM SET_STORAGE_PROTECT SET_STORAGE_RECOVERY SET_TRANSACTION_ISOLATION SWITCH_SUBSPACE UPDATE_SUBSPACE_TCB_INFO
DFHSMST	Handles the following requests: COLLECT_RESOURCE_STATS COLLECT_STATISTICS
DFHSMSVC	Gets DSAs
DFHSMSY	SM domain system task: issues STORAGE_NOTIFY requests
DFHSMTRI	Interprets SM domain trace entries
DFHSMVN	SM domain system task: issues MVS_STORAGE_NOTIFY requests
DFHSMVP	Detects and manages MVS storage constraints
DFHS2AD	Handles the following requests: ADD_SUBPOOL DELETE_SUBPOOL END_SUBPOOL_BROWSE GET_NEXT_SUBPOOL INQUIRE_SUBPOOL START_SUBPOOL_BROWSE
DFHS2GF	Handles the following requests: FREEMAIN GETMAIN INQUIRE_ELEMENT_LENGTH
DFHS2MC2	SM domain macro-compatibility interface that handles the following request for 64-bit classes: INQUIRE_ELEMENT_LENGTH
DFHS2MF	SM domain macro-compatibility interface FREEMAIN function for 64-bit storage
DFHS2MG	SM domain macro-compatibility interface GETMAIN function for 64-bit storage
DFHS2PP	SM domain pagepool manager functions
DFHS2SQ	SM domain suspend queue manager function
DFHS2SR	Handles the following requests: INQUIRE_ACCESS INQUIRE_ACCESS_TOKEN INQUIRE_MEMLIMIT

Sockets domain (SO)

The sockets domain provides TCP/IP services to CICS. It includes a TCP/IP listener system task, the TCPIPSERVICE resource to manage the listener, and domain gates to operate on a TCP/IP connection.

Sockets domain's specific gates

The specific gates provide access for other domains to functions that are provided by the SO domain.

SOAD gate, ADD_REPLACE_TCPIPSERVICE function

The ADD_REPLACE_TCPIPSERVICE function is called at RDO time to install a TCPIPSERVICE resource. If the status is OPEN then the service is also opened using the SORD REGISTER function. A catalog entry is written to record the installed resource.

Input Parameters

BACKLOG
The value of the backlog parameter passed to the TCP/IP listen function for this service. It specifies how many connection requests TCP/IP will queue for this service.

IPADDRESS
The specific IP address that the listener will bind to for this service.

MAXDATA_LENGTH
The maximum length of data that may be received by CICS.

PORTNUMBER
The port number to listen on.

SOCKETCLOSE
The interval from the time of the initial receive request for incoming data, after which CICS is to time out the socket.

SSL
Whether or not connections to this service are to be secured using the Secure Sockets Layer (SSL) protocols.

Values for the parameter are:

```
CLIENTAUTH
NO
YES
ATTLSAWARE
```

STATUS
The status of the service.

Values for the parameter are:

```
CLOSED
OPEN
```

TCPIPSERVICE_NAME
The name of the TCPIPSERVICE resource.

URM_NAME
The name of the user-replaceable program.

ATTACHSEC
Optional Parameter

The level of attach-time security required for TCP/IP connections to CICS Clients.

AUTHENTICATION
Optional Parameter

The authentication and identification scheme to be used for inbound TCP/IP connections.

Values for the parameter are:
ASSERTED
AUTOMATIC
AUTOREGISTER
BASIC
CERTIFICATE
KERBEROS
NONE

CERTIFICATE_LABEL
Optional Parameter

The name of a certificate within the keyfile that this service will use to authenticate itself to clients with, if the SSL protocol is used.

CIPHER_SUITES
Optional Parameter

A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

DNSGROUP
Optional Parameter

The group name with which CICS will register to Workload Manager, for connection optimization.

GRPCRITICAL
Optional Parameter

Whether or not the service is a critical member of the DNS group.

Values for the parameter are:
CRITICAL
NONCRITICAL

MAXPERSIST
Optional Parameter

The maximum number of persistent connections from web clients that the CICS region allows for this port at any one time.

NUMCIPHERS
Optional Parameter

The number of cipher suites specified in the CIPHER_SUITES parameter.

PRIVACY
Optional Parameter

The level of SSL encryption required for inbound connections to this service that is specified by the CIPHERS attribute.

Values for the parameter are:
NOTSUPPORTED
REQUIRED
SUPPORTED

PROTOCOL
Optional Parameter

The application level protocol used on the TCP/IP port.

TRANSACTION
Optional Parameter

The transaction ID of the transaction to attach for each connection to this service.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

CATALOG_ERROR

The following values are returned when RESPONSE is EXCEPTION:

AT_MAXSOCKETS
AUTHENTICATION_UNAVAILABLE
CERTIFICATE_INVALID
INVALID_NAME
INVALID_STATUS
PORT_IN_USE
PORT_NOTAUTH
SERVICE_OPEN
SSL_NOT_AVAILABLE
TCPIP_CLOSED
TCPIP_INACTIVE
UNKNOWN_IP_ADDRESS
UNSUPPORTED_CIPHER

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOAD gate, DELETE_TCPIPSERVICE function

The DELETE_TCPIPSERVICE function is called at RDO time to remove an installed tcpipservice definition. If the status is OPEN then the tcpipservice is not removed. The catalog entry is removed for the discarded resource.

Input Parameters

TCPIPSERVICE_NAME

is the name of the tcpipservice.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

NOT_FOUND
SERVICE_OPEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, ACCEPT function

Accept a new connection on a listening socket.

Input Parameters

SOCKET_TOKEN

A token that is generated when a socket is created, and is used subsequently to identify the socket.

On this function, the token identifies the listening socket.

LIFETIME

Optional Parameter

The lifetime of the socket.

Values for the parameter are:

PERSISTENT
SHARED

TASK

TIMEOUT_VALUE
Optional Parameter

The interval after which a request will time out.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:

ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:

TASK_CANCELLED
TIMED_OUT

CLIENT_SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

On this function, the token identifies the connection that has been accepted. On subsequent requests, the token is passed on the **SOCKET_TOKEN** parameter.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, BIND function

Bind a socket to an IP address and port number.

Input Parameters

IP_ADDRESS

Optional Parameter

The binary IP address of the target.

PORT

Optional Parameter

The binary port number of the target.

SOCKET_TOKEN

A token that is generated when a socket is created, and is used subsequently to identify the socket.

MINIMUM_DATA_LENGTH

Optional Parameter

The minimum amount of data that must be received before the request is considered to be complete.

STRING_PORT

Optional Parameter

The port number of the target, expressed as a string.

TIMEOUT_VALUE

Optional Parameter

The interval after which a request will time out.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN
```

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:

TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, CANCEL function

Cancel any outstanding asynchronous input or output on a socket.

Input Parameters

SOCKET_TOKEN

A token that is generated when a socket is created, and is used subsequently to identify the socket.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:

ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:

TASK_CANCELLED

TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, CLOSE function

The CLOSE function is called to close the socket connection to the TCP/IP client.

Input Parameters

CONDITIONAL
Optional Parameter

A binary value indicating whether a request to close a socket is conditional. A conditional request to close the socket will fail if the socket is in use.

Values for the parameter are:
NO
YES

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

Output Parameters

SWITCH_TCB
Optional parameter

A binary parameter that specifies whether the socket domain should switch to the socket TCB.

Values of the parameter are:
YES
NO

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED

SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, CONNECT function

Connect a socket to another host and port.

Input Parameters

SWITCH_TCB
Optional parameter

Specifies if a switch to a socket TCB is required.

Values of the parameter are:
NO
YES

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

CERTIFICATE_LABEL
Optional Parameter

The label of an X.509 certificate that is used during the SSL handshake for the connection.

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the **CIPHER_SUITES** parameter.

CIPHER_SUITES
Optional Parameter

A string of up to 56 hexadecimal digits that encodes a list of up to 28 2-digit cipher suite codes.

IP_ADDRESS
Optional Parameter

The binary IP address of the target.

PORT
Optional Parameter

The binary port number of the target.

SSL
Optional Parameter

A binary parameter that specifies whether the socket supports the secure sockets layer (SSL).

Values for the parameter are:
NO
YES

MINIMUM_DATA_LENGTH
Optional Parameter

The minimum amount of data that must be received before the request is considered to be complete.

STRING_PORT
Optional Parameter

The port number of the target, expressed as a string.

TIMEOUT_VALUE
Optional Parameter

The interval after which a request will time out.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is PURGED:

```
TASK_CANCELLED
TIMED_OUT
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, CREATE function

This function creates a new socket.

Input Parameters

SWITCH_TCB

Optional parameter

Specifies if a switch to a socket TCB is required.

Values of the parameter are:

NO
YES

LIFETIME

Optional Parameter

The lifetime of the socket.

Values for the parameter are:

PERSISTENT
SHARED
TASK

QUEUE_TIMEOUT

Optional Parameter

A parameter that indicates whether a request to create a socket will be queued if no sockets can be created immediately, and whether the request will be queued for ever or will time out.

Values for the parameter are:

FOREVER
NO
YES

QUEUE_TIMEOUT

Optional Parameter

A parameter that indicates whether a request to create a socket will be queued if no sockets can be created immediately, and whether the request will be queued for ever or will time out.

Values for the parameter are:

FOREVER
NO
YES

TIMEOUT_VALUE

Optional Parameter

The interval after which a request will time out.

TRANSPORT

Optional Parameter

The type of IP transport supported by the socket.

Values for the parameter are:

TCP
UDP

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOCK_FAILURE

LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

SOCK gate, ESTABLISH function

This function associates the calling task with the socket.

Input Parameters

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

XM_STORE
Optional Parameter

A binary parameter that indicates whether the socket token is to be stored in the transaction's transaction manager block.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is PURGED:

```
TASK_CANCELLED
TIMED_OUT
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, GET_DATA_LENGTH function

Return the numbrr of bytes of data that can be read on the socket.

Input Parameters

SOCKET_TOKEN

A token that is generated when a socket is created, and is used subsequently to identify the socket.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_FAILURE
LOOP
```

SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:

ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:

TASK_CANCELLED
TIMED_OUT

BYTES_AVAILABLE

The number of bytes of data that are available to be read.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, GET_SOCKET_OPTS function

Return the attributes of a socket.

Input Parameters

SWITCH_TCB

Optional parameter

Specifies if a switch to a socket TCB is required.

Values of the parameter are:

NO
YES

SOCKET_TOKEN

A token that is generated when a socket is created, and is used subsequently to identify the socket.

LIFETIME

Optional Parameter

The lifetime of the socket.

Values for the parameter are:
PERSISTENT
SHARED
TASK

SO_LINGER
Optional Parameter

A sockets parameter that controls socket shutdown behavior, allowing the socket to shut down gracefully.

SO_REUSE_IP_ADDRESS
Optional Parameter

A binary parameter that specifies whether the socket can reuse an IP address.

Values for the parameter are:
NO
YES

TCP_NODELAY
Optional Parameter

A binary parameter that specifies whether to send small messages on the socket without buffering them first.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is PURGED:

```
TASK_CANCELLED
TIMED_OUT
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, LISTEN function

The LISTEN function is the main routine for the SO domain listener task CSOL. When the listener task starts it branches into the LISTEN function of the SOCK gate. This allows the listener code to be written at the domain level rather than the task level.

Input Parameters

BACKLOG
The value of the backlog parameter for the TCP/IP listen function for the current TCPIPSERVICE. It specifies how many connection requests TCP/IP will queue for the service.

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

TIMEOUT_VALUE
Optional Parameter

The interval after which a request will time out.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
```

SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, RECEIVE function

The RECEIVE function receives a buffer of data from a TCP/IP connected client.

Input Parameters

CALLBACK_GATE
Optional Parameter

The gate at which the domain that requested the function will be notified when the request is complete.

IP_ADDRESS
Optional Parameter

The binary IP address of the target.

STRING_IP_ADDRESS
Optional Parameter

The IP address of the target, expressed as a string.

MINIMUM_DATA_LENGTH
Optional Parameter

The minimum amount of data that must be received before the request is considered to be complete.

PEEK
Optional Parameter

A binary parameter that indicates whether the read request should look at data without removing it from the socket's receive buffer.

Values for the parameter are:
NO
YES

PEEK_BUFFER
Optional Parameter

The buffer in which peek data is returned when PEEK(YES) is specified.

Values for the parameter are:
NO
YES

PORT
Optional Parameter

The binary port number of the target.

RECEIVE_BUFFER
The buffer that receives the data.

RECEIVE_TYPE
Optional Parameter

A parameter that specifies whether a receive request is asynchronous or synchronous.

Values for the parameter are:
ASYNC
SYNC

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

STRING_IP_ADDRESS
Optional Parameter

The IP address of the target, expressed as a string.

STRING_PORT
Optional Parameter

The port number of the target, expressed as a string.

TIMEOUT
Optional Parameter

Specifies how the timeout interval is determined. If the parameter is not specified, or if TIMEOUT(SOCKETCLOSE) is specified, the timeout is taken from the TCPIPSERVICE definition. If TIMEOUT(SCLOSELIM) is specified, the timeout is the lower value between the TCPIPSERVICE **SOCKETCLOSE** value and 30 seconds. If TIMEOUT(DEFAULT) is specified, the timeout is 30 seconds.

Values for the parameter are:
DEFAULT
SCLOSELIM
SOCKETCLOSE

TIMEOUT_VALUE
Optional Parameter

The interval after which a request will time out.

USER_TOKEN
Optional Parameter

A token that the caller supplies to identify the request. The token is returned to the user at the callback gate when the request is complete.

Output Parameters

SSL
Optional output parameter SSL added:

A binary parameter that specifies whether the socket that received the data is an SSL socket.

Values of the parameter are:
YES
NO

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:

ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, RECEIVE_SSL_DATA function

The RECEIVE_SSL_DATA function is called to receive data from a connected TCP/IP client if the connection is secured using SSL.

Input Parameters

RECEIVE_BUFFER
The buffer that receives the data.

SOCKET_ADDR
The address of the socket.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR

CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, RELINQUISH function

Relinquish a task's association with a persistent socket.

Input Parameters

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR

MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, RESERVE function

Reserve a task's association with a persistent socket.

Input Parameters

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

TRANNUM
The transaction number of the task.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE

NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, SCHEDULE_RECEIVER_TASK function

Schedule a new receiver task to be attached.

Input Parameters

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR

STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, SEND function

The SEND function sends a buffer of data to a connected TCP/IP client.

Input Parameters

SEND_BUFFER
The buffer of data to be sent.

IP_ADDRESS
Optional Parameter

The binary IP address of the target.

PORT
Optional Parameter

The binary port number of the target.

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

STRING_IP_ADDRESS
Optional Parameter

The IP address of the target, expressed as a string.

STRING_PORT
Optional Parameter

The port number of the target, expressed as a string.

TIMEOUT_VALUE
Optional Parameter

The interval after which a request will time out.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED

INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, SEND_SSL_DATA function

The SEND_SSL_DATA function is called to send data to a connected TCP/IP client if the connection is secured using SSL.

Input Parameters

SEND_BUFFER
The buffer of data to be sent.

SOCKET_ADDR
The address of the socket.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR

MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:
TASK_CANCELLED
TIMED_OUT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, SET_SOCKET_OPTS function

Set the attributes of a socket.

Input Parameters

SWITCH_TCB
Optional parameter

Specifies if a switch to a socket TCB is required.

Values of the parameter are:
NO
YES

SOCKET_TOKEN
A token that is generated when a socket is created, and is used subsequently to identify the socket.

SO_LINGER
Optional Parameter

A sockets parameter that controls socket shutdown behavior, allowing the socket to shut down gracefully.

SO_REUSE_IP_ADDRESS
Optional Parameter

A binary parameter that specifies whether the socket can reuse an IP address.

Values for the parameter are:
NO
YES

SSL
Optional Parameter

A binary parameter that specifies whether the socket supports the secure sockets layer (SSL).

Values for the parameter are:
NO

YES

TCP_NODELAY

Optional Parameter

A binary parameter that specifies whether to send small messages on the socket without buffering them first.

Values for the parameter are:

NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE

The following values are returned when RESPONSE is EXCEPTION:

ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is PURGED:

TASK_CANCELLED
TIMED_OUT

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, SURRENDER function

This function requests the owner of a dormant session table entry (STE) to surrender control of it so that its resources can be used by another transaction. A dormant STE is one that is between transactions: it is waiting for another client interaction in a persistent connection.

Input Parameters

STE_PTR

The address of the session table entry (STE).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_FAILURE
LOOP
SOCKET_IN_USE
```

The following values are returned when RESPONSE is EXCEPTION:

```
ADDRESS_IN_USE
ADDRESS_NOT_AVAILABLE
ALREADY_ASSOCIATED
CLIENT_ERROR
CONNECTION_CLOSED
CONNECTION_REFUSED
INSUFFICIENT_STORAGE
INSUFFICIENT_THREADS
INVALID_OPTION
IO_ERROR
MISSING_OPTION
NEVER_ASSOCIATED
NO_CONNECTION
NO_SOCKET_AVAILABLE
NOT_AUTHORIZED
NOT_PENDING
NOTIFICATION_UNAVAILABLE
NOTIFIED
SCHEDULED
SSL_HANDSHAKE_ERROR
STATE_ERROR
TCP_NOT_ACTIVE
UNKNOWN_SESSION_TOKEN
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is PURGED:

```
TASK_CANCELLED
TIMED_OUT
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCK gate, SWITCH_TO_SOCKET_TCB function

The SWITCH_TO_SOCKET_TCB function switches to the required socket TCB.

Input Parameters

SOCKET_TOKEN

Optional Parameter

A token that is generated when a socket is created, and is used subsequently to identify the socket.

Output Parameters

OLD_TCB_TOKEN

Optional Parameter

The token that represents the TCB the caller of the SWITCH_TO_SOCKET_TCB function was executing on.

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
CHANGE_MODE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
ALREADY_ON_SOCKET_TCB
INCORRECT_TCB_SWITCH_SEQ
IO_ERROR
UNKNOWN_SESSION_TOKEN
```

The following values are returned when RESPONSE is PURGED:

```
TASK_CANCELLED
```

SOCK gate, SWITCH_FROM_SOCKET_TCB function

The SWITCH_FROM_SOCKET_TCB function switches from the socket TCB to the TCB that the SWITCH_TO_SOCKET_TCB was made from.

Input Parameters

SOCKET_TOKEN

Optional Parameter

A token that is generated when a socket is created, and is used subsequently to identify the socket.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
CHANGE_MODE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
ALREADY_ON_SOCKET_TCB
INCORRECT_TCB_SWITCH_SEQ
NOT_ON_SOCKET_TCB
UNKNOWN_SESSION_TOKEN
```

The following values are returned when RESPONSE is PURGED:

```
TASK_CANCELLED
```

SOCK gate, SWITCH_FOR_ASYNC_IO function

The SWITCH_FOR_ASYNC_IO function switches from the required socket TCB so that an ASYNC I/O can be issued.

Input Parameters

SOCKET_TOKEN

Optional Parameter

A token that is generated when a socket is created, and is used subsequently to identify the socket.

ON_SOCKET_TCB

A binary parameter that specifies whether the caller is already executing on a socket TCB

Values of the parameter are:

YES
NO

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP
CHANGE_MODE_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ALREADY_ON_SOCKET_TCB
INCORRECT_TCB_SWITCH_SEQ
IO_ERROR
UNKNOWN_SESSION_TOKEN

The following values are returned when RESPONSE is PURGED:

TASK_CANCELLED

SOIS gate, DELETE_CERTIFICATE_DATA function

The DELETE_CERTIFICATE_DATA deletes certificate data from the sockets repository.

Input Parameters

REPOSITORY_TOKEN

a token representing a certificate exported to the repository.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
CEEPIPI_ERROR
LISTENER_ATTACH_FAILURE
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:

AT_MAXSOCKETS
HOSTNAME_TRUNCATED
IIOPLISTENER_NO
IO_ERROR
MAXSOCKETS_HARD_LIMIT
REPOSITORY_ERROR
TCPIP_ALREADY_CLOSED
TCPIP_ALREADY_OPEN
TCPIP_UNAVAILABLE
UNKNOWN_CLIENT_ADDRESS
UNKNOWN_CLIENT_HOSTNAME

UNKNOWN_LISTEN_TOKEN
UNKNOWN_SERVER_ADDRESS
UNKNOWN_SERVER_HOSTNAME
UNKNOWN_SESSION_TOKEN
UNKNOWN_SOCKET_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOIS gate, EXPORT_CERTIFICATE_DATA function

The EXPORT_CERTIFICATE_DATA function saves a certificate in the sockets repository.

Input Parameters

CERTIFICATE_INFORMATION

is a block representing the certificate.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
CEEPIPI_ERROR
LISTENER_ATTACH_FAILURE
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:

AT_MAXSOCKETS
HOSTNAME_TRUNCATED
IIOPLISTENER_NO
IO_ERROR
MAXSOCKETS_HARD_LIMIT
REPOSITORY_ERROR
TCPIP_ALREADY_CLOSED
TCPIP_ALREADY_OPEN
TCPIP_UNAVAILABLE
UNKNOWN_CLIENT_ADDRESS
UNKNOWN_CLIENT_HOSTNAME
UNKNOWN_LISTEN_TOKEN
UNKNOWN_SERVER_ADDRESS
UNKNOWN_SERVER_HOSTNAME
UNKNOWN_SESSION_TOKEN
UNKNOWN_SOCKET_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REPOSITORY_TOKEN

Optional Parameter

is a token that represents the saved certificate data.

SOIS gate, IMPORT_CERTIFICATE_DATA function

The IMPORT_CERTIFICATE_DATA imports certificate data from the sockets repository.

Input Parameters

CERTIFICATE_INFORMATION

is a block representing the certificate.

REPOSITORY_TOKEN

Optional Parameter

a token representing a certificate exported to the repository.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
CEEPIPI_ERROR
LISTENER_ATTACH_FAILURE
LOCK_FAILURE
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
AT_MAXSOCKETS
HOSTNAME_TRUNCATED
IIOPLISTENER_NO
IO_ERROR
MAXSOCKETS_HARD_LIMIT
REPOSITORY_ERROR
TCPIP_ALREADY_CLOSED
TCPIP_ALREADY_OPEN
TCPIP_UNAVAILABLE
UNKNOWN_CLIENT_ADDRESS
UNKNOWN_CLIENT_HOSTNAME
UNKNOWN_LISTEN_TOKEN
UNKNOWN_SERVER_ADDRESS
UNKNOWN_SERVER_HOSTNAME
UNKNOWN_SESSION_TOKEN
UNKNOWN_SOCKET_TOKEN
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CERTIFICATE_USERID

Optional Parameter

is the userid associated with the certificate.

SOIS gate, INITIALIZE_ENVIRONMENT function

The INITIALIZE_ENVIRONMENT function is called during SO domain startup to create and initialize the CEEPIPI Language Environment pre-initialized environment for invokcation of C functions.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
CEEPIPI_ERROR
LISTENER_ATTACH_FAILURE
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:

AT_MAXSOCKETS
HOSTNAME_TRUNCATED
IIOPLISTENER_NO
IO_ERROR
MAXSOCKETS_HARD_LIMIT
REPOSITORY_ERROR
TCPIP_ALREADY_CLOSED
TCPIP_ALREADY_OPEN
TCPIP_UNAVAILABLE
UNKNOWN_CLIENT_ADDRESS
UNKNOWN_CLIENT_HOSTNAME
UNKNOWN_LISTEN_TOKEN
UNKNOWN_SERVER_ADDRESS
UNKNOWN_SERVER_HOSTNAME
UNKNOWN_SESSION_TOKEN
UNKNOWN_SOCKET_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOIS gate, INQUIRE function

The INQUIRE function is called by tasks that have been attached by the listener in response to a new TCP/IP connection. It provides TCP/IP and socket information about the connection and the connected client.

Input Parameters

CLIENT_CERTIFICATE

Optional Parameter

A buffer in which the X.509 certificate presented by the client is returned to the caller.

CLIENT_HOSTNAME

Optional Parameter

A buffer in which the full host name of the client is returned to the caller.

GENERIC_HOSTNAME

Optional Parameter

A buffer in which the full generic host name of the CICS region, as known to the DNS in a connection optimization environment, is returned to the caller.

KERBEROS_PRINCIPAL

Optional Parameter

The principal for Kerberos authentication.

LISTEN_TOKEN

Optional Parameter

A token representing the opened TCP/IP service.

LOCKHELD
Optional Parameter

Specifies whether the caller already holds the lock for searching the LTE chain.

Values for the parameter are:
NO
YES

REALM
Optional Parameter

The realm name under which the CICS region is running.

SERVER_HOSTNAME
Optional Parameter

A buffer in which the full host name of the CICS region is returned to the caller.

SOCKET_ADDR
Optional Parameter

The address of the socket.

SOCKET_TOKEN
Optional Parameter

A token that represents the socket.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
CEEPIPI_ERROR
LISTENER_ATTACH_FAILURE
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:
AT_MAXSOCKETS
HOSTNAME_TRUNCATED
IIOPLISTENER_NO
IO_ERROR
MAXSOCKETS_HARD_LIMIT
REPOSITORY_ERROR
TCPIP_ALREADY_CLOSED
TCPIP_ALREADY_OPEN
TCPIP_UNAVAILABLE
UNKNOWN_CLIENT_ADDRESS
UNKNOWN_CLIENT_HOSTNAME
UNKNOWN_LISTEN_TOKEN
UNKNOWN_SERVER_ADDRESS
UNKNOWN_SERVER_HOSTNAME
UNKNOWN_SESSION_TOKEN
UNKNOWN_SOCKET_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACTSOCKETS
Optional Parameter

The number of sockets that are currently active.

ATTACHSEC
Optional Parameter

The level of attach-time user security specified in the TCPIPSERVICE definition.

Values for the parameter are:
LOCAL
VERIFY

AUTHENTICATION
Optional Parameter

The authentication and identification scheme that is used for the connection.

Values for the parameter are:
ASSERTED
AUTOMATIC
AUTOREGISTER
BASIC
CERTIFICATE
KERBEROS
NONE

AUTHENTICATION
Optional Parameter

The authentication and identification scheme to be used for inbound TCP/IP connections.

Values for the parameter are:
AUTOMATIC
AUTOREGISTER
BASIC
CERTIFICATE
KERBEROS
NONE

CERTIFICATE_STATUS
Optional Parameter

The status of the X.509 certificate associated with the connection.

Values for the parameter are:
NONE
REGISTERED
UNREGISTERED
UNTRUSTED
UNKNOWN

CERTIFICATE_USERID
Optional Parameter

The user ID associated with the certificate.

CIPHER_SELECTED
Optional Parameter

The code for the cipher suite that was selected for use during the SSL connection.

CLIENT_BIN_IP_ADDRESS
Optional Parameter

The 32-bit binary IP address of the client.

CLIENT_IP_ADDRESS
Optional Parameter

The text representation of the IP address of the client.

CLIENT_IP_ADDRESS_LEN
Optional Parameter

The length of the text representation of the client IP address.

CLIENT_IPFAMILY
Optional Parameter

The format of the client IP address.

CONNECTION_SECURE
Optional Parameter

Whether the connection is SSL secured.

Values for the parameter are:
- YES
- NO

CONNECTIONS
Optional Parameter

Either the number of connections for the service represented by the supplied TCPIPSERVICE name, or the total number of TCP/IP connections to all of the currently active services.

DNS_STATUS
Optional Parameter

The Domain Name System (DNS) registration status of the service.

Values for the parameter are:
- DEREGERROR
- DEREGISTERED
- NOTAPPLIC
- REGERROR
- REGISTERED
- UNAVAILABLE
- UNREGISTERED

FORCE_CLOSE
Optional Parameter

Specifies whether the socket must be closed after the client has received a response (YES), or whether the socket remains open to allow the client to maintain a persistent connection (NO).

Values for the parameter are:
- YES
- NO

GROUP_NAME
Optional Parameter

The name of the dynamic DNS group that is registered with the MVS Work Load Manager for this service.

LISTENER_PORT
Optional Parameter

The port number that the connection was received on.

LISTENER_STATUS
Optional Parameter

The current status of the SO domain listener task.

Values for the parameter are:
CLOSED
CLOSING
IMMCLOSE
IMMCLOSING
OPEN
OPENING

MAXDATA_LENGTH
Optional Parameter

The maximum length of data that CICS will receive when operating as an HTTP server.

MAXSOCKETS
Optional Parameter

The value of the **MAXSOCKETS** system initialization parameter.

PEER_BIN_IP_ADDRESS
Optional Parameter

The binary IP address of the peer client or server.

PRIVACY
Optional Parameter

The level of SSL encryption required for inbound connections to this service.

Values for the parameter are:
NOTSUPPORTED
REQUIRED
SUPPORTED

PROTOCOL
Optional Parameter

The application level protocol used on the TCP/IP port.

Values for the parameter are:
ECI
HTTP
USER

SERVER_BIN_IP_ADDRESS
Optional Parameter

The 32-bit binary IP address of the CICS region.

SERVER_IP_ADDRESS
Optional Parameter

The text representation of the IP address of the CICS region.

SERVER_IP_ADDRESS_LEN
Optional Parameter

The length of the text representation of the server IP address.

SERVER_IPFAMILY
Optional Parameter

The format of the server IP address.

SSLTYPE
Optional Parameter

Specifies whether or not SSL is being used to secure this connection.

Values for the parameter are:
CLIENTAUTH

NO
YES
ATTLSAWARE

TCPIP_STATUS
Optional Parameter

The status of TCP/IP in the CICS region.

Values for the parameter are:
CLOSED
CLOSING
IMMCLOSE
IMMCLOSING
OPEN
OPENING

TCPIPSERVICE_NAME
Optional Parameter

The name of the service that attached the task.

TRANSID
Optional Parameter

The transaction ID associated with the service.

TSQ_PREFIX
Optional Parameter

The temporary storage queue prefix specified in the TCPIPSERVICE resource for this connection.

URM_NAME
Optional Parameter

The name of the user-replaceable program specified in the TCPIPSERVICE resource for this connection.

USER_TOKEN
Optional Parameter

The user token associated with the connection.

VALIDATION_HASH
Optional Parameter

SOIS gate, INQUIRE_CONNECTION function

Return information about a TCP/IP connection.

Input Parameters

CLIENT_HOSTNAME
Optional Parameter

A buffer in which the full host name of the client is returned to the caller.

SERVER_HOSTNAME
Optional Parameter

A buffer in which the full host name of the CICS region is returned to the caller.

SOCKET_TOKEN
The token that represents the connection.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
CEEPIPI_ERROR

LISTENER_ATTACH_FAILURE
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:

AT_MAXSOCKETS
HOSTNAME_TRUNCATED
IIOPLISTENER_NO
IO_ERROR
MAXSOCKETS_HARD_LIMIT
REPOSITORY_ERROR
TCPIP_ALREADY_CLOSED
TCPIP_ALREADY_OPEN
TCPIP_UNAVAILABLE
UNKNOWN_CLIENT_ADDRESS
UNKNOWN_CLIENT_HOSTNAME
UNKNOWN_LISTEN_TOKEN
UNKNOWN_SERVER_ADDRESS
UNKNOWN_SERVER_HOSTNAME
UNKNOWN_SESSION_TOKEN
UNKNOWN_SOCKET_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APPLDATA

Optional Parameter

The application data associated with the request.

CIPHER_SELECTED

Optional Parameter

The code for the cipher suite that was selected for use during the SSL connection.

CLIENT_BIN_IP_ADDRESS

Optional Parameter

The binary IP address of the client.

CLIENT_BIN_PORT

Optional Parameter

The binary port number of the client.

CLIENT_IP_ADDRESS

Optional Parameter

The IP address of the client.

CLIENT_IP_ADDRESS_LEN

Optional Parameter

The length of the text representation of the client IP address.

CLIENT_IPFAMILY

Optional Parameter

The format of the client IP address.

CLIENT_PORT

Optional Parameter

The port number of the client.

CLIENTLOC
Optional Parameter

The cluster connection type.

MVSIMAGE
Optional Parameter

The MVS image associated with the request.

SERVAUTH
Optional Parameter

The RACF SERVAUTH profile associated with the request.

SERVER_BIN_IP_ADDRESS
Optional Parameter

The binary IP address of the server.

SERVER_BIN_PORT
Optional Parameter

The binary port number of the server.

SERVER_IP_ADDRESS
Optional Parameter

The IP address of the server.

SERVER_IP_ADDRESS_LEN
Optional Parameter

The length of the text representation of the server IP address.

SERVER_IPFAMILY
Optional Parameter

The format of the server IP address.

SERVER_PORT
Optional Parameter

The port number of the server.

TCPIP_JOBNAME
Optional Parameter

The name of the TCP/IP job associated with the request.

TCPIP_ZONENAME
Optional Parameter

The name of the TCP/IP network security zone associated with the request.

SOIS gate, INQUIRE_PARAMETERS function

Returns the current values of the parameters for the SO domain. The values might have changed from their initial values, as specified in the system initialization parameters.

Input Parameters

CIPHER_SUITES
Optional Parameter

A binary representation of the cipher suites that are used to encrypt data.

CRL_PROFILE
Optional Parameter

The current value of the **CRLPROFILE** system initialization parameter.

Output Parameters

REASON
The values for the parameter are as follows:

ABEND
INVALID_CIPHERS
INVALID_FORMAT
INVALID_FUNCTION
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONFDATA
Optional Parameter

The current value of the **CONFDATA** system initialization parameter.

Values for the parameter are as follows:
HIDETC
SHOW

ENCRYPTION
Optional Parameter

The current value of the **ENCRYPTION** system initialization parameter.

Values for the parameter are as follows:
ALL
TLS12
STRONG

KEYRING
Optional Parameter

The current value of the **KEYRING** system initialization parameter.

MAXSOCKETS
Optional Parameter

The current value of the **MAXSOCKETS** system initialization parameter.

MAXSSLTCBS
Optional Parameter

The current value of the **MAXSSLTCBS** system initialization parameter.

SESSION_CACHE
Optional Parameter

The current value of the **SSLCACHE** system initialization parameter.

Values for the parameter are as follows:
CICS
SYSPLEX

SSLDELAY
Optional Parameter

The current value of the **SSLDELAY** system initialization parameter.

TCPIP
Optional Parameter

The current value of the **TCPIP** system initialization parameter.

Values for the parameter are as follows:
NO
YES

SOIS gate, INQUIRE_SOCKET_TOKEN function

Return the socket token for the current task.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
CEEPIPI_ERROR
LISTENER_ATTACH_FAILURE
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:

AT_MAXSOCKETS
HOSTNAME_TRUNCATED
IIOPLISTENER_NO
IO_ERROR
MAXSOCKETS_HARD_LIMIT
REPOSITORY_ERROR
TCPIP_ALREADY_CLOSED
TCPIP_ALREADY_OPEN
TCPIP_UNAVAILABLE
UNKNOWN_CLIENT_ADDRESS
UNKNOWN_CLIENT_HOSTNAME
UNKNOWN_LISTEN_TOKEN
UNKNOWN_SERVER_ADDRESS
UNKNOWN_SERVER_HOSTNAME
UNKNOWN_SESSION_TOKEN
UNKNOWN_SOCKET_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCKET_TOKEN

The socket token for the current task.

SOIS gate, INQUIRE_STATISTICS function

The INQUIRE_STATISTICS function returns gathered statistics about an open tcpipservice.

Input Parameters

LISTEN_TOKEN

A token that represents the opened tcpipservice.

RESET

A value that indicates whether to reset the statistics.

Values for the parameter are as follows:

NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
CEEPIPI_ERROR
LISTENER_ATTACH_FAILURE
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:

AT_MAXSOCKETS
HOSTNAME_TRUNCATED
IIOPLISTENER_NO
IO_ERROR
MAXSOCKETS_HARD_LIMIT
REPOSITORY_ERROR
TCPIP_ALREADY_CLOSED
TCPIP_ALREADY_OPEN
TCPIP_UNAVAILABLE
UNKNOWN_CLIENT_ADDRESS
UNKNOWN_CLIENT_HOSTNAME
UNKNOWN_LISTEN_TOKEN
UNKNOWN_SERVER_ADDRESS
UNKNOWN_SERVER_HOSTNAME
UNKNOWN_SESSION_TOKEN
UNKNOWN_SOCKET_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ATTACH_COUNT

Optional Parameter

The total number of tasks that were attached to handle incoming connections.

PEAK_CONNECTIONS

Optional Parameter

The high-water mark for connections since the last reset.

RECV_BYTES

Optional Parameter

The number of bytes received from TCP/IP.

RECV_COUNT

Optional Parameter

The number of times that TCP/IP receive was called.

SEND_BYTES

Optional Parameter

The number of bytes that were sent to TCP/IP.

SEND_COUNT

Optional Parameter

The number of times that TCP/IP send was called.

SOIS gate, SET function

The SET function is called to open, close or immediately close the SO domain within a region. This is called in response to a SET TCPIP operator or SPI command.

Input Parameters

ATTACHSEC

Optional Parameter

The level of attach-time user security required for this connection

Values for the parameter are:

LOCAL
VERIFY

MAXSOCKETS
Optional Parameter

The maximum number of IP sockets that can be managed by the CICS sockets domain. Used with **TCPIP_STATUS(OPEN)**

TCPIP_STATUS
Optional Parameter

The intended status of the domain.

Values for the parameter are:
CLOSED
IMMCLOSE
OPEN

TRACE_SUPPRESSION
Optional Parameter

A binary value indicating whether trace is to be suppressed.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
CEEPIPI_ERROR
LISTENER_ATTACH_FAILURE
LOCK_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:
AT_MAXSOCKETS
HOSTNAME_TRUNCATED
IIOPLISTENER_NO
IO_ERROR
MAXSOCKETS_HARD_LIMIT
REPOSITORY_ERROR
TCPIP_ALREADY_CLOSED
TCPIP_ALREADY_OPEN
TCPIP_UNAVAILABLE
UNKNOWN_CLIENT_ADDRESS
UNKNOWN_CLIENT_HOSTNAME
UNKNOWN_LISTEN_TOKEN
UNKNOWN_SERVER_ADDRESS
UNKNOWN_SERVER_HOSTNAME
UNKNOWN_SESSION_TOKEN
UNKNOWN_SOCKET_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

NEWMAXSOCKETS
Optional Parameter

The actual value of MAXSOCKETS. If the userid under which the CICS job is running does not have superuser authority, CICS might set the MAXSOCKETS limit to a smaller value than requested.

SOIS gate, SET_PARAMETERS function

The SET_PARAMETERS function is called during CICS initialization when the SIT is processed. It sets the startup parameters for the SO domain.

Input Parameters

CONFDATA
Optional Parameter

The value of the **CONFDATA** system initialization parameter.

Values for the parameter are:
HIDETC
SHOW

CRL_PROFILE
Optional Parameter

The value of the **CRLPROFILE** system initialization parameter.

ENCRYPTION
Optional Parameter

The value of the **ENCRYPTION** system initialization parameter.

IIOPLISTENER
Optional Parameter

The value of the **IIOPLISTENER** system initialization parameter.

Values for the parameter are:
NO
YES

KEYRING
Optional Parameter

The value of the **KEYRING** system initialization parameter.

MAXSOCKETS
Optional Parameter

The value of the **MAXSOCKETS** system initialization parameter.

MAXSSLTCBS
Optional Parameter

The value of the **MAXSSLTCBS** system initialization parameter.

SESSION_CACHE
Optional Parameter

The value of the **SSLCACHE** system initialization parameter.

Values for the parameter are:
CICS
SYSPLEX

SSLDELAY
Optional Parameter

The value of the **SSLCACHE** system initialization parameter.

TCPIP
Optional Parameter

The value of the **TCPIP** system initialization parameter.

Values for the parameter are:
YES
NO

Output Parameters

REASON

The values for the parameter are:
ABEND
INVALID_CIPHERS
INVALID_FORMAT
INVALID_FUNCTION
LOOP

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOIS gate, VALIDATE_CIPHERS function

This function accepts a string of cipher suites and removes any that are not supported.

Input Parameters

CIPHER_SUITES

The list of cipher suites to be validated.

Output Parameters

REASON

The values for the parameter are:
ABEND
INVALID_CIPHERS
INVALID_FORMAT
INVALID_FUNCTION
LOOP

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOIS gate, VERIFY_IP_ADDRESS function

This function verifies the format and value of an IP address, returning if required its char(16) value. It will return UNKNOWN_SERVER_ADDRESS is the input is not a correct IPv4 or IPv6 address format.

Input Parameters

SWITCH_TCB

Optional parameter

Specifies if a switch to a socket TCB is required.

Values of the parameter are:
NO
YES

SERVER_HOSTNAME

Optional Parameter

The host name of the target IP address.

SERVER_IP_ADDRESS

Optional Parameter

The target IP address.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
CEEPIPI_ERROR
LISTENER_ATTACH_FAILURE
LOCK_FAILURE
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
AT_MAXSOCKETS
HOSTNAME_TRUNCATED
IIOPLISTENER_NO
IO_ERROR
MAXSOCKETS_HARD_LIMIT
REPOSITORY_ERROR
TCPIP_ALREADY_CLOSED
TCPIP_ALREADY_OPEN
TCPIP_UNAVAILABLE
UNKNOWN_CLIENT_ADDRESS
UNKNOWN_CLIENT_HOSTNAME
UNKNOWN_LISTEN_TOKEN
UNKNOWN_SERVER_ADDRESS
UNKNOWN_SERVER_HOSTNAME
UNKNOWN_SESSION_TOKEN
UNKNOWN_SOCKET_TOKEN
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SERVER_BIN_IP_ADDRESS

The binary form of the IP address.

SOLS gate, LISTEN function

This function listens for incoming connections. The ports to listen on are controlled by installing and opening TCPIPSERVICE definitions. The function is called from the system task CSOL that is attached by the socket domain at startup. It returns when TCP/IP is closed or CICS shuts down.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOCK_FAILURE
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
CONNECTION_CLOSED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOPL gate, CREATE_SOCKET_POOL function

The CREATE_SOCKET_POOL function creates a pool to hold opened client HTTP connections in a dormant state for connection pooling.

Input Parameters

TIMEOUT_VALUE
The time for which CICS keeps a client HTTP connection open after the CICS application has finished using it, as specified in the SOCKETCLOSE attribute of the URIMAP resource for the connection.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`NOT_FOUND`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCKET_POOL_TOKEN
The unique identifier for the pool of opened client HTTP connections.

SOPL gate, INQUIRE_SOCKET_POOL function

The INQUIRE_SOCKET_POOL function inquires on a pool of opened client HTTP connections that CICS is holding in a dormant state for connection pooling.

Input Parameters

SOCKET_POOL_TOKEN
The unique identifier for the pool of opened client HTTP connections.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`NOT_FOUND`

The following values are returned when RESPONSE is INVALID:
`INVALID_TOKEN`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCKET_COUNT
Optional Parameter

The number of client HTTP connections that are in the pool in a dormant state.

TIMEOUT_VALUE
Optional Parameter

The time for which CICS keeps a client HTTP connection open after the CICS application has finished using it, as specified in the SOCKETCLOSE attribute of the URIMAP resource for the connection.

SOPL gate, DELETE_SOCKET_POOL function

The DELETE_SOCKET_POOL function deletes a pool of opened client HTTP connections that CICS is holding in a dormant state for connection pooling.

Input Parameters

SOCKET_POOL_TOKEN
The unique identifier for the pool of opened client HTTP connections.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOPL gate, GET_SOCKET_FROM_POOL function

The GET_SOCKET_FROM_POOL function retrieves an opened client HTTP connection from a pool to supply to a task.

Input Parameters

SOCKET_POOL_TOKEN
The unique identifier for the pool of opened client HTTP connections.

Output Parameters

SWITCH_TCB
Optional parameter SWITCH_TCB added:

A binary parameter that specifies whether the socket domain should switch to the socket TCB.

Values of the parameter are:
YES
NO

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCKET_TOKEN
A unique identifier for an individual client HTTP connection.

SOPL gate, RETURN_SOCKET_TO_POOL function

The RETURN_SOCKET_TO_POOL function returns an opened client HTTP connection to a pool.

Input Parameters

SOCKET_POOL_TOKEN
The unique identifier for the pool of opened client HTTP connections.

SOCKET_TOKEN
A unique identifier for an individual client HTTP connection.

Output Parameters

SWITCH_TCB
Optional parameter SWITCH_TCB added:

A binary parameter that specifies whether the socket domain should switch to the socket TCB.

Values of the parameter are:
YES

NO

REASON

The following values are returned when RESPONSE is INVALID:

INVALID_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOPL gate, CLOSE_SOCKET_IN_POOL function

The CLOSE_SOCKET_IN_POOL function closes and removes an opened client HTTP connection that is being held in a dormant state in a pool.

Input Parameters

SOCKET_POOL_TOKEN

The unique identifier for the pool of opened client HTTP connections.

SOCKET_TOKEN

A unique identifier for an individual client HTTP connection.

Output Parameters

SWITCH_TCB

Optional parameter SWITCH_TCB added:

A binary parameter that specifies whether the socket domain should switch to the socket TCB.

Values of the parameter are:

YES

NO

REASON

The following values are returned when RESPONSE is INVALID:

INVALID_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOPL gate, RECLAIM_SOCKET function

The RECLAIM_SOCKET function is used when the CICS region is at its MAXSOCKETS limit, to free sockets that are being used for client HTTP connections held in a dormant state in a pool.

Input Parameters

None.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOPL gate, INQUIRE_STATISTICS function

The INQUIRE_STATISTICS function returns gathered statistics about a pool of opened client HTTP connections.

Input Parameters

SOCKET_POOL_TOKEN
The unique identifier for the pool of opened client HTTP connections.

RESET
A value indicating if the statistics should be reset.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOCKET_COUNT
Optional Parameter

The current number of open client HTTP connections held in the pool for reuse.

SOCKET_COUNT_PEAK
Optional Parameter

The peak number of open client HTTP connections held in the pool for reuse.

RECLAIMED_COUNT
Optional Parameter

The number of pooled connections that were closed in the pool by CICS to reclaim their sockets.

TIMED_OUT_COUNT
Optional Parameter

The number of pooled connections that were closed in the pool by CICS because they reached their timeout value without being reused.

SORD gate, DEREGISTER function

The DEREGISTER function is called to close a TCPIPSERVICE. The listener task closes the listening socket and no more connections to the port are permitted. Any tasks handling existing connections are allowed to end normally.

Input Parameters

LISTEN_TOKEN
is a token representing the opened tcpipservice.

DNSGROUPNAME
Optional Parameter

The group name with which CICS registers to Workload Manager, for connection optimization.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP

UNKNOWN_POST_CODE

The following values are returned when RESPONSE is EXCEPTION:

AT_MAXSOCKETS
INSUFFICIENT_STORAGE
NOT_PERMITTED_TO_BIND
PORT_IN_USE
TCPIP_CLOSED
TCPIP_INACTIVE
TCPIP_SERVICE_ERROR
UNKNOWN_ADDRESS
UNKNOWN_LISTEN_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SORD gate, IMMCLOSE function

The IMMCLOSE function is called to immediately close a TCP/IP service. The listener task closes the listening socket and no more connections to the port are permitted. All existing connections are closed and any tasks handling them are abnormally ended.

Input Parameters

LISTEN_TOKEN

is a token representing the opened TCP/IP service.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOCK_FAILURE
LOOP
UNKNOWN_POST_CODE

The following values are returned when RESPONSE is EXCEPTION:

AT_MAXSOCKETS
INSUFFICIENT_STORAGE
NOT_PERMITTED_TO_BIND
PORT_IN_USE
TCPIP_CLOSED
TCPIP_INACTIVE
TCPIP_SERVICE_ERROR
UNKNOWN_ADDRESS
UNKNOWN_LISTEN_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SORD gate, REGISTER function

The REGISTER function is called to open a TCP/IP service. It registers all the parameters of the service with the listener task.

Input Parameters

AUTHENTICATION
Optional Parameter

The authentication and identification scheme to be used for inbound TCP/IP connections.

Values for the parameter are:
```
AUTOMATIC
AUTOREGISTER
BASIC
CERTIFICATE
KERBEROS
NONE
```

BACKLOG
Optional Parameter

The value of the backlog parameter passed to the TCP/IP listen function for this service. It specifies how many connection requests TCP/IP will queue for this service.

IPADDRESS
The specific IP address that the listener will bind to for this service.

PORT_NUMBER
The TCP/IP port number on which to listen for a new connection.

RECV_TIMEOUT
Specifies whether or not receives should timeout, and if so, after how long.

SERVICE_NAME
The name of the TCP/IP service.

SSL
Specifies whether or not connections to this service are to be secured using the Secure Sockets Layer protocols.

Values for the parameter are:
```
CLIENTAUTH
NO
YES
ATTLSAWARE
```

TRANID
The transaction ID that is to be attached when a new connection is made to the listening port.

URM
The name of a user-replacable program that the handler transaction for this service will invoke during request processing.

ATTACHSEC
Optional Parameter

The level of attach-time user security specified in the TCPIPSERVICE definition.

CERTIFICATE_LABEL
Optional Parameter

The name of a certificate within the keyfile that this service will use to authenticate itself to clients with, if the SSL protocol is used.

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the CIPHER_SUITES parameter.

CIPHER_SUITES
Optional Parameter

A binary representation of the cipher suites that are used to encrypt data.

DNSGROUPNAME
Optional Parameter

The group name with which CICS registers to Workload Manager, for connection optimization.

DNSGRPCRITICAL
Optional Parameter

A binary value indicating whether the service is a critical member of the DNS group. When a critical service closes or fails, CICS deregisters the group name from Workload Manager.

Values for the parameter are:
- CRITICAL
- NONCRITICAL

MAXDATA_LENGTH
Optional Parameter

The maximum length of data that CICS will receive when operating as an HTTP server.

MAXPERSIST
Optional Parameter

The maximum number of persistent connections from web clients that the CICS region allows for this port at any one time.

PRIVACY
Optional Parameter

The level of SSL encryption required for inbound connections to this service.

Values for the parameter are:
- NOTSUPPORTED
- REQUIRED
- SUPPORTED

PROTOCOL
Optional Parameter

The application level protocol used on the TCP/IP port.

Values for the parameter are:
- ECI
- HTTP
- USER

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
- ABEND
- LOCK_FAILURE
- LOOP
- UNKNOWN_POST_CODE

The following values are returned when RESPONSE is EXCEPTION:
- AT_MAXSOCKETS
- INSUFFICIENT_STORAGE
- NOT_PERMITTED_TO_BIND

PORT_IN_USE
TCPIP_CLOSED
TCPIP_INACTIVE
TCPIP_SERVICE_ERROR
UNKNOWN_ADDRESS
UNKNOWN_LISTEN_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

LISTEN_TOKEN
A token representing the opened TCP/IP service. This is subsequently used to close the service.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SORD gate, REGISTER_NOTIFICATION function

This function is called by a client domain of the SO domain. After the registration call returns, the client domains SOCB notify gate may be driven asynchronously at any time a new TCP/IP connection arrives for a TCPIPSERVICE which has the PROTOCOL parameter set to the same as that registered by this call.

Input Parameters

CALLBACK_GATE
The gate at which the client domain is called back

PROTOCOL
The protocol for which the client domain wants to be called back.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOCK_FAILURE
LOOP
UNKNOWN_POST_CODE

The following values are returned when RESPONSE is EXCEPTION:
AT_MAXSOCKETS
INSUFFICIENT_STORAGE
NOT_PERMITTED_TO_BIND
PORT_IN_USE
TCPIP_CLOSED
TCPIP_INACTIVE
TCPIP_SERVICE_ERROR
UNKNOWN_ADDRESS
UNKNOWN_LISTEN_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SORL gate, UPDATE_REVOCATION_LIST function

Update a certificate revocation list (CRL) in the LDAP server that is specified in the **CRLPROFILE** system initialization parameter.

Input Parameters

REVOCATION_LIST
The new certificate revocation list

LDAP_ADMIN_DN
Optional Parameter

The LDAP administrator distinguished name

LDAP_ADMIN_PW
Optional Parameter

The LDAP administrator password

Output Parameters

REASON
The values for the parameter are:

```
ABEND
INVALID_CRL
INVALID_FORMAT
INVALID_FUNCTION
LDAP_ERROR
LOOP
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOTB gate, END_BROWSE function

The END_BROWSE function is called by CEMT and the SPI to end browsing tcpipservices.

Input Parameters

BROWSE_TOKEN
is a token representing the browse.

Output Parameters

REASON
The values for the parameter are:

```
INVALID_BROWSE_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOTB gate, GET_NEXT function

The GET_NEXT function is called by CEMT and the SPI for browsing TCPIPSERVICE resources. It returns information about an installed TCPIPSERVICE resources.

Input Parameters

BROWSE_TOKEN
A token representing the browse.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
INVALID_BROWSE_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TCPIPSERVICE_NAME
The name of the service that attached the task.

ATTACHSEC
Optional Parameter

The level of attach-time user security specified in the TCPIPSERVICE definition.

AUTHENTICATION
Optional Parameter

The authentication and identification scheme to be used for inbound TCP/IP connections.

Values for the parameter are:
AUTOMATIC
AUTOREGISTER
BASIC
CERTIFICATE
KERBEROS
NONE

BACKLOG
Optional Parameter

The backlog value associated with the service.

CERTIFICATE_LABEL
Optional Parameter

The certificate label associated with the service.

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the CIPHER_SUITES parameter.

CIPHER_SUITES
Optional Parameter

A binary representation of the cipher suites that are used to encrypt data.

CONNECTIONS
Optional Parameter

Either the number of connections for the service represented by the supplied TCPIPSERVICE name, or the total number of TCP/IP connections to all of the currently active services.

DNSGROUP
Optional Parameter

The group name with which CICS registers to Workload Manager, for connection optimization.

DNSSTATUS
Optional Parameter

The Domain Name System (DNS) registration status of the service.

Values for the parameter are:
DEREGERROR
DEREGISTERED
NOTAPPLIC
REGERROR
REGISTERED

UNAVAILABLE
UNREGISTERED

GRPCRITICAL
Optional Parameter

A binary value indicating whether the TCPIPSERVICE resource is a critical member of the DNS group. When a critical TCPIPSERVICE resource closes or fails, CICS deregisters the group name from Workload Manager.

Values for the parameter are:
CRITICAL
NONCRITICAL

IPADDRESS
Optional Parameter

The IP address that the service is bound to.

MAXDATA_LENGTH
Optional Parameter

The maximum length of data that CICS will receive when operating as an HTTP server.

MAXPERSIST
Optional Parameter

The maximum number of persistent connections from web clients that the CICS region allows for this port at any one time.

PORT
Optional Parameter

The port number associated with the service.

PRIVACY
Optional Parameter

The level of SSL encryption required for inbound connections to this service.

Values for the parameter are:
NOTSUPPORTED
REQUIRED
SUPPORTED

PROTOCOL
Optional Parameter

The application level protocol used on the TCP/IP port.

Values for the parameter are:
ECI
HTTP
USER

SOCKETCLOSE
Optional Parameter

The interval from the time of the initial receive request for incoming data, after which CICS is to time out the socket.

SSL
Optional Parameter

The SSL setting for the service.

Values for the parameter are:
CLIAUTH
NO
YES

ATTLSAWARE

STATUS
Optional Parameter

The current status of the service.

Values for the parameter are:
CLOSED
CLOSING
IMMCLOSING
OPEN
OPENING

TRANSID
Optional Parameter

The transaction ID associated with the service.

URM
Optional Parameter

The name of the user-replaceable program associated with the service.

SOTB gate, INQUIRE_TCPIPSERVICE function

The INQUIRE_TCPIPSERVICE function is called by CEMT and the SPI to inquire on a TCPIPSERVICE resource. It returns information about an installed TCPIPSERVICE resource.

Input Parameters

TCPIPSERVICE_NAME
The name of the TCP/IP service.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ATTACHSEC
Optional Parameter

The level of attach-time user security specified in the TCPIPSERVICE definition.

AUTHENTICATION
Optional Parameter

The authentication and identification scheme to be used for inbound TCP/IP connections.

Values for the parameter are:
AUTOMATIC
AUTOREGISTER
BASIC
CERTIFICATE
KERBEROS
NONE

BACKLOG
Optional Parameter

The value of the backlog parameter passed to the TCP/IP listen function for this service. It specifies how many connection requests TCP/IP will queue for this service.

CERTIFICATE_LABEL
Optional Parameter

The certificate label associated with the service.

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the CIPHER_SUITES parameter.

CIPHER_SUITES
Optional Parameter

A binary representation of the cipher suites that are used to encrypt data.

CONNECTIONS
Optional Parameter

Either the number of connections for the service represented by the supplied TCPIPSERVICE name, or the total number of TCP/IP connections to all of the currently active services.

DNSGROUP
Optional Parameter

The group name with which CICS registers to Workload Manager, for connection optimization.

DNSSTATUS
Optional Parameter

The Domain Name System (DNS) registration status of the service.

Values for the parameter are:
```
DEREGERROR
DEREGISTERED
NOTAPPLIC
REGERROR
REGISTERED
UNAVAILABLE
UNREGISTERED
```

GRPCRITICAL
Optional Parameter

A binary value indicating whether the TCPIPSERVICE resource is a critical member of the DNS group. When a critical TCPIPSERVICE resource closes or fails, CICS deregisters the group name from Workload Manager.

Values for the parameter are:
```
CRITICAL
NONCRITICAL
```

IPADDRESS
Optional Parameter

The IP address that the service is bound to.

MAXDATA_LENGTH
Optional Parameter

The maximum length of data that CICS will receive when operating as an HTTP server.

MAXPERSIST
Optional Parameter

The maximum number of persistent connections from web clients that the CICS region allows for this port at any one time.

PORT

Optional Parameter

The port number associated with the service.

PRIVACY

Optional Parameter

The level of SSL encryption required for inbound connections to this service.

Values for the parameter are:

```
NOTSUPPORTED
REQUIRED
SUPPORTED
```

PROTOCOL

Optional Parameter

The application level protocol used on the TCP/IP port.

Values for the parameter are:

```
ECI
HTTP
USER
```

SOCKETCLOSE

Optional Parameter

The interval from the time of the initial receive request for incoming data, after which CICS is to time out the socket.

SSL

Optional Parameter

The SSL setting for the service.

Values for the parameter are:

```
CLIAUTH
NO
YES
ATTLSAWARE
```

STATUS

Optional Parameter

The current status of the service.

Values for the parameter are:

```
CLOSED
CLOSING
IMMCLOSING
OPEN
OPENING
```

TRANSID

Optional Parameter

The transaction ID associated with the service.

URM

Optional Parameter

The name of the user-replaceable program associated with the service.

VALIDATION_HASH

Optional Parameter

A hash based on the contents of the listener table entry for the socket.

SOTB gate, SET_TCPIPSERVICE function

The SET_TCPIPSERVICE function is called by CEMT and the SPI to set tcpipservice parameters.

Input Parameters

TCPIPSERVICE_NAME
is the name of the tcpipservice.

BACKLOG
Optional Parameter

is the value of the backlog parameter passed to the TCP/IP listen function for this service. It specifies how many connection requests TCP/IP will queue for this service.

DNSSTATUS
Optional Parameter

The state of the Workload Manager's Domain Name System (DNS) registration of this TCPIPSERVICE.

Values for the parameter are:
DEREGISTERED

MAXDATA_LENGTH
Optional Parameter

The maximum length of data that CICS will receive when operating as an HTTP server.

STATUS
Optional Parameter

is either OPEN or CLOSED.

Values for the parameter are:
CLOSED
IMMCLOSED
OPEN

URM
Optional Parameter

is the name of a user-replacable program that the handler transaction for this service will invoke during request processing.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
IIOPLISTENER_NO
INVALID_STATUS
NOT_FOUND
PORT_IN_USE
PORT_NOT_AUTHORISED
TCPIP_CLOSED
TCPIP_INACTIVE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SOTB gate, START_BROWSE function

The START_BROWSE function is called by CEMT and the SPI for an browsing tcpipservices.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
AT_MAXSOCKETS
BROWSE_END
IIOPLISTENER_NO
INVALID_BROWSE_TOKEN
INVALID_STATUS
NOT_FOUND
PORT_IN_USE
PORT_NOT_AUTHORISED
TCPIP_CLOSED
TCPIP_INACTIVE
UNKNOWN_IP_ADDRESS
URM_NOT_POSSIBLE
```

BROWSE_TOKEN

is a token representing the browse.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Sockets domain's generic gates

Table 49 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 49. Sockets domain's generic gates

Gate	Trace	Functions	Format
SODM	SO 0101 SO 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	SO 0A01 SO 0A02	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
SOXM	SO 0901 SO 0902	INQUIRE_DATA_LENGTH GET_DATA DESTROY_TOKEN	XMXM
SORM	SO 1300 SO 1301	CREATE DISCARD INQUIRE INQUIRE_BY_NAME SET DISCARD	RLCB

Modules

Module	Function
DFHSOAD	Handles the following requests: ADD_REPLACE_TCPIPSERVICE DELETE_TCPIPSERVICE

Module	Function
DFHSOCK	Handles the following requests: LISTEN SEND RECEIVE CLOSE SEND_SSL_DATA RECV_SSL_DATA
DFHSODM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHSODUF	Formats the SO domain control blocks
DFHSOIS	Handles the following requests: INITIALIZE_ENVIRONMENT INQUIRE SET_PARAMETERS INQUIRE_STATISTICS VERIFY EXPORT_CERTIFICATE_DATA IMPORT_CERTIFICATE_DATA DELETE_CERTIFICATE_DATA INET_PTON INET_NTOP
DFHSONT	Handles the following requests: MXT_NOTIFY MXT_CHANGE_NOTIFY
DFHSOPL	Handles the following requests: CREATE_SOCKET_POOL INQUIRE_SOCKET_POOL DELETE_SOCKET_POOL GET_SOCKET_FROM_POOL RETURN_SOCKET_TO_POOL CLOSE_SOCKET_IN_POOL RECLAIM_SOCKET INQUIRE_STATISTICS
DFHSORD	Handles the following requests: REGISTER DEREGISTER IMMCLOSE
DFHSOSE	Handles the following requests: INITIALIZE_SSL SECURE_SOC_INIT SECURE_SOC_READ SECURE_SOC_WRITE SECURE_SOC_CLOSE SECURE_SOC_RESET TERMINATE_SSL EXPORT_CERTIFICATE_DATA IMPORT_CERTIFICATE_DATA DELETE_CERTIFICATE_DATA
DFHSOSM	Handles the following requests: STORAGE_NOTIFY

Module	Function
DFHSOTB	Handles the following requests: INQUIRE_TCPIPSERVICE START_BROWSE GET_NEXT END_BROWSE SET_TCPIPSERVICE
DFHSOTRI	Interprets SO domain trace entries

Statistics domain (ST)

The statistics domain controls the collection of resource statistics for a CICS system (the monitoring domain collects task statistics). The statistics domain collects data at user-specified intervals, at system quiesce or logical end of day, and when requested by the user, and writes it to the statistics data sets in SMF format. This can subsequently be used by the statistics offline utility to produce formatted reports.

Statistics domain's specific gates

The specific gates provide access for other domains to functions that are provided by the ST domain.

STST gate, COLLECT_RESOURCE_STATS function

The COLLECT_RESOURCE_STATS function of the STST format is used by the EXEC API to ask a domain to collect its monitoring data collection information.

Input Parameters

AC_APPLICATION_NAME
The address and length of an application name. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

AC_MAJOR_VERSION
The major version number of an application. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

AC_MINOR_VERSION
The minor version number of an application. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

AC_MICRO_VERSION
The micro version number of an application. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

LONG_RESOURCE_ID_DATA
Optional Parameter

The address and length of the resource identifier.

AC_PLATFORM_NAME
The address and length of a platform name. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

RESOURCE_STATISTICS_DATA
The address and length of the area into which the requested statistics are to be placed.

RESID_TOKEN
Optional Parameter

A token representing the resource ID required.

RESOURCE_ID
Optional Parameter

The address and length of the resource identifier.

RESOURCE_ID_2
Optional Parameter

The address and length of the resource identifier.

RESOURCE_ID_3
Optional Parameter

The address and length of the resource identifier.

RESOURCE_TYPE
Optional Parameter

The type of resource on which statistics are required.

SHOW_RECORDS
Whether private or private and public records are to be searched. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
ID_NOT_FOUND
NOT_AVAILABLE
TYPE_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LAST_RESET_TIME
Optional Parameter

indicates the time at which the statistics fields were last reset.

STST gate, COLLECT_STATISTICS function

The COLLECT_STATISTICS function of the STST format is used by the statistics domain to ask a domain to collect its statistics.

Input Parameters

DATA
indicates whether the domain being called is requested to return its statistics to the caller.

Values for the parameter are:

```
NO
YES
```

END_OF_DAY
indicates whether all statistics fields are to be reset.

Values for the parameter are:

```
NO
```

YES

RESET
indicates whether certain statistics fields are to be reset.

Values for the parameter are:
NO
YES

RESET_TIME
is the time of day to be used as the time at which the statistics fields were last reset.

RESOURCE_TYPE
Optional Parameter

indicates the resource in the AP domain on which statistics are to be collected.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
INCOMPLETE_DATA
NOT_AVAILABLE
TYPE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STST gate, DISABLE_STATISTICS function

The DISABLE_STATISTICS function of the STST gate is used to disable statistics interval collections.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STST gate, INQ_STATISTICS_OPTIONS function

The INQ_STATISTICS_OPTIONS function of the STST gate is used to return information associated with the statistics domain options.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

COLLECT
indicates whether interval statistics are being collected (and their counts reset).

Values for the parameter are:
NO
YES

EOD_TIME_OF_DAY
is the time of day at which end-of-day statistics are collected.
INTERVAL
is the interval at which statistics are being collected if COLLECT is YES.
NEXT_COLLECTION_TIME
is the time of the next collection of statistics. If COLLECT is YES, it is the earlier of the next interval collection time and the logical end-of-day time; if COLLECT is NO, it is the logical end-of-day time.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STST gate, RECORD_STATISTICS function

The RECORD_STATISTICS function of the STST gate is used to record statistics.

Input Parameters

STATISTICS_DATA
specifies the address and length of data requested.
STATISTICS_TYPE
indicates the type of statistics collection, either a normal collection or unsolicited.

Values for the parameter are:
```
COLLECTION
USS
```

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_DATA_FORMAT
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STST gate, REQUEST_STATISTICS function

The REQUEST_STATISTICS function of the STST gate is used to request a collection of statistics.

Input Parameters

REQUEST_TOKEN
uniquely identifies the collection of statistics requested by the caller.
RESET
indicates whether certain statistics fields are to be reset.

Values for the parameter are:
```
NO
YES
```
DOMAIN_TOKEN
Optional Parameter

identifies the domain from which the statistics are to be collected.
RESOURCE_TYPE
Optional Parameter

indicates the resource in the AP domain on which statistics are to be collected.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
INCOMPLETE_DATA
NOT_AVAILABLE
TYPE_NOT_FOUND
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_RESET
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STST gate, SET_STATISTICS_OPTIONS function

The SET_STATISTICS_OPTIONS function of the STST gate is used to set statistics options.

Input Parameters

COLLECT

Optional Parameter

indicates whether interval statistics are to be collected (and their counts reset).

Values for the parameter are:

```
NO
YES
```

COLLECT_UPDATE_ACTION

Optional Parameter

is the action to be taken when changing the COLLECT option value from NO to YES, or from YES to NO.

Values for the parameter are:

```
NOACTION
RECORD_RESETNOW
RECORDNOW
RESETNOW
```

EOD_TIME_OF_DAY

Optional Parameter

is the time of day at which end-of-day statistics are to be collected.

INTERVAL

Optional Parameter

is the interval at which statistics are to be collected if COLLECT is YES.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
COLL_ACTION_NO_UPDATE
```

The following values are returned when RESPONSE is INVALID:

```
INV_COLL_UPDATE_ACTION
```

INVALID_COLLECT
INVALID_EOD_TIME_OF_DAY
INVALID_INTERVAL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STST gate, STATISTICS_COLLECTION function

The STATISTICS_COLLECTION function of the STST gate is used to initiate a collection of statistics.

Input Parameters

COLLECTION_TYPE
indicates whether this is an interval collection or end-of-day collection of statistics.

Values for the parameter are:
EOD
INT

DATA
indicates whether the domain being called is requested to return its statistics to the caller.

Values for the parameter are:
NO
YES

END_OF_DAY
indicates whether all statistics fields are to be reset.

Values for the parameter are:
NO
YES

RESET
indicates whether certain statistics fields are to be reset.

Values for the parameter are:
NO
YES

SYSTEM_TERMINATING
Optional Parameter

indicates whether this is the last collection for the CICS run.
YES is used for the end-of-day collection that is taken when CICS is shut down.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Statistics domain's generic gates

Table 50 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 50. Statistics domain's generic gates

Gate	Trace	Functions	Format
DMDM	ST 0001 ST 0002	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
TISR	ST 0005 ST 0006	NOTIFY	TISR

In initialization processing, the statistics domain sets the initial statistics options:

- Collecting interval
- Logical end of day
- Collecting status.

For a cold start, the collecting interval defaults to 1 hour, the logical end of day defaults to midnight, and the collecting status defaults to ON. For any other type of start, the information comes from the global catalog.

In quiesce processing, the statistics domain collects and records statistics from all other domains.

In termination processing, the statistics domain collects and records end-of-day statistics.

Statistics domain's generic gates

Table 50 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 51. Statistics domain's generic gates

Gate	Trace	Functions	Format
DMDM	ST 0001 ST 0002	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
TISR	ST 0005 ST 0006	NOTIFY	TISR

In initialization processing, the statistics domain sets the initial statistics options:

- Collecting interval
- Logical end of day
- Collecting status.

For a cold start, the collecting interval defaults to 1 hour, the logical end of day defaults to midnight, and the collecting status defaults to ON. For any other type of start, the information comes from the global catalog.

In quiesce processing, the statistics domain collects and records statistics from all other domains.

In termination processing, the statistics domain collects and records end-of-day statistics.

Statistics domain's generic formats

Table 52 describes the generic formats owned by the domain and shows the functions performed on the calls.

Table 52. Statistics domain's generic formats

Format	Calling module	Function
STST	DFHEIQMS	COLLECT_RESOURCE_STATS
	DFHSTST	COLLECT_STATISTICS

Note: In the descriptions of the formats, the input parameters are input not to the statistics domain, but to the domain being called by the statistics domain. Similarly, the output parameters are output by the domain that was called by the statistics domain, in response to the call.

STST gate, COLLECT_RESOURCE_STATS function

The COLLECT_RESOURCE_STATS function of the STST format is used by the EXEC API to ask a domain to collect its monitoring data collection information.

Input Parameters

AC_APPLICATION_NAME
The address and length of an application name. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

AC_MAJOR_VERSION
The major version number of an application. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

AC_MINOR_VERSION
The minor version number of an application. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

AC_MICRO_VERSION
The micro version number of an application. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

LONG_RESOURCE_ID_DATA
Optional Parameter

The address and length of the resource identifier.

AC_PLATFORM_NAME
The address and length of a platform name. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

RESOURCE_STATISTICS_DATA
The address and length of the area into which the requested statistics are to be placed.

RESID_TOKEN
Optional Parameter

A token representing the resource ID required.

RESOURCE_ID
Optional Parameter

The address and length of the resource identifier.

RESOURCE_ID_2
Optional Parameter

The address and length of the resource identifier.

RESOURCE_ID_3
Optional Parameter

The address and length of the resource identifier.

RESOURCE_TYPE
Optional Parameter

The type of resource on which statistics are required.

SHOW_RECORDS
Whether private or private and public records are to be searched. This input parameter is ignored if the resource type specified by the **RESOURCE_TYPE** parameter is not supported as a private resource for applications deployed on platforms.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
ID_NOT_FOUND
NOT_AVAILABLE
TYPE_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LAST_RESET_TIME
Optional Parameter

indicates the time at which the statistics fields were last reset.

STST gate, COLLECT_STATISTICS function

The COLLECT_STATISTICS function of the STST format is used by the statistics domain to ask a domain to collect its statistics.

Input Parameters

DATA
indicates whether the domain being called is requested to return its statistics to the caller.

Values for the parameter are:

```
NO
YES
```

END_OF_DAY
indicates whether all statistics fields are to be reset.

Values for the parameter are:

```
NO
YES
```

RESET
indicates whether certain statistics fields are to be reset.

Values for the parameter are:

NO

YES

RESET_TIME

is the time of day to be used as the time at which the statistics fields were last reset.

RESOURCE_TYPE

Optional Parameter

indicates the resource in the AP domain on which statistics are to be collected.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND

LOOP

The following values are returned when RESPONSE is EXCEPTION:

INCOMPLETE_DATA

NOT_AVAILABLE

TYPE_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHSTDBX	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHSTDUF	Formats the ST domain control blocks in a CICS system dump
DFHSTST	Handles the following requests: INQ_STATISTICS_OPTIONS RECORD_STATISTICS REQUEST_STATISTICS SET_STATISTICS_OPTIONS STATISTICS_COLLECTION DISABLE_STATISTICS
DFHSTTI	Handles the NOTIFY request
DFHSTTRI	Interprets ST domain trace entries
DFHSTUE	Provides a SET_EXIT_STATUS routine to enable or disable a user exit.

Timer domain (TI)

The timer domain provides interval timing and alarm clock services for CICS domains. These are processes that cause an action to occur at some predetermined future time. This service (called "notifying") can be performed after a specific interval, at periodic intervals, at a specified time of day, or at a specific time of day every day.

Timer domain's specific gates

The specific gates provide access for other domains to functions that are provided by the TI domain.

TIMF gate, CONVERT_TIME function

This function converts a time value in any of a number of formats into the CICS ABSTIME format.

Input Parameters

DATE_STRING

A human-readable text string containing a date and time value in one of the following formats:

- RFC3339
- RFC1123
- RFC1036
- asctime()

TODCLOCK

The time of day expressed in the format of the z/Series Time-of-Day clock.

UTCTIME

The time expressed in the UTCtime format that is used in X.509 certificates.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
DAYNUM_INVALID
GMT_INCORRECT
INVALID_ABSTIME
MONTH_INVALID
TIME_INVALID
UNSUPPORTED_FORMAT
WEEKDAY_INVALID
YEAR_INVALID
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is INVALID:

```
NO_INPUT_TIME
```

ABSTIME

The time specified in ABSTIME format consisting of an eight-byte packed decimal number containing the number of milliseconds since midnight on 1 January 1900. The parameter can be specified in the range -9435484800000 to +255611289599999, corresponding to years from 1601 to 9999 respectively.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TIMF gate, FORMAT_TIME function

This function formats a time specified in ABSTIME format into one or more date or time formats.

Input Parameters

ABSTIME

A time specified in ABSTIME format consisting of an eight-byte packed decimal number containing the number of milliseconds since midnight on 1 January 1900. The parameter can be specified in the range -9435484800000 to +255611289599999, corresponding to years from 1601 to 9999 respectively.

ZONE

Optional Parameter

The time zone associated with the **ABSTIME** parameter.

Values for the parameter are:

```
GMT
LOCAL
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
DAYNUM_INVALID
GMT_INCORRECT
INVALID_ABSTIME
MONTH_INVALID
TIME_INVALID
UNSUPPORTED_FORMAT
WEEKDAY_INVALID
YEAR_INVALID
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is INVALID:

```
NO_INPUT_TIME
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BINARY_DAY

Optional Parameter

The day of the month, expressed as a binary number.

BINARY_DAY_OF_YEAR

Optional Parameter

The day of the year, expressed as a binary number.

BINARY_HOUR

Optional Parameter

The hours portion of the time in `hh:mm:ss.ddd` format, expressed as a binary number.

BINARY_MILLISECOND

Optional Parameter

The fractional seconds portion of the time in `hh:mm:ss.ddd` format, expressed as a binary number.

BINARY_MINUTE
Optional Parameter

The minute section of the time , expressed as a binary number.

BINARY_MONTH
Optional Parameter

The month of the year, expressed as a binary number.

BINARY_SECOND
Optional Parameter

The seconds portion of the time in `hh:mm:ss.ddd` format, expressed as a binary number.

BINARY_YEAR
Optional Parameter

The year, expressed as a binary number.

DAY
Optional Parameter

The day of the month.

DAY_OF_YEAR
Optional Parameter

The day of the year.

HOUR
Optional Parameter

The hours portion of the time in `hh:mm:ss.ddd` format.

JULIAN_DATE
Optional Parameter

The Julian date

MILLISECOND
Optional Parameter

The fractional seconds portion of the time in `hh:mm:ss.ddd` format, expressed as a binary number.

MINUTE
Optional Parameter

The minutes portion of the time in `hh:mm:ss.ddd` format, expressed as a binary number.

MONTH
Optional Parameter

The month of the year

RFC1123_DATE
Optional Parameter

The date in RFC1123 format.

RFC3339_DATE
Optional Parameter

The date in RFC3339 format.

SECOND
Optional Parameter

The whole seconds portion of the time in `hh:mm:ss.ddd` format, expressed as a binary number.

TIMER_UNITS
Optional Parameter

The time expressed in zSeries timer units (1/300 second).

WEEKDAY_NUMBER

Optional Parameter

The index of the day within the week. Sunday has an index of 0.

YEAR

Optional Parameter

The year.

TIMF gate, INQUIRE_TIME function

This function returns the current time in one or more formats.

Input Parameters

ZONE

Optional Parameter

The time zone for which the time is to be returned.

Values for the parameter are:

```
GMT
LOCAL
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
DAYNUM_INVALID
GMT_INCORRECT
INVALID_ABSTIME
MONTH_INVALID
TIME_INVALID
UNSUPPORTED_FORMAT
WEEKDAY_INVALID
YEAR_INVALID
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is INVALID:

```
NO_INPUT_TIME
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ABSTIME

A time specified in ABSTIME format consisting of an eight-byte packed decimal number containing the number of milliseconds since midnight on 1 January 1900. The parameter can be specified in the range -9435484800000 to +255611289599999, corresponding to years from 1601 to 9999 respectively.

BINARY_DAY

Optional Parameter

The day of the month, expressed as a binary number.

BINARY_DAY_OF_YEAR

Optional Parameter

The day of the year, expressed as a binary number.

BINARY_HOUR
Optional Parameter

The hours portion of the time in hh:mm:ss.ddd format, expressed as a binary number.

BINARY_MILLISECOND
Optional Parameter

The fractional seconds portion of the time in hh:mm:ss.ddd format, expressed as a binary number.

BINARY_MINUTE
Optional Parameter

The minute section of the time , expressed as a binary number.

BINARY_MONTH
Optional Parameter

The month of the year, expressed as a binary number.

BINARY_SECOND
Optional Parameter

The seconds portion of the time in hh:mm:ss.ddd format, expressed as a binary number.

BINARY_YEAR
Optional Parameter

The year, expressed as a binary number.

DAY
Optional Parameter

The day of the month.

DAY_OF_YEAR
Optional Parameter

The day of the year.

HOUR
Optional Parameter

The hours portion of the time in hh:mm:ss.ddd format.

JULIAN_DATE
Optional Parameter

The Julian date

MILLISECOND
Optional Parameter

The fractional seconds portion of the time in hh:mm:ss.ddd format, expressed as a binary number.

MINUTE
Optional Parameter

The minutes portion of the time in hh:mm:ss.ddd format, expressed as a binary number.

MONTH
Optional Parameter

The month of the year

RFC1123_DATE
Optional Parameter

The date in RFC1123 format.

RFC3339_DATE
Optional Parameter

The date in RFC3339 format.

SECOND
Optional Parameter

The whole seconds portion of the time in `hh:mm:ss.ddd` format, expressed as a binary number.

TIMER_UNITS
Optional Parameter

The time expressed in zSeries timer units (1/300 second).

TODCLOCK
Optional Parameter

The time of day expressed in the format of the z/Series Time-of-Day clock.

WEEKDAY_NUMBER
Optional Parameter

The index of the day within the week. Sunday has an index of 0.

YEAR
Optional Parameter

The year.

TISR gate, CANCEL function

The CANCEL function of the TISR gate is used to cancel a timer request that has already been initiated by one of these functions:

Input Parameters

TIMER_TOKEN
is the token that was returned when the timer request was made.

Output Parameters

REASON
The values for the parameter are:

```
REQUEST_NOT_FOUND
TOO_LATE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TISR gate, INQUIRE_EXPIRATION_TOKEN function

The INQUIRE_EXPIRATION_TOKEN function of the TISR gate is used by the dispatcher domain during its initialization.

Output Parameters

EXPIRATION_TOKEN
is a token used during initialization of the dispatcher domain.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TISR gate, REQUEST_NOTIFY_INTERVAL function

The REQUEST_NOTIFY_INTERVAL function of the TISR gate is used to request the timer domain to notify the calling domain after a specified real interval of time. The calling domain can request a NOTIFY on a one-off basis or periodically, and can specify the type of NOTIFY to be expected.

Input Parameters

DOMAIN_TOKEN
is a token that is to be passed as a parameter on the NOTIFY call.

NOTIFY_TYPE
specifies whether the attached task or the timer task is to be used to notify the calling domain after the specified interval of time.

Values for the parameter are:
ATTACHED_TASK
TIMER_TASK

PERIODIC_NOTIFY
specifies whether the requested NOTIFY is to be repeated at the specified interval until canceled (YES), or is to be just a one-off NOTIFY (NO).

Values for the parameter are:
NO
YES

STCK_INTERVAL
specifies an interval as a doubleword binary interval in stored clock (STCK) format, where bit 51 of the doubleword represents 1 microsecond.

ATTACH_MODE
Optional Parameter

is the optional TCB mode in which the attached NOTIFY task is to run.

Values for the parameter are:
CO
FO
QR
RO

ATTACH_PRIORITY
Optional Parameter

defines the priority, in the range 0 through 255, at which the requested NOTIFY task is to be attached.

ATTACH_TASK_TIMEOUT
Optional Parameter

defines the value, in seconds, of a wait in the attached task after which the dispatcher causes a timeout.

ORIGIN_DATE
Optional Parameter

defines the date from which the timer domain is to start the interval timing for this request. This parameter is mandatory if ORIGIN_TIME has been specified. It holds the origin date as MMDDYYYY.

ORIGIN_TIME
Optional Parameter

defines the local time of day from which the timer domain is to start the interval timing for this request. The value in decimal digits is specified in the form HHMMSS:

HH Hours in the range 00 through 23
MM Minutes in the range 00 through 59
SS Seconds in the range 00 through 59

Output Parameters

REASON
The values for the parameter are:
INVALID_INTERVAL

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TIMER_TOKEN
is the token that is returned by the timer domain. The timer token may be used to cancel the NOTIFY request.

TISR gate, REQUEST_NOTIFY_TIME_OF_DAY function

The REQUEST_NOTIFY_TIME_OF_DAY function of the TISR gate is used to inform the timer domain that an alarm call is required from the timer domain (that is, a NOTIFY) at the specified time of day. The calling domain can request a NOTIFY on a one-off basis or daily, and the type of NOTIFY to be expected.

Input Parameters

DOMAIN_TOKEN
is a token that is to be passed as a parameter on the NOTIFY call.

NOTIFY_TYPE
specifies whether the attached task or the timer task is to be used to notify the calling domain after the specified interval of time.

Values for the parameter are:
```
ATTACHED_TASK
TIMER_TASK
```

PERIODIC_NOTIFY
specifies whether the requested NOTIFY is to be repeated at the specified interval until canceled (YES), or is to be just a one-off NOTIFY (NO).

Values for the parameter are:
```
NO
YES
```

REQUESTED_TIME
is the time of day at which the NOTIFY function is to be invoked. The value is specified in the form HHMMSS.

ATTACH_MODE
Optional Parameter

is the optional TCB mode in which the attached NOTIFY task is to run.

Values for the parameter are:
```
CO
FO
QR
RO
```

ATTACH_PRIORITY
Optional Parameter

defines the priority, in the range 0 through 255, at which the requested NOTIFY task is to be attached.

ATTACH_TASK_TIMEOUT
Optional Parameter

defines the value, in seconds, of a wait in the attached task after which the dispatcher causes a timeout.

Output Parameters

REASON
The values for the parameter are:
```
TOO_LATE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TIMER_TOKEN
is the token that is returned by the timer domain. The timer token may be used to cancel the NOTIFY request.

Timer domain's generic gates

Table 53 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 53. Timer domain's generic gates

Gate	Trace	Functions	Format
DMDM	TI 0001 TI 0002	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

In initialization and quiesce processing, the timer domain performs only internal routines.

The timer domain does no termination processing.

Timer domain's generic formats

Table 54 describes the generic formats owned by the application domain and shows the functions performed on the calls.

Table 54. Timer domain's generic formats

Format	Calling module	Function
TISR	DFHTISR	NOTIFY

Note: In the descriptions of the formats that follow, the input parameters are input not to the timer domain, but to the domain being called by the timer domain. Similarly, the output parameters are output by the domain that was called by the timer domain, in response to the call.

TISR gate, NOTIFY function

The NOTIFY function of the TISR format is used by the timer domain itself to notify a domain after its requested interval or time has expired.

Input Parameters

DOMAIN_TOKEN
is a token that is to be passed as a parameter on the NOTIFY call.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHTIDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHTIDUF	Formats the timer domain's control blocks
DFHTISR	Handles the following requests: REQUEST_NOTIFY_INTERVAL REQUEST_NOTIFY_TIME_OF_DAY CANCEL INQUIRE_EXPIRATION_TOKEN
DFHTITRI	Interprets timer domain trace entries

Trace domain (TR)

The trace domain is used by CICS system code and user application programs to record details of the sequence of events occurring in the system. The basic unit of information created for this purpose is called a *trace entry*.

Trace domain's specific gates

The specific gates provide access for other domains to functions that are provided by the TR domain.

TRFT gate, TRACE_PUT function

This function is invoked to write a feature trace entry to the active trace destinations.

Input Parameters

FEATURE_TRACE_TOKEN
A token that the feature uses to identify itself to the CICS trace domain.

POINT_ID
is a number, unique within the calling domain, that identifies the trace entries made from this call.

DATA1
DATA2
DATA3
DATA4
DATA5
DATA6
DATA7
Optional Parameter

BLOCK descriptions of up to seven areas to be included in the data section of the trace entry. They appear in numerical order in the entry. Each data field is preceded by a 2-byte length field. If the DFHTRF4 format is in use, the parameters are BLOCK64 so that 64-bit data can be traced.
The maximum total length of data that can be traced in one call is 4000 bytes. The total length of all the data fields and all their 2-byte length fields must be within this limit.

EXCEPTION_TRACE
Optional Parameter

A binary value indicating whether the trace entry is for an exception trace.

Values for the parameter are:

NO

YES

RETURN_ADDR

Optional Parameter

is used by DFHTRFT to provide a return address for the module that originally requested the trace entry.

Output Parameters

REASON

The following values are returned when RESPONSE is INVALID:

DEREGISTERED_FEATURE

INV_FEATURE_TRACE_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRPT gate, TRACE_PUT function

This function is invoked to write a trace entry to the active trace destinations.

Input Parameters

POINT_ID

is a number, unique within the calling domain, that identifies the trace entries made from this call.

DATA1

DATA2

DATA3

DATA4

DATA5

DATA6

DATA7

Optional Parameter

are BLOCK descriptions of up to seven areas to be included in the data section of the trace entry. They appear in numerical order in the entry. Each data field is preceded by a 2-byte length field. If the DFHTRP4 format is in use, the parameters are BLOCK64 so that 64-bit data can be traced.

The maximum total length of data that can be traced in one call is 4000 bytes. The total length of all the data fields and all their 2-byte length fields must be within this limit.

DOMAIN_TOKEN

Optional Parameter

A token that identifies the calling domain to the trace domain.

RETURN_ADDR

Optional Parameter

is used by DFHTRPT to provide a return address for the module that originally requested the trace entry.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRSR gate, ACTIVATE_TRAP function

The ACTIVATE_TRAP function of the TRSR gate is used to activate the FE global trap/trace exit (DFHTRAP).

Output Parameters

REASON

The values for the parameter are:

DFHTRAP_NOT_FOUND
DFHTRAP_UNUSABLE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRSR gate, DEACTIVATE_TRAP function

The DEACTIVATE_TRAP function of the TRSR gate is used to deactivate the FE global trap/trace exit (DFHTRAP).

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON

Optional Parameter

The values for the parameter are:

AUX_TRACE_STOPPED
CANT_GET_AUX_BUFFER
CANT_GET_GTF_BUFFER
DFHTRAO_NOT_AVAILABLE
DFHTRAP_NOT_FOUND
DFHTRAP_UNUSABLE
INVALID_AUTOSWITCH_STATUS
INVALID_TABLE_SIZE
NO_SPACE
OPEN_FAILED

TRSR gate, INQUIRE_AUXILIARY_TRACE function

The INQUIRE_AUXILIARY_TRACE function of the TRSR gate is used to return the current state of the auxiliary trace.

Output Parameters

AUTOSWITCH_STATUS

Indicates whether or not an automatic switch to the inactive CICS auxiliary extent is to occur once only when the current extent fills up, or that such automatic switching should occur "continuously" whenever the current extent fills up.

Values for the parameter are:

CONTINUOUS
OFF
ONCE

AUXILIARY_STATUS

Indicates the current status of auxiliary trace.

Values for the parameter are:

PAUSED
STARTED

STOPPED

EXTENT
indicates the currently active CICS auxiliary trace extent; that is, the extent that is already in use or is used if CICS auxiliary tracing is started.

Values for the parameter are:
DFHAUXT
DFHBUXT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON
Optional Parameter

The values for the parameter are:
AUX_TRACE_STOPPED
CANT_GET_AUX_BUFFER
CANT_GET_GTF_BUFFER
DFHTRAO_NOT_AVAILABLE
DFHTRAP_NOT_FOUND
DFHTRAP_UNUSABLE
INVALID_AUTOSWITCH_STATUS
INVALID_TABLE_SIZE
NO_SPACE
OPEN_FAILED

TRSR gate, INQUIRE_GTF_TRACE function

The INQUIRE_GTF_TRACE function of the TRSR gate is used to return the current state of the GTF trace.

Output Parameters

GTF_STATUS
indicates whether CICS tracing to GTF is active (STARTED) or inactive (STOPPED).

Values for the parameter are:
STARTED
STOPPED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON
Optional Parameter

The values for the parameter are:
AUX_TRACE_STOPPED
CANT_GET_AUX_BUFFER
CANT_GET_GTF_BUFFER
DFHTRAO_NOT_AVAILABLE
DFHTRAP_NOT_FOUND
DFHTRAP_UNUSABLE
INVALID_AUTOSWITCH_STATUS
INVALID_TABLE_SIZE
NO_SPACE
OPEN_FAILED

TRSR gate, INQUIRE_INTERNAL_TRACE function

The INQUIRE_INTERNAL_TRACE function of the TRSR gate is used to return the status of the internal trace and the current size of the internal trace table.

Output Parameters

INTERNAL_STATUS

indicates whether internal trace is active (STARTED) or inactive (STOPPED).

Values for the parameter are:

```
STARTED
STOPPED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TABLE_SIZE

is the size of the current internal trace table in KB (KB equals 1024 bytes).

REASON

Optional Parameter

The values for the parameter are:

```
AUX_TRACE_STOPPED
CANT_GET_AUX_BUFFER
CANT_GET_GTF_BUFFER
DFHTRAO_NOT_AVAILABLE
DFHTRAP_NOT_FOUND
DFHTRAP_UNUSABLE
INVALID_AUTOSWITCH_STATUS
INVALID_TABLE_SIZE
NO_SPACE
OPEN_FAILED
```

TRSR gate, PAUSE_AUXILIARY_TRACE function

The PAUSE_AUXILIARY_TRACE function of the TRSR gate is used to stop auxiliary tracing without closing the currently active extent.

Output Parameters

REASON

The values for the parameter are:

```
AUX_TRACE_STOPPED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRSR gate, SET_AUX_TRACE_AUTOSWITCH function

The SET_AUX_TRACE_AUTOSWITCH function of the TRSR gate is used to allow the autoswitch facility for the CICS auxiliary trace data set to be enabled or disabled.

Input Parameters

AUTOSWITCH_STATUS

Indicates whether or not an automatic switch to the inactive CICS auxiliary extent is to occur once only when the current extent fills up, or that such automatic switching should occur "continuously" whenever the current extent fills up.

Values for the parameter are:

```
CONTINUOUS
```

OFF
ONCE

Output Parameters

REASON

The values for the parameter are:

INVALID_AUTOSWITCH_STATUS

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRSR gate, SET_INTERNAL_TABLE_SIZE function

The SET_INTERNAL_TABLE_SIZE function of the TRSR gate is used to change the size of the internal trace table during a CICS run.

Input Parameters

TABLE_SIZE

The required size for the internal trace table, specified as a number of KB (1 KB equals 1024 bytes). This parameter is rounded up to the nearest multiple of 4 KB. The lower limit is 16 KB. The upper limit is 1048576 KB (1 GB).

CICS obtains MVS 64-bit (above-the-bar) storage (outside the CICS DSAs) for the internal trace table.

When you change the size of the internal trace table, internal tracing is suspended while CICS makes the changes. Data that was in the existing internal trace table is deleted.

To change the size of an internal trace table in 64-bit storage, CICS issues a z/OS request to adjust the guard area to increase or decrease the storage that can be used by the trace table. If insufficient or no additional storage is available, CICS does not increase the size of the trace table.

Output Parameters

REASON

The values for the parameter are:

INVALID_TABLE_SIZE
NO_SPACE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRSR gate, START_AUXILIARY_TRACE function

The START_AUXILIARY_TRACE function of the TRSR gate is used to open the current auxiliary trace extent (if it is closed) and start tracing to it.

Output Parameters

REASON

The values for the parameter are:

CANT_GET_AUX_BUFFER
DFHTRAO_NOT_AVAILABLE
OPEN_FAILED

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRSR gate, START_GTF_TRACE function

The START_GTF_TRACE function of the TRSR gate is used to start the tracing of CICS activity to GTF. It is the responsibility of the user to ensure that GTF has been started in MVS with at least TRACE=USR. If it has not, CICS issues the GTF calls but they are ignored by GTF.

Output Parameters

REASON

The values for the parameter are:

CANT_GET_GTF_BUFFER

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRSR gate, START_INTERNAL_TRACE function

The START_INTERNAL_TRACE function of the TRSR gate is used to activate tracing to the internal trace table.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON

Optional Parameter

The values for the parameter are:

AUX_TRACE_STOPPED
CANT_GET_AUX_BUFFER
CANT_GET_GTF_BUFFER
DFHTRAO_NOT_AVAILABLE
DFHTRAP_NOT_FOUND
DFHTRAP_UNUSABLE
INVALID_AUTOSWITCH_STATUS
INVALID_TABLE_SIZE
NO_SPACE
OPEN_FAILED

TRSR gate, STOP_AUXILIARY_TRACE function

The STOP_AUXILIARY_TRACE function of the TRSR gate is used to stop auxiliary tracing and close the currently active auxiliary trace extent.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON

Optional Parameter

The values for the parameter are:

AUX_TRACE_STOPPED
CANT_GET_AUX_BUFFER
CANT_GET_GTF_BUFFER
DFHTRAO_NOT_AVAILABLE
DFHTRAP_NOT_FOUND
DFHTRAP_UNUSABLE
INVALID_AUTOSWITCH_STATUS
INVALID_TABLE_SIZE

NO_SPACE
OPEN_FAILED

TRSR gate, STOP_GTF_TRACE function

The STOP_GTF_TRACE function of the TRSR gate is used to stop tracing of CICS activity to GTF.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON

Optional Parameter

The values for the parameter are:

AUX_TRACE_STOPPED
CANT_GET_AUX_BUFFER
CANT_GET_GTF_BUFFER
DFHTRAO_NOT_AVAILABLE
DFHTRAP_NOT_FOUND
DFHTRAP_UNUSABLE
INVALID_AUTOSWITCH_STATUS
INVALID_TABLE_SIZE
NO_SPACE
OPEN_FAILED

TRSR gate, STOP_INTERNAL_TRACE function

The STOP_INTERNAL_TRACE function of the TRSR gate is used to deactivate tracing to the internal trace table.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON

Optional Parameter

The values for the parameter are:

AUX_TRACE_STOPPED
CANT_GET_AUX_BUFFER
CANT_GET_GTF_BUFFER
DFHTRAO_NOT_AVAILABLE
DFHTRAP_NOT_FOUND
DFHTRAP_UNUSABLE
INVALID_AUTOSWITCH_STATUS
INVALID_TABLE_SIZE
NO_SPACE
OPEN_FAILED

TRSR gate, SWITCH_AUXILIARY_EXTENTS function

The SWITCH_AUXILIARY_EXTENTS function of the TRSR gate allows switching from one auxiliary trace extent to the other.

Output Parameters

REASON

The values for the parameter are:

OPEN_FAILED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Trace domain's generic gates

Table 55 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 55. Trace domain's generic gates

Gate	Trace	Functions	Format
DMDM	ST 0001 ST 0002	PRE_INITIALIZE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
KETI	TR 0201 TR 0202	NOTIFY_RESET	KETI

In preinitialization processing, the trace domain establishes the initial tracing status:

- A suitably sized internal trace table is created.
- If internal tracing or GTF tracing is required, set on the trace master flag.
- If required, start internal tracing and CICS GTF tracing.
- As required, set the auxiliary tracing switch status to 'started' or 'stopped'.

The information always comes from the system initialization parameters - trace domain will always initialise with a cold start.

In initialization processing, the trace domain starts auxiliary tracing if it is required.

The trace domain does no quiesce processing.

In termination processing, the trace domain stops auxiliary tracing if it is active.

Modules

Module	Function
DFHTRAO	Auxiliary trace output subroutines for interfacing with BSAM. Loaded separately below the 16 MB line when auxiliary trace is started.
DFHTRAP	FE global trap/trace exit program. Loaded separately above the 16 MB line when the trap is activated.
DFHTRDM	Processes requests to the DMDM gate of the trace domain. Part of the DFHSIP load module.
DFHTRPT	Processes requests to the TRPT gate of the trace domain. Part of the DFHSIP load module.
DFHTRPX	Processes, within the calling domain, all TRACE_PUT requests that do not require special handling. Part of the DFHSIP load module.
DFHTRSR	Processes requests to the TRSR and KETI gates of the trace domain. Part of the DFHSIP load module.
DFHTRSU	Processes domain subroutine requests of format TRSU. Part of the DFHSIP load module.

Temporary storage domain (TS domain)

The temporary storage domain manages requests for temporary storage to be used by CICS or user applications.

Temporary storage for a CICS region can be in three locations:

Main storage
Main temporary storage is in 64-bit (above-the-bar) storage in the CICS region.

Auxiliary storage
Auxiliary temporary storage is in a nonindexed VSAM data set named DFHTEMP. You define the available space and any additional extents when you set up this data set. Some 31-bit storage is used in the CICS region for VSAM buffers to make control intervals available from the VSAM data set.

Shared temporary storage pools in a z/OS coupling facility
Shared temporary storage pools (or TS pools) are in a z/OS coupling facility managed by a temporary storage data sharing server (or TS server).

CICS uses the temporary storage domain for the following tasks:

- Manage temporary storage models
- Locate a matching temporary storage model for a specified temporary storage queue name
- Create temporary storage queues
- Add items to temporary storage queues
- Retrieve items from temporary storage queues
- Delete temporary storage queues
- Inquire on and browse existing temporary storage queues
- Interact with shared temporary storage pools

For more information about temporary storage, see CICS temporary storage overview in Improving performance.

Temporary storage domain's specific gates

The specific gates provide access for other domains to functions that are provided by the TS domain.

TSAD gate, ADD_REPLACE_TSMODEL function

Add or replace a temporary storage model

Input Parameters

MAIN
A binary value that specifies whether queues matching this model are to be held in main storage. If **MAIN(NO)** is specified, the queues are held in auxiliary storage.

Values for the parameter are:
NO
YES

PREFIX
The character string that is to be used as the prefix for this model. The prefix may be up to 16 characters in length.

RECOVERABLE
A binary value that specifies whether queues matching this model are to be recoverable.

Values for the parameter are:
NO
YES

SECURITY
A binary value that specifies whether security checking is to be performed for queues matching this model.

Values for the parameter are:
NO
YES

TSMODEL_NAME
The name of the temporary storage model.

POOL_NAME
Optional Parameter

The 8-character name of the shared TS pool associated with the model.

REMOTE_PREFIX
Optional Parameter

The character string that is to be used as the prefix on the remote system. The prefix may be up to 16 characters in length.

SYSID
Optional Parameter

The name of the remote system where the temporary storage queue resides.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DUPLICATE_PREFIX
INVALID_NAME
INVALID_PREFIX
INVALID_REMOTE_PREFIX
RDO_DISABLED

DUPLICATE_PREFIX_NAME
When **REASON(DUPLICATE_PREFIX)** is returned, the name of the existing prefix that clashes with the prefix for this model.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSAD gate, DELETE_TSMODEL function

Delete a temporary storage model.

Input Parameters

TSMODEL_NAME
The name of the model to be deleted.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NOT_FOUND
RDO_DISABLED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSAD gate, INITIALISE function

Initialize temporary storage models from the catalog.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
DUPLICATE_PREFIX
INVALID_NAME
INVALID_PREFIX
INVALID_REMOTE_PREFIX
NOT_FOUND
RDO_DISABLED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSBR gate, CHECK_PREFIX function

Checks whether there are any queues with the prefix provided.

Input Parameters

PREFIX
The queue prefix to be checked.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
DUPLICATE
NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSBR gate, END_BROWSE function

Ends the browse.

Input Parameters

BROWSE_TOKEN
A token that identifies the browse session.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
DUPLICATE
NOT_FOUND
QUEUE_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSBR gate, GET_NEXT function

Returns information about the next queue in the browse.

Input Parameters

BROWSE_TOKEN
A token that represents the browse session.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END

QUEUE_NAME
is the name of the queue.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CREATION_TIME
Optional Parameter

is the time at which the queue was created.

LAST_REFERENCED_TIME
Optional Parameter

is the time at which the queue was last referenced.

MAXIMUM_ITEM_LENGTH
Optional Parameter

is the length of the longest item in the queue.

MINIMUM_ITEM_LENGTH
Optional Parameter

is the length of the shortest item in the queue.

RECOVERABLE
Optional Parameter

returns whether the queue is recoverable or not.

Values for the parameter are:
NO
YES

STORAGE_TYPE
Optional Parameter

indicates whether the queue is held in main or auxiliary storage.

Values for the parameter are:
AUXILIARY
MAIN

TOTAL_ITEMS
Optional Parameter

is the total number of items in the queue on completion of the operation.

TOTAL_LENGTH
Optional Parameter

is the sum of the lengths of all the items in the queue.

TRANSID
Optional Parameter

is the id of the transaction whcih created the queue.

TSBR gate, INQUIRE_QUEUE function

Input Parameters

QUEUE_NAME
is the name of the queue being created or appended to.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
QUEUE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CREATION_TIME
Optional Parameter

is the time at which the queue was created.

EXPIRYINT
Optional Parameter

Displays the expiry interval, in hours, for temporary storage queues associated with this TS model.

EXPIRYINTMIN
Optional Parameter

Displays the expiry interval, in minutes, for temporary storage queues associated with this TS model.

LAST_REFERENCED_TIME
Optional Parameter

is the time at which the queue was last referenced.

MAXIMUM_ITEM_LENGTH
Optional Parameter

is the length of the longest item in the queue.

MINIMUM_ITEM_LENGTH
Optional Parameter

is the length of the shortest item in the queue.

QUEUE_TYPE
Optional Parameter

The type of queue.

Values for the parameter are:
CICS
USER

RECOVERABLE
Optional Parameter

returns whether the queue is recoverable or not.

Values for the parameter are:
NO
YES

STORAGE_TYPE
Optional Parameter

indicates whether the queue is held in main or auxiliary storage.

Values for the parameter are:
AUXILIARY

MAIN

TOTAL_ITEMS
Optional Parameter

is the total number of items in the queue on completion of the operation.

TOTAL_LENGTH
Optional Parameter

is the sum of the lengths of all the items in the queue.

TRANSID
Optional Parameter

is the id of the transaction whcih created the queue.

TSBR gate, START_BROWSE function

Input Parameters

QUEUE_NAME
Optional Parameter

is the name of the queue being created or appended to.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END
DUPLICATE
NOT_FOUND
QUEUE_NOT_FOUND

BROWSE_TOKEN
A token that represents the browse session.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSMB gate, END_BROWSE function

End the browse operation on the set of temporary storage models.

Input Parameters

BROWSE_TOKEN
See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

Output Parameters

REASON
The values for the parameter are:

INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSMB gate, GET_NEXT function

In a browse operation, return information about a temporary storage model.

Input Parameters

BROWSE_TOKEN
See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
INVALID_BROWSE_TOKEN
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSMODEL_NAME

The name of the temporary storage model.

MAIN

Optional Parameter

A binary value that indicates whether the temporary storage queues that match this model are to be held in main storage. If MAIN(NO) is specified, the queues are held on auxiliary storage.

Values for the parameter are:

```
NO
YES
```

POOL_NAME

Optional Parameter

The name of the shared temporary storage pool used with the model.

PREFIX

Optional Parameter

The character string used as a prefix for queues that match the temporary storage model.

RECOVERABLE

Optional Parameter

A binary value that indicates whether the queue is recoverable.

Values for the parameter are:

```
NO
YES
```

REMOTE_PREFIX

Optional Parameter

The character string used as a prefix on a remote system for queues that match the temporary storage model.

SECURITY

Optional Parameter

A binary value that indicates whether security checking is to be performed for queues that match this model.

Values for the parameter are:

```
NO
YES
```

SYSID

Optional Parameter

The name of the connection to the remote system where the temporary storage queue resides.

TSMB gate, INQUIRE_TSMODEL function

List the attributes of a temporary storage model.

Input Parameters

TSMODEL_NAME
The name of the temporary storage model.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

EXPIRYINTMIN
Optional Parameter

The number of minutes after which a queue matching this model becomes eligible for automatic deletion if it is not referenced.

EXPIRY_INTERVAL
Optional Parameter

The number of hours after which a queue matching this model becomes eligible for automatic deletion if it is not referenced.

MAIN
A binary value that specifies whether queues matching this model are to be held in main temporary storage. If **MAIN(NO)** is specified, the queues are held in auxiliary temporary storage.

Values for the parameter are:
NO
YES

POOL_NAME
Optional Parameter

The 8-character name of a shared TS pool associated with the model.

PREFIX
The character string that is to be used as the prefix for this model. The prefix can be up to 16 characters in length.

RECOVERABLE
A binary value that specifies whether queues matching this model are to be recoverable.

Values for the parameter are:
NO
YES

REMOTE_NAME
Optional Parameter

The name of the connection that links the local system to the remote system where the temporary storage queue resides

REMOTE_PREFIX
Optional Parameter

The character string that is to be used as the prefix on the remote system. The prefix can be up to 16 characters in length.

SECURITY
A binary value that specifies whether security checking is to be performed for queues matching this model.

Values for the parameter are:
NO
YES

SYSID
Optional Parameter

The name of the remote system where the temporary storage queue resides.

TSMB gate, INQUIRE_EXPINT function

Inquire on the smallest expiry interval value for temporary storage models in the CICS region.

Input Parameters

None.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MIN_EXPINT
The smallest nonzero value, in hours, for an expiry interval in the installed TSMODEL resource definitions in the CICS region.

TSMB gate, MATCH function

Find the model that is the best match with the queue name provided.

Input Parameters

QUEUE_NAME
The name of the queue to be matched with temporary storage models.

SEARCH
Optional Parameter

Specifies whether the search is confined to temporary storage models, or extended to the cache of models for the current unit of work.

Values for the parameter are:
```
MODELS_ONLY
UOW_CACHE
```

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
BROWSE_END
INVALID_BROWSE_TOKEN
NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

MAIN
Optional Parameter

A binary value that indicates whether the temporary storage queues that match this model are to be held in main storage. If MAIN(NO) is specified, the queues are held on auxiliary storage.

Values for the parameter are:
```
NO
YES
```

POOL_NAME
Optional Parameter

The name of the shared temporary storage pool used with the model.

POOL_TOKEN
Optional Parameter

A token that identifies the temporary storage pool associated with the pool name.

PREFIX
Optional Parameter

The character string used as a prefix for queues that match the temporary storage model.

RECOVERABLE
Optional Parameter

A binary value that indicates whether the queue is recoverable.

Values for the parameter are:
```
NO
YES
```

REMOTE_NAME
Optional Parameter

The name of the temporay storage queue on the remote system.

REMOTE_PREFIX
Optional Parameter

The character string that is used as the prefix on the remote system.

SECURITY
Optional Parameter

A binary value that indicates whether security checking is to be performed for queues that match this model.

Values for the parameter are:
```
NO
YES
```

SYSID
Optional Parameter

The name of the connection to the remote system where the temporary storage queue resides.

TSMODEL_NAME
Optional Parameter

The name of the matching temporary storage model.

TSMB gate, START_BROWSE function

Start a browse operation on temporary storage models.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
BROWSE_END
INVALID_BROWSE_TOKEN
NOT_FOUND
```

BROWSE_TOKEN
See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSPT gate, GET function

This function retrieves the first item in a "put" queue.

Input Parameters

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

QUEUE_NAME
is the name of the queue being created or appended to.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
QUEUE_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:

```
NO
YES
```

TSPT gate, GET_RELEASE function

This function retrieves and deletes the first item in a "put" queue. If the queue has one item, the queue is deleted.

Input Parameters

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

QUEUE_NAME
is the name of the queue being created or appended to.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
LOCKED
QUEUE_DELETED
QUEUE_NOT_FOUND
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

TSPT gate, GET_RELEASE_SET function

This function retrieves the first item in a "put" queue into set storage and then deletes it. If the queue has one item, the queue is deleted.

Input Parameters

QUEUE_NAME
is the name of the queue being created or appended to.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
LOCKED
QUEUE_DELETED
QUEUE_NOT_FOUND

ITEM_DATA
returns the address and length of the item data.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

TSPT gate, GET_SET function

This function retrieves the first item in a "put" queue into a set storage area.

Input Parameters

QUEUE_NAME
is the name of the queue being created or appended to.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
QUEUE_NOT_FOUND

ITEM_DATA
returns the address and length of the item data.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

TSPT gate, PUT function

If the queue does not already exist, this function creates a queue with the single item provided.

Input Parameters

ITEM_DATA
is the address and length of the item being written.

QUEUE_NAME
is the name of the queue being created or appended to.

SUSPEND
indicates whether or not the request will be suspended if there is insufficient auxiliary storage to satisfy the request. This option is ignored if the queue is in main storage.

Values for the parameter are:
NO
YES

BMS
Optional Parameter

indicates whether or not BMS owns this queue.

Values for the parameter are:
NO
YES

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

IC Optional Parameter

this option indicates whether or not Interval Control owns this queue. If the queue already exists and is an IC queue then IC(YES) must be specified on the request. Otherwise an INVALID response is returned.

Values for the parameter are:
NO
YES

IC_DATA
Optional Parameter

is the address and length of an optional ICE.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
DUPLICATE_NAME
INSUFFICIENT_STORAGE
INVALID_LENGTH
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE

IO_ERROR
LOCKED
QUEUE_DELETED
QUEUE_FULL
QUEUE_REMOTE

QUEUE_CREATION_TIME
returns the store clock time at which the queue was created.

RECOVERABLE
returns whether the queue is recoverable or not.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSPT gate, PUT_REPLACE function

If the queue does not exist, this function creates the queue with the item provided. If the queue does exist, the first item in the queue is replaced by the item provided.

Input Parameters

ITEM_DATA
is the address and length of the item being written.

QUEUE_NAME
is the name of the queue being created or appended to.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_LENGTH
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
LOCKED
QUEUE_DELETED
QUEUE_REMOTE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSPT gate, RELEASE function

This function deletes a "put" queue.

Input Parameters

QUEUE_NAME
is the name of the queue being created or appended to.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
LOCKED
QUEUE_DELETED
QUEUE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSQR gate, ALLOCATE_SET_STORAGE function

This function allocates set storage of the requested length.

Input Parameters

REQUESTED_LENGTH
The intended length of the storage to be allocated.

CALLER
Optional Parameter

indicates whether this request originated from an EXEC or macro call. The default is MACRO.

Values for the parameter are:
EXEC
MACRO

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_LENGTH
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
ITEM_NOT_FOUND
LOCKED
QUEUE_DELETED
QUEUE_FULL
QUEUE_NOT_FOUND
QUEUE_REFERENCED
QUEUE_REMOTE

ADDRESS
The address of the allocated storage.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACTUAL_LENGTH
Optional Parameter

The actual length of the allocated storage.

TSQR gate, DELETE function

This function deletes the specified queue.

Input Parameters

QUEUE_NAME
The name of the queue to be deleted.

CALLER
Optional Parameter

Indicates whether this request originated from an EXEC or macro call. The default is MACRO.

Values for the parameter are:
EXEC

MACRO

LAST_REFERENCED_TIME
Optional Parameter

The time of the last reference to the queue.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
LOCKED
QUEUE_DELETED
QUEUE_NOT_FOUND
QUEUE_REFERENCED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSQR gate, READ_INTO function

This function reads the specified queue item into a buffer provided by the caller. The read cursor for the queue is set to the item number provided. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data is truncated (but an OK response is returned).

Input Parameters

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

ITEM_NUMBER
is the number of the item to be updated.

QUEUE_NAME
is the name of the queue being created or appended to.

CALLER
Optional Parameter

indicates whether this request originated from an EXEC or macro call. The default is MACRO.

Values for the parameter are:
EXEC
MACRO

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
ITEM_NOT_FOUND
QUEUE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

TOTAL_ITEMS
Optional Parameter

is the total number of items in the queue on completion of the operation.

TSQR gate, READ_NEXT_INTO function

This function increments the read cursor by one and reads that item number into the buffer provided by the caller. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data will have been truncated.

Input Parameters

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

QUEUE_NAME
is the name of the queue being created or appended to.

CALLER
Optional Parameter

indicates whether this request originated from an EXEC or macro call. The default is MACRO.

Values for the parameter are:
EXEC
MACRO

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
ITEM_NOT_FOUND
QUEUE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

ITEM_NUMBER
Optional Parameter

returns the number of the item just read.

TOTAL_ITEMS
Optional Parameter

is the total number of items in the queue on completion of the operation.

TSQR gate, READ_NEXT_SET function

This function increments the queue's read cursor by one and reads that item number into a storage area obtained by TS.

Input Parameters

QUEUE_NAME
is the name of the queue being created or appended to.

CALLER
Optional Parameter

indicates whether this request originated from an EXEC or macro call. The default is MACRO.

Values for the parameter are:
EXEC
MACRO

SET_STORAGE_CLASS
Optional Parameter

specifies the class of storage into which the item will be read. This may be either TASK (the default) or TERMINAL. If TERMINAL is specified, the item is read into a TIOA.

Values for the parameter are:
TASK
TERMINAL

TCTTE_ADDRESS
Optional Parameter

is the address of the TCTTE - required if SET_STORAGE_CLASS(TERMINAL) is specified.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
ITEM_NOT_FOUND
QUEUE_NOT_FOUND

ITEM_DATA
returns the address and length of the item data.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

ITEM_NUMBER
Optional Parameter

returns the number of the item just read.

TOTAL_ITEMS
Optional Parameter

is the total number of items in the queue on completion of the operation.

TSQR gate, READ_SET function

This function reads the specified queue item into a storage area obtained by TS. The read cursor for the queue is set to the input item number.

Input Parameters

ITEM_NUMBER
is the number of the item to be updated.

QUEUE_NAME
is the name of the queue being created or appended to.

CALLER
Optional Parameter

indicates whether this request originated from an EXEC or macro call. The default is MACRO.

Values for the parameter are:
EXEC
MACRO

SET_STORAGE_CLASS
Optional Parameter

specifies the class of storage into which the item will be read. This may be either TASK (the default) or TERMINAL. If TERMINAL is specified, the item is read into a TIOA.

Values for the parameter are:
TASK
TERMINAL

TCTTE_ADDRESS
Optional Parameter

is the address of the TCTTE - required if SET_STORAGE_CLASS(TERMINAL) is specified.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
ITEM_NOT_FOUND
QUEUE_NOT_FOUND

ITEM_DATA
returns the address and length of the item data.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

TOTAL_ITEMS
Optional Parameter

is the total number of items in the queue on completion of the operation.

TSQR gate, REWRITE function

This function updates the specified item in an existing queue. The read cursor is unchanged.

Input Parameters

ITEM_DATA
is the address and length of the item being written.

ITEM_NUMBER
is the number of the item to be updated.

QUEUE_NAME
is the name of the queue being created or appended to.

SUSPEND
indicates whether or not the request will be suspended if there is insufficient auxiliary storage to satisfy the request. This option is ignored if the queue is in main storage.

Values for the parameter are:
NO
YES

CALLER
Optional Parameter

indicates whether this request originated from an EXEC or macro call. The default is MACRO.

Values for the parameter are:
EXEC
MACRO

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_LENGTH
INVALID_QUEUE_NAME
INVALID_QUEUE_TYPE
IO_ERROR
ITEM_NOT_FOUND
LOCKED
QUEUE_DELETED
QUEUE_NOT_FOUND
QUEUE_REMOTE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TOTAL_ITEMS
Optional Parameter

is the total number of items in the queue on completion of the operation.

TSQR gate, WRITE function

If the queue does not exist, this function creates a queue with the single item provided, and the queue's "read cursor" is set to zero.

Input Parameters

ITEM_DATA
is the address and length of the item being written.

QUEUE_NAME
is the name of the queue being created or appended to.

STORAGE_TYPE
indicates whether the queue is to be created in main or auxiliary storage. Note that this option is ignored if the queue already exists.

Values for the parameter are:
```
AUXILIARY
MAIN
```

SUSPEND
indicates whether or not the request will be suspended if there is insufficient auxiliary storage to satisfy the request. This option is ignored if the queue is in main storage.

Values for the parameter are:
```
NO
YES
```

BMS
Optional Parameter

indicates whether or not BMS owns this queue.

Values for the parameter are:
```
NO
YES
```

CALLER
Optional Parameter

indicates whether this request originated from an EXEC or macro call. The default is MACRO.

Values for the parameter are:
```
EXEC
MACRO
```

FMH
Optional Parameter

indicates whether the data contains an FMH.

Values for the parameter are:
```
NO
YES
```

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
INSUFFICIENT_STORAGE
INVALID_LENGTH
INVALID_QUEUE_NAME
```

INVALID_QUEUE_TYPE
IO_ERROR
LOCKED
QUEUE_DELETED
QUEUE_FULL
QUEUE_REMOTE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TOTAL_ITEMS
Optional Parameter

is the total number of items in the queue on completion of the operation.

TSRM gate, INQUIRE_QUEUE function

Determine whether a temporary storage queue exists.

Input Parameters

QUEUE_NAME
The name of the temporary storage queue.

QUEUE_CREATION_TIME
Optional Parameter

The time the queue was created.

Output Parameters

QUEUE_EXISTS
A binary value indicating whether the named queue exists.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSH gate, ADD_POOL function

Create a temporary storage pool.

Input Parameters

POOL_NAME
The name of the pool.

Output Parameters

POOL_TOKEN
A token that identifies the new temporary storage pool.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSH gate, DELETE function

This function deletes the specified queue.

Input Parameters

QUEUE_NAME
is the name of the queue being created or appended to.

POOL_TOKEN
Mandatory Parameter

is a token for the shared TS pool.

TRANSACTION_NUMBER
Optional Parameter

is the 4-byte transaction number (in packed-decimal format).

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
IO_ERROR
QUEUE_NOT_FOUND
SERVER_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSH gate, END_BROWSE function

End a browse operation on a set of temporary storage queues.

Input Parameters

BROWSE_TOKEN
A token that identifies the browse operation. See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
IO_ERROR
QUEUE_NOT_FOUND
SERVER_ERROR
TSPOOL_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSH gate, END_TSPOOL_BROWSE function

End a browse operation on temporary storage pools.

Input Parameters

BROWSE_TOKEN
A token that identifies the browse operation. See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
IO_ERROR
QUEUE_NOT_FOUND
SERVER_ERROR
TSPOOL_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSH gate, GET_NEXT function

Return the next temporary storage queue in a browse operation.

Input Parameters

BROWSE_TOKEN

See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
IO_ERROR
SERVER_ERROR
```

QUEUE_NAME

The name of the temporary storage queue.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LAST_REFERENCED_TIME

Optional Parameter

The time at which the temporary storage queue was last referenced.

MAXIMUM_ITEM_LENGTH

Optional Parameter

The maximum size of an item in the temporary storage queue.

MINIMUM_ITEM_LENGTH

Optional Parameter

The minimum size of an item in the temporary storage queue.

TOTAL_ITEMS

Optional Parameter

The total number of items in the temporary storage queue.

TOTAL_LENGTH

Optional Parameter

The length of the temporary storage queue.

TRANSID

Optional Parameter

The identifier of the transaction that created the temporary storage queue.

TSSH gate, GET_NEXT_TSPOOL function

In a browse operation, return information about a temporary storage pool.

Input Parameters

BROWSE_TOKEN

See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
```

POOL_NAME

The name of the temporary storage pool.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONNECTED
Optional Parameter

A binary value indicating whether the temporary storage pool is connected.

Values for the parameter are:
NO
YES

POOL_TOKEN
Optional Parameter

A token that represents the temporary storage pool.

TSSH gate, INITIALISE function

Initialize the Shared TS interface.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_LENGTH
INVALID_QUEUE_NAME
IO_ERROR
ITEM_NOT_FOUND
MAXIMUM_QUEUES_REACHED
POOL_NAME_NOT_FOUND
QUEUE_FULL
QUEUE_NOT_FOUND
SERVER_ERROR
SYSID_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSH gate, INQUIRE_POOL_TOKEN function

Return a token for the pool corresponding to the SYSID provided.

Input Parameters

POOL_NAME
The name of the pool being inquired upon.

SYSID
The name of the SYSID being inquired upon.

SYSID_TABLE_TOKEN
Optional Parameter

A token that represents the SYSID table.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
POOL_NAME_NOT_FOUND
SYSID_NOT_FOUND

POOL_TOKEN
A token for the shared TS pool.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSH gate, INQUIRE_QUEUE function

Inquire on the attributes of a temporary storage queue.

Input Parameters

QUEUE_NAME
The name of the queue.

KEY_COMPARISON
Optional Parameter

Specifies the constraints on the inquiry.

Values for the parameter are:
EQ
GT
GTEQ

POOL_TOKEN
Optional Parameter

A token that identifies a pool containing the specified queue.

TRANSACTION_NUMBER
Optional Parameter

The 4-byte transaction number (in packed-decimal format).

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
IO_ERROR
QUEUE_NOT_FOUND
SERVER_ERROR

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

LAST_REFERENCED_TIME
Optional Parameter

The time at which the queue was last referenced.

MAXIMUM_ITEM_LENGTH
Optional Parameter

The length of the longest item in the queue.

MINIMUM_ITEM_LENGTH
Optional Parameter

The length of the shortest item in the queue.

OUTPUT_QUEUE_NAME
Optional Parameter

The name of the queue whose information is returned. Note that this might differ from **QUEUE_NAME** unless **KEY_COMPARISON(EQ)** is specified.

TOTAL_ITEMS
Optional Parameter

The total number of items in the queue.

TOTAL_LENGTH
Optional Parameter

The sum of the lengths of all the items in the queue.

TRANSID
Optional Parameter

The identifier of the transaction that created the queue.

TSSH gate, INQUIRE_SYSID_TABLE_TOKEN function

Returns the SYSID_TABLE_TOKEN for the region.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
INSUFFICIENT_STORAGE
INVALID_LENGTH
INVALID_QUEUE_NAME
IO_ERROR
ITEM_NOT_FOUND
MAXIMUM_QUEUES_REACHED
POOL_NAME_NOT_FOUND
QUEUE_FULL
QUEUE_NOT_FOUND
SERVER_ERROR
SYSID_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SYSID_TABLE_TOKEN

The SYSID_TABLE_TOKEN.

TSSH gate, INQUIRE_TSPOOL function

Retrieve information about a shared temporary storage pool.

Input Parameters

POOL_NAME

The name of the shared temporary storage pool.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
TSPOOL_NOT_FOUND
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CONNECTED

Optional Parameter

A binary value indicating whether the pool is connected.

Values for the parameter are:

```
NO
YES
```

POOL_TOKEN

Optional Parameter

A token that identifies the temporary storage pool.

TSSH gate, READ_INTO function

This function reads the specified queue item into a buffer provided by the caller. The read cursor for the queue is set to the item number provided. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data is truncated (but an OK response is returned).

Input Parameters

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

ITEM_NUMBER
is the number of the item to be updated.

QUEUE_NAME
is the name of the queue being created or appended to.

POOL_TOKEN
Optional Parameter

is a token for the shared TS pool.

TRANSACTION_NUMBER
Optional Parameter

is the 4-byte transaction number (in packed-decimal format).

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_QUEUE_NAME
IO_ERROR
ITEM_NOT_FOUND
QUEUE_NOT_FOUND
SERVER_ERROR
```

FMH
indicates whether the data contains an FMH.

Values for the parameter are:

```
NO
YES
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TOTAL_ITEMS
is the total number of items in the queue on completion of the operation.

TSSH gate, READ_NEXT_INTO function

This function increments the read cursor by one and reads that item number into the buffer provided by the caller. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data will have been truncated.

Input Parameters

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

QUEUE_NAME
is the name of the queue being created or appended to.

POOL_TOKEN
Optional Parameter

is a token for the shared TS pool.

TRANSACTION_NUMBER
Optional Parameter

is the 4-byte transaction number (in packed-decimal format).

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
IO_ERROR
ITEM_NOT_FOUND
QUEUE_NOT_FOUND
SERVER_ERROR

FMH
indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

ITEM_NUMBER
returns the number of the item just read.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TOTAL_ITEMS
is the total number of items in the queue on completion of the operation.

TSSH gate, READ_NEXT_SET function

This function increments the queue's read cursor by one and reads that item number into a storage area obtained by TS.

Input Parameters

QUEUE_NAME
is the name of the queue being created or appended to.

POOL_TOKEN
Optional Parameter

is a token for the shared TS pool.

TRANSACTION_NUMBER
Optional Parameter

is the 4-byte transaction number (in packed-decimal format).

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_QUEUE_NAME
IO_ERROR
ITEM_NOT_FOUND
QUEUE_NOT_FOUND
SERVER_ERROR

FMH
indicates whether the data contains an FMH.

Values for the parameter are:
NO
YES

ITEM_DATA
returns the address and length of the item data.

ITEM_NUMBER
returns the number of the item just read.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TOTAL_ITEMS
is the total number of items in the queue on completion of the operation.

TSSH gate, READ_SET function

This function reads the specified queue item into a storage area obtained by TS. The read cursor for the queue is set to the input item number.

Input Parameters

ITEM_NUMBER
is the number of the item to be updated.

QUEUE_NAME
is the name of the queue being created or appended to.

POOL_TOKEN
Optional Parameter

is a token for the shared TS pool.

TRANSACTION_NUMBER
Optional Parameter

is the 4-byte transaction number (in packed-decimal format).

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_QUEUE_NAME
IO_ERROR
ITEM_NOT_FOUND
QUEUE_NOT_FOUND
SERVER_ERROR
```

FMH
indicates whether the data contains an FMH.

Values for the parameter are:

```
NO
YES
```

ITEM_DATA
returns the address and length of the item data.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TOTAL_ITEMS
is the total number of items in the queue on completion of the operation.

TSSH gate, REWRITE function

This function updates the specified item in an existing queue. The read cursor is unchanged.

Input Parameters

FMH
indicates whether the data contains an FMH.

Values for the parameter are:

```
NO
YES
```

ITEM_DATA
is the address and length of the item being written.
ITEM_NUMBER
is the number of the item to be updated.
QUEUE_NAME
is the name of the queue being created or appended to.
SUSPEND
indicates whether or not the request will be suspended if there is insufficient auxiliary storage to satisfy the request. This option is ignored if the queue is in main storage.

Values for the parameter are:
NO
YES
POOL_TOKEN
Optional Parameter

is a token for the shared TS pool.
TRANSACTION_NUMBER
Optional Parameter

is the 4-byte transaction number (in packed-decimal format).

Output Parameters
REASON
The following values are returned when RESPONSE is EXCEPTION:
INSUFFICIENT_STORAGE
INVALID_LENGTH
INVALID_QUEUE_NAME
IO_ERROR
ITEM_NOT_FOUND
QUEUE_NOT_FOUND
SERVER_ERROR
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
TOTAL_ITEMS
is the total number of items in the queue on completion of the operation.

TSSH gate, START_BROWSE function
Start browsing temporary storage queues.

Input Parameters
POOL_TOKEN
A token that identifies the temporary storage pool to be browsed.
QUEUE_NAME
Optional Parameter

The name of the temporary storage queue.

Output Parameters
REASON
The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
IO_ERROR
QUEUE_NOT_FOUND
SERVER_ERROR
TSPOOL_NOT_FOUND

BROWSE_TOKEN
See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSH gate, START_TSPOOL_BROWSE function

Start browsing the temporary storage pools.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
IO_ERROR
QUEUE_NOT_FOUND
SERVER_ERROR
TSPOOL_NOT_FOUND
```

BROWSE_TOKEN
See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSH gate, WRITE function

If the queue does not exist, this function creates a queue with the single item provided, and the queue's "read cursor" is set to zero.

Input Parameters

FMH
indicates whether the data contains an FMH.

Values for the parameter are:

```
NO
YES
```

ITEM_DATA
is the address and length of the item being written.
QUEUE_NAME
is the name of the queue being created or appended to.
SUSPEND
indicates whether or not the request will be suspended if there is insufficient auxiliary storage to satisfy the request. This option is ignored if the queue is in main storage.

Values for the parameter are:

```
NO
YES
```

EXPIRYINTMIN
Optional Parameter

is the expiry interval in minutes.
POOL_TOKEN
Optional Parameter

is a token for the shared TS pool.
TRANSACTION_NUMBER
Optional Parameter

is the 4-byte transaction number (in packed-decimal format).

TRANSID
Optional Parameter

is the id of the transaction which issued this request.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INSUFFICIENT_STORAGE
INVALID_LENGTH
INVALID_QUEUE_NAME
IO_ERROR
MAXIMUM_QUEUES_REACHED
QUEUE_FULL
SERVER_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TOTAL_ITEMS
is the total number of items in the queue on completion of the operation.

TSSR gate, SET_BUFFERS function

Sets the number of TS buffers to be used.

Input Parameters

BUFFERS
the number of buffers required.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSR gate, SET_START_TYPE function

Input Parameters

START_TYPE
The intended start type.

Values for the parameter are:

```
AUTO
COLD
```

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSSR gate, SET_STRINGS function

This function sets the number of strings to be used.

Input Parameters

STRINGS
the number of strings to be used.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Temporary storage domain's generic gates

Table 56 summarizes the generic gates of the domain. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 56. Temporary storage domain's generic gates

Gate	Trace	Functions	Format
DMDM	TS 0101 TS 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	TS 0501 TS 0502	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
APUE	TS 0601 TS 0602	SET_EXIT_STATUS	APUE

Temporary storage domain's callback formats

Table 57 describes the callback formats owned by the domain and shows the functions performed on the calls.

Table 57. Temporary storage domain's callback formats

Format	Calling module	Function
TSIC	DFHTSRM	DELIVER_IC_RECOVERY_DATA SOLICIT_INQUIRES

Note: In the descriptions of the formats, the input parameters are input not to the Temporary storage domain, but to the domain being called by the Temporary storage domain. Similarly, the output parameters are output by the domain that was called by the Temporary storage domain, in response to the call.

TSIC format, DELIVER_IC_RECOVERY_DATA function

The temporary storage domain uses this call-back format to deliver its recovery information for a temporary storage queue to the interval control component of the application domain.

Input Parameters

BMS
A binary value that indicates whether the queue was created by BMS.

Values for the parameter are:
NO
YES

IC A binary value that indicates whether the queue was created by interval control.

Values for the parameter are:
NO
YES

QUEUE_CREATION_TIME
The time (in store clock format) at which the queue was created.

QUEUE_NAME
The name of the queue.

RECOVERABLE
A binary value that indicates whether the queue is recoverable.

Values for the parameter are:
NO
YES

IC_DATA
Optional Parameter

The address and length of the interval control element (ICE) that is associated with the queue.

IN_DOUBT_OPERATION
Optional Parameter

The operation corresponding to the data being delivered.

Values for the parameter are:
GET_RELEASE
PUT
RELEASE

Output Parameters

DISCARD
A binary value that indicates whether the temporary storage domain should delete the queue.

Values for the parameter are:
NO
YES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TSIC format, SOLICIT_INQUIRES function

Temporary storage domain uses this call-back format to advise the interval control component in the AP domain that TS domain is ready to receive INQUIRE requests.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHTSBR	Handles the following requests: INQUIRE_QUEUE START_BROWSE GET_NEXT END_BROWSE CHECK_PREFIX
DFHTSCL	Automatically deletes temporary storage queues after their expiry interval

Module	Function
DFHTSDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHTSITR	Interprets TS domain trace entries
DFHTSP	Handles the following requests: PUT PUT_REPLACE GET GET_SET GET_RELEASE GET_RELEASE_SET RELEASE
DFHTSPT	Handles the following requests: PUT PUT_REPLACE GET GET_SET GET_RELEASE GET_RELEASE_SET RELEASE
DFHTSQR	Handles the following requests: WRITE REWRITE READ_INTO READ_SET READ_NEXT_INTO READ_NEXT_SET DELETE
DFHTSRM	Handles the following requests: PERFORM_PREPARE PERFORM_COMMIT PERFORM_SHUNT PERFORM_UNSHUNT START_BACKOUT END_BACKOUT START_DELIVERY DELIVER_RECOVERY END_DELIVERY TAKE_KEYPOINT
DFHTSSH	Handles the following requests: INITIALIZE INQUIRE_POOL_TOKEN INQUIRE_SYSID_TABLE_TOKEN WRITE REWRITE READ_INTO READ_NEXT_INTO READ_SET READ_NEXT_SET DELETE START_BROWSE GET_NEXT END_BROWSE INQUIRE_QUEUE

Module	Function
DFHTSSR	Handles the following requests: SET_START_TYPE SET_BUFFERS SET_STRINGS SET_EXIT_STATUS
DFHTSST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATISTICS
DFHTSSQ	Handles the suspension and resumption of tasks that attempt to write data to main temporary storage that would exceed TSMAINLIMIT

Exits

The temporary storage domain has four global user exit points: XTSQRIN, XTSQROUT, XTSPTIN and XTSPTOUT.

User domain (US)

The user domain manages CICS users and their security attributes.

User domain's specific gates

The specific gates provide access for other domains to functions that are provided by the US domain.

USAD gate, ADD_USER_WITH_PASSWORD function

The ADD_USER_WITH_PASSWORD function of the USAD gate is used to add a user to the CICS(R) region and verify the associated password or oidcard.

Input Parameters

PASSWORD

is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

SIGNON_TYPE

is the type of signon for the userid (specified by the USERID value).

Values for the parameter are:

```
ATTACH_SIGN_ON
DEFAULT_SIGN_ON
IRC_SIGN_ON
LU61_SIGN_ON
LU62_SIGN_ON
NON_TERMINAL_SIGN_ON
PRESET_SIGN_ON
USER_SIGN_ON
XRF_SIGN_ON
```

USERID

is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH

is the length of the USERID value.

APPLID

Optional Parameter

is the application identifier for the CICS region.

ENTRY_PORT_NAME
Optional Parameter

is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

ENTRY_PORT_TYPE
Optional Parameter

is the type of the optional entry port to be assigned to the userid (specified by the USERID value). This parameter is only valid if ENTRY_PORT_NAME is also specified.

Values for the parameter are:
```
TERMINAL
CONSOLE
```

GROUPID
Optional Parameter

is an identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

GROUPID_LENGTH
Optional Parameter

is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

NEW_PASSWORD
Optional Parameter

is a new password, 1 through 10 alphanumeric characters, to be assigned to the userid (specified by the USERID value). This parameter is only valid if PASSWORD is also specified.

OIDCARD
Optional Parameter

is an optional oidcard (operator identification card); a 65-byte field containing further security data from a magnetic strip reader (MSR) on 32xx devices.

PASSWORD_TYPE
Optional Parameter

specifies if the password is masked.

SCOPE_CHECK
Optional Parameter

indicates whether or not scope checking is to be performed for this function call.

Values for the parameter are:
```
NO
YES
```

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
DEL_TIMEOUT_ENTRY_FAILED
EXTRACT_FAILED
GETMAIN_FAILED
```

The following values are returned when RESPONSE is EXCEPTION:
```
ALREADY_SIGNED_ON
APPLICATION_NOTAUTH
ENQ_LIMIT_EXCEEDED
```

ENTRY_PORT_NOTAUTH
ESM_INACTIVE
ESM_TRANQUIL
GROUP_ACCESS_REVOKED
INQUIRE_PW_DATA_FAILED
INVALID_GROUPID
INVALID_NEW_PASSWORD
INVALID_OIDCARD
INVALID_PASSWORD
INVALID_USERID
NEW_PASSWORD_REQUIRED
OIDCARD_REQUIRED
PASSWORD_REQUIRED
SECLABEL_CHECK_FAILED
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE
USERID_NOT_IN_GROUP
USERID_REVOKED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETERS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USER_TOKEN
is the token identifying the userid in the user domain.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

USAD gate, ADD_USER_WITHOUT_PASSWORD function

The ADD_USER_WITHOUT_PASSWORD function of the USAD gate is used to add a user to the CICS region without verifying any password or oidcard.

Input Parameters

SIGNON_TYPE
is the type of signon for the userid (specified by the USERID value).

Values for the parameter are:

ATTACH_SIGN_ON
DEFAULT_SIGN_ON
IRC_SIGN_ON
LU61_SIGN_ON
LU62_SIGN_ON
NON_TERMINAL_SIGN_ON
PRESET_SIGN_ON
USER_SIGN_ON
XRF_SIGN_ON

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

APPLID
Optional Parameter

is the application identifier for the CICS region.

ENTRY_PORT_NAME
Optional Parameter

is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

ENTRY_PORT_TYPE
Optional Parameter

is the type of the optional entry port to be assigned to the userid (specified by the USERID value). This parameter is only valid if ENTRY_PORT_NAME is also specified.

Values for the parameter are:
```
CONSOLE
TERMINAL
```

GROUPID
Optional Parameter

is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

GROUPID_LENGTH
Optional Parameter

is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

SCOPE_CHECK
Optional Parameter

indicates whether or not scope checking is to be performed for this function call.

Values for the parameter are:
```
NO
YES
```

SUSPEND
Optional Parameter

indicates whether a wait during add user processing is acceptable.

Values for the parameter are:
```
NO
YES
```

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
DEL_TIMEOUT_ENTRY_FAILED
EXTRACT_FAILED
GETMAIN_FAILED
```

The following values are returned when RESPONSE is EXCEPTION:
```
ALREADY_SIGNED_ON
APPLICATION_NOTAUTH
ENQ_LIMIT_EXCEEDED
ENTRY_PORT_NOTAUTH
```

ESM_INACTIVE
ESM_TRANQUIL
GROUP_ACCESS_REVOKED
INVALID_GROUPID
INVALID_USERID
SECLABEL_CHECK_FAILED
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE
USER_NOT_LOCATED
USERID_NOT_IN_GROUP
USERID_REVOKED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETERS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USER_TOKEN
is the token identifying the userid in the user domain.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

USAD gate, DELETE_USER function

The DELETE_USER function of the USAD gate is used to delete the user from the CICS region.

Input Parameters

SIGNOFF_TYPE
is the type of signoff for the userid identified by the SECURITY_TOKEN value.

Values for the parameter are:
ATTACH_SIGN_OFF
DEFERRED_SIGN_OFF
DELETE_SIGN_OFF
LINK_SIGN_OFF
NON_TERMINAL_SIGN_OFF
PRESET_SIGN_OFF
TIMEOUT_SIGN_OFF
USER_SIGN_OFF
USRDELAY_SIGN_OFF
XRF_SIGN_OFF

USER_TOKEN
is the token identifying the userid in the user domain.

DELETE_IMMEDIATE
Optional Parameter

indicates whether the user should be deleted immediately.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ADD_TIMEOUT_ENTRY_FAILED
FREEMAIN_FAILED

The following values are returned when RESPONSE is EXCEPTION:

DEFAULT_USER_TOKEN
ESM_INACTIVE
ESM_TRANQUIL
INVALID_USER_TOKEN
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE

Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE

Optional Parameter

is the optional 32-bit SAF response code to the call.

USAD gate, INQUIRE_DEFAULT_USER function

The INQUIRE_DEFAULT_USER function of the USAD gate is used to inquire about the attributes of the default user (specified on the DFLTUSER system initialization parameter).

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
ADD_TIMEOUT_ENTRY_FAILED
DEL_EXPIRED_ENTRY_FAILED
DEL_TIMEOUT_ENTRY_FAILED
DIR_MANAGER_ADD_FAILED
DIR_MANAGER_DELETE_FAILED
DIR_MANAGER_LOCATE_FAILED
EXTRACT_FAILED
FREEMAIN_FAILED
GETMAIN_FAILED
INQUIRE_PW_DATA_FAILED
LOOP
SEC_DOMAIN_ADD_FAILED
SEC_DOMAIN_DELETE_FAILED

The following values are returned when RESPONSE is EXCEPTION:

ACCOUNT_INVALID
ALREADY_SIGNED_ON
APPLICATION_NOTAUTH
DEFAULT_USER_TOKEN
ENQ_LIMIT_EXCEEDED
ENTRY_PORT_NOTAUTH
ESM_INACTIVE
ESM_TRANQUIL
GROUP_ACCESS_REVOKED

INVALID_GROUPID
INVALID_NEW_PASSWORD
INVALID_OIDCARD
INVALID_PARAMETERS
INVALID_PASSWORD
INVALID_USER_TOKEN
INVALID_USERID
NEW_PASSWORD_REQUIRED
OIDCARD_REQUIRED
PASSWORD_REQUIRED
SECLABEL_CHECK_FAILED
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE
USER_NOT_LOCATED
USERID_NOT_DEFINED
USERID_NOT_DETERMINED
USERID_NOT_FOUND
USERID_NOT_IN_GROUP
USERID_REVOKED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACEE_PTR

Optional Parameter

is a pointer to the access control environment element, the control block that is generated by an external user (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER's ACEEis returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

CURRENT_GROUPID

Optional Parameter

is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

CURRENT_GROUPID_LENGTH

Optional Parameter

is the 8-bit length of the GROUPID value.

NATIONAL_LANGUAGE

Optional Parameter

is a three-character code identifying the national language for the userid. It can have any of the values in "National language codes (three-characters)" on page 1410.

OPERATOR_CLASSES

Optional Parameter

identifies the operator classes to which the user belongs. This is a 24-bit value, with each bit determining whether or not the user is a member of that class.

OPERATOR_IDENT

Optional Parameter

is the operator identification code, 1 through 3 alphanumeric characters, for the userid.

OPERATOR_PRIORITY
Optional Parameter

is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the userid.

TIMEOUT
Optional Parameter

is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal:

1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

USERID
Optional Parameter

is the identifier of the user (a userid of 1 through 10 alphanumeric characters).

USERID_LENGTH
Optional Parameter

is the length of the USERID value.

USERNAME
Optional Parameter

is an optional buffer into which the attributes of the user are placed.

XRF_REFLECTABLE
Optional Parameter

indicates whether or not you want CICS to sign off the userid following an XRF takeover.

Values for the parameter are:

```
NO
YES
```

USAD gate, INQUIRE_USER function

The INQUIRE_USER function of the USAD gate is used to inquire about the attributes of the user represented by the user token.

Input Parameters

USER_TOKEN
is the token identifying the userid in the user domain.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_USER_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACEE_PTR
Optional Parameter

is a pointer to the access control environment element, the control block that is generated by an external user (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER's ACEEis returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

CURRENT_GROUPID
Optional Parameter

is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

CURRENT_GROUPID_LENGTH
Optional Parameter

is the 8-bit length of the GROUPID value.

ENTRY_PORT_NAME
Optional Parameter

is the name of the entry port assigned to the userid.

ENTRY_PORT_TYPE
Optional Parameter

is the type of the entry port assigned to the userid. This parameter is only valid if ENTRY_PORT_NAME is also specified.

Values for the parameter are:
TERMINAL
CONSOLE

NATIONAL_LANGUAGE
Optional Parameter

is a three-character code identifying the national language for the userid. It can have any of the values in “National language codes (three-characters)” on page 1410.

OPERATOR_CLASSES
Optional Parameter

identifies the operator classes to which the user belongs. This is a 24-bit value, with each bit determining whether or not the user is a member of that class.

OPERATOR_IDENT
Optional Parameter

is the operator identification code, 1 through 3 alphanumeric characters, for the userid.

OPERATOR_PRIORITY
Optional Parameter

is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the userid.

TIMEOUT
Optional Parameter

is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.
1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

USERID
Optional Parameter

is the identifier of the user (a userid of 1 through 10 alphanumeric characters).

USERID_LENGTH
Optional Parameter

is the length of the USERID value.

USERNAME
Optional Parameter

is an optional buffer into which the attributes of the user are placed.

XRF_REFLECTABLE
Optional Parameter

indicates whether or not you want CICS to sign off the userid following an XRF takeover.

Values for the parameter are:
NO
YES

USAD gate, VALIDATE_USERID function

The VALIDATE_USERID function of the USAD gate is used to verify that the specified userid is a valid userid.

Input Parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
GROUP_ACCESS_REVOKED
SECURITY_INACTIVE
USERID_NOT_DEFINED
USERID_NOT_DETERMINED
USERID_REVOKED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USAD gate, NOTIFY_USERID function

The NOTIFY_USERID function of the USAD gate is used to record that a user ID should be removed when it is no longer in use or user details have changed.

Input Parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
GROUP_ACCESS_REVOKED
SECURITY_INACTIVE
USERID_NOT_DEFINED
USERID_NOT_DETERMINED
USERID_REVOKED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USAD gate, ADD_USER_VIA_ICRX function

The ADD_USER_VIA_ICRX function of the USAD gate provides the External Security Manager with an ICRX that can be mapped to a user.

Input Parameters

ICRX

Is the extended identity context reference (ICRX) of the user.

SUSPEND

Optional parameter.

Indicates whether a wait during add user processing is acceptable.

Values for the parameter are:

NO
YES

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

DEL_TIMEOUT_ENTRY_FAILED
EXTRACT_FAILED
GETMAIN_FAILED

The following values are returned when RESPONSE is EXCEPTION:

ALREADY_SIGNED_ON
APPLICATION_NOTAUTH
ENQ_LIMIT_EXCEEDED
ENTRY_PORT_NOTAUTH
ESM_INACTIVE
ESM_TRANQUIL
GROUP_ACCESS_REVOKED
INVALID_GROUPID
INVALID_USERID
SECLABEL_CHECK_FAILED
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE
USER_NOT_LOCATED
USERID_NOT_IN_GROUP
USERID_REVOKED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETERS

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SAF_RESPONSE

Optional parameter.

Is the optional 32-bit SAF response code to the call.

ESM_RESPONSE

Optional parameter.

Is the optional 32-bit ESM response code to the call.

USERID_LENGTH

Optional parameter.

Is the length of the user ID value.

USERID

Optional parameter.

Is the identifier of the user (a 1- to 10-character alphanumeric user ID) added to the security domain.

USAD gate, INQUIRE_ICRX function

The INQUIRE_ICRX function of the USAD gate obtains ICRX data from the External Security Manager.

Input Parameters

USER_TOKEN

Optional parameter.

Is the token identifying the user ID in the user domain.

OUT_ICRX

Optional parameter.

Is the ICRX representing the user ID.

ICRX_TYPE

Optional parameter.

Is the type of ICRX being requested.

Values for the parameter are:

COMPLETE: returns a copy of the RACF-generated ICRX.
PSEUDO: returns a pseudo ICRX.

RETRY

Optional parameter.

Used if the buffer size specified for the ICRX in a prior call was insufficient.

RETRY(YES) returns a previously created ICRX. RETRY(NO) is the default.

DNAME

Optional parameter.

The distinguished name associated with the ICRX-defined user ID.

REALM

Optional parameter.

Is the realm associated with the ICRX-defined user ID.

IN_RETRY_TOKEN

Optional parameter.

Used if the buffer size specified for the ICRX in a prior call was insufficient.

This parameter must be set to the value returned in the previous OUT_RETRY_TOKEN.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

ICRX_NOT_AVAILABLE
INVALID_USER_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

OUT_RETRY_TOKEN

Optional parameter.

Used if the buffer size specified for the ICRX in a prior call was insufficient.

If the buffer size was insufficient and OUT_RETRY_YES is specified, this parameter provides state data for the next call.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ICRX
Is the extended identity context reference (ICRX) of the user.

USAD gate, RELEASE_ICRX function

The RELEASE_ICRX function of the USAD gate tells the External Security Manager to release the ICRX storage from cache.

Input Parameters

USER_TOKEN
Optional parameter.

Is the token identifying the ICRX in the user domain.

STORAGE_TYPE
Indicates the ICRX storage location.

The values are:
`BUFFER`: for internal CICS buffer
`CACHE`: for the RACF cache
`BOTH`: to release both `BUFFER` and `CACHE`

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`FUNCTION_NOT_SUPPORTED`
`ICRX_INVALID`
`INVALID_USER_TOKEN`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USAD gate, ICRX_TO_USERID function

The ICRX_TO_USERID function of the USAD gate maps an ICRX to a user ID.

Input Parameters

ICRX
Is the extended identity context reference (ICRX) of the user.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
`USERID_NOT_DETERMINED`
`ICRX_INVALID`
`ICRX_NOT_AVAILABLE`
`SECURITY_INACTIVE`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USERID_LENGTH
Is the length of the USERID value.

USERID
Is the user ID that is mapped to the ICRX.

USAD gate, GET_ASSOCIATED_DATA_LIST function

The GET_ASSOCIATED_DATA_LIST function of the USAD gate obtains a list of tasks that match the supplied filters.

Input Parameters

TASK_LIST
Is the identifier of the list of tasks.

INPUT_LIST_SIZE
Optional parameter.

Is the length of the input list.

DNAME
Optional parameter.

Is the distinguished name.

REALM
Optional parameter.

Is the realm associated with the distinguished name.

MERGE
Optional parameter.

Indicates whether to merge. Has the following values:
YES: an input list exists which is used for filtering.
NO: the task list for the entire CICS region is used as the filter list.

MERGED_TASK_LIST
Optional parameter.

Is the output array.

Output Parameters

OUTPUT_LIST_SIZE
Is the length of the output list.

REASON
The following value is returned when RESPONSE is DISASTER:
ABEND

The following value is returned when RESPONSE is EXCEPTION:
INVALID_DNAME_FILTER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USFL gate, FLATTEN_USER function

The FLATTEN_USER function of the USFL gate is used to flatten the user's security state and place into the FLATTENED_USER buffer provided.

Input Parameters

FLATTENED_USER
is the buffer into which the flattened security state is placed.

USER_TOKEN
is the token identifying the userid in the user domain.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
DIR_MANAGER_LOCATE_FAILED
LOOP

SEC_DOM_FLATTEN_FAILED

The following values are returned when RESPONSE is EXCEPTION:

ESM_INACTIVE
ESM_TRANQUIL
INVALID_USER_TOKEN
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE

The following values are returned when RESPONSE is INVALID:

INVALID_FLATTENED_BUFFER
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE

Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE

Optional Parameter

is the optional 32-bit SAF response code to the call.

USFL gate, TAKEOVER function

The TAKEOVER function of the USFL gate is used, when an XRF takeover occurs, to obtain the SNSCOPE ENQ resources for those users who could not obtain it during tracking, because the resources were already held by the active region.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USFL gate, UNFLATTEN_USER function

The UNFLATTEN_USER function of the USFL gate is used to unflatten the user security state data in the FLATTENED_USER buffer, and add the userid to the user domain.

Input Parameters

FLATTENED_USER

is the buffer into which the flattened security state is placed.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
DEL_TIMEOUT_ENTRY_FAILED
DIR_MANAGER_ADD_FAILED
DIR_MANAGER_DELETE_FAILED

FREEMAIN_FAILED
GETMAIN_FAILED
LOOP
SEC_DOM_UNFLATTEN_FAILED
SEC_DOMAIN_DELETE_FAILED

The following values are returned when RESPONSE is EXCEPTION:

ALREADY_SIGNED_ON
APPLICATION_NOTAUTH
ENTRY_PORT_NOTAUTH
ESM_INACTIVE
ESM_TRANQUIL
GROUP_ACCESS_REVOKED
SECLABEL_CHECK_FAILED
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE
USERID_NOT_IN_GROUP
USERID_REVOKED
USERID_UNDEFINED

The following values are returned when RESPONSE is INVALID:

INVALID_FLATTENED_BUFFER
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USER_TOKEN
is the token identifying the userid in the user domain.

USIS gate, SET_USER_DOMAIN_PARMS function

At CICS startup, loads information for the user domain from the system initialization table (SIT) into the user state data.

Input Parameters

APPLID
is the application identifier for the CICS region.

DEFAULT_USERID
is the default userid, as 1 through 10 alphanumeric characters.

DIRECTORY_TIMEOUT_VALUE
is the intersystem refresh delay, in the range 0 through 10080 minutes (up to 7 days), for the default userid.

SIGNON_SCOPE
is the scope for which the default userid can be signed on.

Values for the parameter are:

CICS
MVSIMAGE
NONE
SYSPLEX

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USIS gate, INQUIRE_DOMAIN function

Allows other domains to inquire on the support provided by the user domain.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ICRX_SUPPORTED (YES|NO)
Indicates whether an ICRX (Extended Identity Context Reference) is supported.

USXM gate, ADD_TRANSACTION_USER function

The ADD_TRANSACTION_USER function of the USXM gate sets the user characteristics (as security tokens) for a transaction.

Input Parameters

EDF_USER_TOKEN
Optional Parameter

is the optional EDF user token representing the characteristics of the EDF user of the transaction.

PRINCIPAL_USER_TOKEN
Optional Parameter

is the optional principal user token representing the characteristics of the principal user of the transaction.

SESSION_USER_TOKEN
Optional Parameter

is the optional session user token representing the characteristics of the session user of the transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
ALREADY_SIGNED_ON
DUPLICATE_USER
INVALID_USER_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USXM gate, DELETE_TRANSACTION_USER function

The DELETE_TRANSACTION_USER function of the USXM gate deletes the user token of the specified token type for the transaction.

Input Parameters

TOKEN_TYPE

is the type of user token for the transaction.

Values for the parameter are:

EDF
PRINCIPAL
SESSION

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

NO_USER_TOKEN

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USXM gate, END_TRANSACTION function

The END_TRANSACTION function of the USXM gate deletes all the user token to security token maps for the transaction.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
FREEMAIN_FAILED
LOOP

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USXM gate, FLATTEN_TRANSACTION_USER function

The FLATTEN_TRANSACTION_USER function of the USXM gate creates the contents of a FLAT_TRANSUSER buffer from the principal user of the current transaction.

Input Parameters

FLAT_TRANSUSER

is the buffer to be created.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FLAT_TRANSUSER
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USXM gate, INIT_TRANSACTION_USER function

The INIT_TRANSACTION_USER function of the USXM gate initializes the transaction for the user characteristics identified by the PRINCIPAL_USER_TOKEN value.

Input Parameters

PRINCIPAL_USER_TOKEN

is the optional principal user token representing the characteristics of the principal user of the transaction.

EDF_USER_TOKEN

Optional Parameter

is the optional EDF user token representing the characteristics of the EDF user of the transaction.

SESSION_USER_TOKEN

Optional Parameter

is the optional session user token representing the characteristics of the session user of the transaction.

XMAT_CALL

Optional Parameter

indicates whether the function is called while a transaction is being attached.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
GETMAIN_FAILED
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_USER_TOKEN
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

PRIORITY

is the priority value, in the range 0 through 255 (where 255 is the highest priority), for the user with the token identified by the PRINCIPAL_USER_TOKEN value.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USDOM_TRANSACTION_TOKEN
is the user token to be used for reference to user characteristics only. It is treated as the principal user token until the next ADD_TRANSACTION_USER call for the transaction.

USXM gate, INQUIRE_TRANSACTION_USER function

The INQUIRE_TRANSACTION_USER function of the USXM gate inquires about the user characteristics associated with the transaction identified by the USDOM_TRANSACTION_TOKEN value.

Input Parameters

USDOM_TRANSACTION_TOKEN
Optional Parameter

is the user token to be used for reference to user characteristics only.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACEE_PTR
Optional Parameter

is a pointer to the access control environment element, the control block that is generated by an external user (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER's ACEEis returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

APPLID
Optional Parameter

is the application identifier for the CICS region.

CURRENT_GROUPID
Optional Parameter

is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

CURRENT_GROUPID_LENGTH
Optional Parameter

is the 8-bit length of the CURRENT_GROUPID value.

ENTRY_PORT_NAME
Optional Parameter

is the name of the entry port assigned to the userid.

ENTRY_PORT_TYPE
Optional Parameter

is the type of the entry port assigned to the userid. This parameter is only valid if ENTRY_PORT_NAME is also specified.

Values for the parameter are:
```
TERMINAL
```

CONSOLE

GROUPID_LENGTH
Optional Parameter

The length of the name of the RACF group to which the user was assigned at signon.

NATIONAL_LANGUAGE
Optional Parameter

is a three-character code identifying the national language for the userid. It can have any of the values in “Languages and their codes” on page 694.

OPERATOR_CLASSES
Optional Parameter

identifies the operator classes to which the user belongs. This is a 24-bit value, with each bit determining whether or not the user is a member of that class.

OPERATOR_IDENT
Optional Parameter

is the operator identification code, 1 through 3 alphanumeric characters, for the userid.

OPERATOR_PRIORITY
Optional Parameter

is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the userid.

PRINCIPAL_USER_TOKEN
Optional Parameter

is the token identifying the userid in the user domain.

TIMEOUT
Optional Parameter

is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.

1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

USERID
Optional Parameter

is the identifier of the user (a userid of 1 through 10 alphanumeric characters).

USERID_LENGTH
Optional Parameter

is the length of the USERID value.

USERNAME
Optional Parameter

is an optional buffer into which the attributes of the user are placed.

XRFSOFF
Optional Parameter

indicates whether or not you want CICS to sign off the user following an XRF takeover.

Values for the parameter are:

FORCE
NOFORCE

USXM gate, TERM_TRANSACTION_USER function

The TERM_TRANSACTION_USER function of the USXM gate removes the state information created by an INIT_TRANSACTION_USER function.

Input Parameters

USDOM_TRANSACTION_TOKEN
is the user token to be used for reference to user characteristics only.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
FREEMAIN_FAILED
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USXM gate, UNFLATTEN_TRANSACTION_USER function

The UNFLATTEN_TRANSACTION_USER function of the USXM gate adds (by the ADD_USER_WITHOUT_PASSWORD function of the USAD gate) the user defined by the contents of the supplied FLAT_TRANSUSER buffer.

Input Parameters

FLAT_TRANSUSER
is the buffer to be created.

SUSPEND
Optional Parameter

indicates whether a wait during add user processing is acceptable.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
APPLICATION_NOTAUTH
ENTRY_PORT_NOTAUTH
ESM_INACTIVE
ESM_TRANQUIL
GROUP_ACCESS_REVOKED
INVALID_GROUPID
INVALID_USERID
SECLABEL_CHECK_FAILED
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE
USER_NOT_LOCATED
USERID_NOT_IN_GROUP
USERID_REVOKED
```

PRINCIPAL_USER_TOKEN
is the token identifying the userid in the user domain.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

User domain's generic gates

Table 58 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 58. User domain's generic gates

Gate	Trace	Functions	Format
DMDM	US 0101 US 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
STST	US 0601 US 0602	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST

In initialization processing, performs internal routines to set up the user domain, and gets the initial user options, as for the"USIS gate, SET_USER_DOMAIN_PARMS function" on page 1244.

For a cold start, the user options come from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

User domain also issues console messages during initialization to report whether or not security is active.

In quiesce and termination processing, the user domain performs only internal routines.

Modules

Module	Function
DFHUSAD	Handles the following requests: ADD_USER_WITH_PASSWORD ADD_USER_WITHOUT_PASSWORD DELETE_USER INQUIRE_USER INQUIRE_DEFAULT_USER VALIDATE_USERID NOTIFY_USERID ADD_USER_VIA_ICRX INQUIRE_ICRX RELEASE_ICRX ICRX_TO_USERID GET_ASSOCIATED_DATA_LIST

Module	Function
DFHUSDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHUSDUF	US domain offline dump formatting routine
DFHUSFL	Handles the following requests: FLATTEN_USER UNFLATTEN_USER TAKEOVER
DFHUSIS	Handles the following requests: SET_USER_DOMAIN_PARMS INQUIRE_DOMAIN
DFHUSST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHUSTI	Handles user timeout processing
DFHUSTRI	Interprets US domain trace entries
DFHUSXM	Handles the following requests: ADD_TRANSACTION_USER DELETE_TRANSACTION_USER END_TRANSACTION INIT_TRANSACTION_USER INQUIRE_TRANSACTION_USER FLATTEN_TRANSACTION_USER UNFLATTEN_TRANSACTION_USER

Web domain (WB)

The web domain manages interaction between CICS and web clients, or between CICS as an HTTP client and servers on the Internet, with the exception of Atom feeds, which are managed by the Web 2.0 (W2) domain.

For more information about CICS as an HTTP server and CICS as an HTTP client, see Configuring CICS web support components.

Web domain's specific gates

The specific gates provide access for other domains to functions that are provided by the WB domain.

WBAP gate, END_BROWSE function

The END_BROWSE function defines the end of a browse of the HTTP headers received for an HTTP request.

Input Parameters

DATA_TYPE

Indicates whether the request is a browse operation on HTTP forms data or HTTP headers.

Values for the parameter are:

`FORMFIELD`
`HEADER`

Output Parameters

REASON

The values for the parameter are:

```
FORMFLD_BROWSE_NOT_ACTIVE
HEADER_BROWSE_NOT_ACTIVE
NON_WEB_TRANSACTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBAP gate, GET_HTTP_RESPONSE function

The GET_HTTP_RESPONSE function retrieves the HTTP Response which has been constructed by a Web API application program.

Output Parameters

REASON

The values for the parameter are:

```
NO_PREVIOUS_WEB_SEND
NON_WEB_TRANSACTION
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DOCUMENT_TOKEN

Optional Parameter

A token that identifies the copy of the document stored on the last EXEC CICS WEB SEND command.

WBAP gate, GET_MESSAGE_BODY function

The GET_MESSAGE_BODY function retrieves the previously constructed body of an HTTP response.

Input Parameters

CLIENT_CODEPAGE

Optional Parameter

ASCII Code page into which the data is to be converted before being passed back to the caller

CONTAINER_NAME

Optional Parameter

The name of the container that will receive the message body.

CONTAINER_POOL

Optional Parameter

The container pool of which the named container is a member.

CONVERT

Optional Parameter

indicates whether or not data is to undergo code page conversion.

Values for the parameter are:

```
DEFAULT
NO
YES
```

DATA_BUFFER

Optional Parameter

The buffer into which the data is to be placed.

TRUNCATE
Optional Parameter

A binary value that specifies how data that is not returned on the first call is handled. **TRUNCATE(NO)** specifies that the rest of the data will be returned on subsequent calls. **TRUNCATE(YES)** specifies that the extra data will be truncated and will not be returned.

Values for the parameter are:
NO
YES

SERVER_CODEPAGE
Optional Parameter

EBCDIC Code page of the data to be passed back

Output Parameters

REASON
The values for the parameter are:
BODY_INCOMPLETE
BODY_TRUNCATED
BODY_TRUNCATED
CHUNK_INCOMPLETE
CLOSESTATUS_INVAL_NONHTTP
INVALID_CLIENT_CODEPAGE
INVALID_CLIENT_CODEPAGE
INVALID_CODEPAGE_COMBIN
INVALID_MEDIATYPE
INVALID_SERVER_CODEPAGE
INVALID_SERVER_CODEPAGE
NON_WEB_TRANSACTION
PARTIAL_BODY

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REQUEST_TYPE
Optional Parameter

Indicates whether we are processing an HTTP Request.

Values for the parameter are:
HTTP
NON_HTTP

SET_BLOCK
Optional Parameter

Address of a block of storage containing the message body

WBAP gate, INITIALIZE_TRANSACTION function

The INITIALIZE_TRANSACTION function is used to initialize a transaction whose primary client is a WRB but whose code is not part of the WB component (such as the Pipeline Manager). It verifies the Web environment and returns useful Web state data.

Input Parameters

CLIENT_CODEPAGE
Optional Parameter

The code page used by the client.

MEDIATYPE
Optional Parameter

The internet media type specified on the request.

URI
Optional Parameter

The URI specified in the request.

Output Parameters

REASON
The values for the parameter are:

```
INITIALIZATION_FAULT
NON_WEB_TRANSACTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

PIPELINE
Optional Parameter

The PIPELINE resource associated with the inbound request.

TCPIPSERVICE
Optional Parameter

The TCPIPSERVICE resource associated with the inbound request.

WEBSERVICE
Optional Parameter

The WEBSERVICE resource associated with the inbound request.

WBAP gate, INQUIRE function

The INQUIRE function passes back information pertaining to an HTTP request.

Input Parameters

CLIENT_NAME
Optional Parameter

Buffer to contain TCP/IP name of client from which HTTP request was received.

HOST_BUFFER
Optional Parameter

HTTP_METHOD
Optional Parameter

Buffer to contain HTTP method specified on the HTTP request

HTTP_VERSION
Optional Parameter

Buffer to contain HTTP version specified on the HTTP request

QUERYSTRING
Optional Parameter

Buffer to contain HTTP query string specified on the HTTP request

SERVER_NAME
Optional Parameter

Buffer to contain TCP/IP name of CICS

URI
Optional Parameter

Buffer to contain URI specified on the HTTP request

Output Parameters

REASON
The values for the parameter are:
INVALID_REQUEST_FORMAT
NON_WEB_TRANSACTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CERTIFICATE_TOKEN
Optional Parameter

eight byte token identifying SSL certificate of client issuing this HTTP request

CLIENT_ADDR
Optional Parameter

Fullword containing IP address of the client from which the HTTP request was received

REQUEST_TYPE
Optional Parameter

Indicates whether we are processing an HTTP Request.

Values for the parameter are:
HTTP
NON_HTTP

SCHEME
Optional Parameter

Values for the parameter are:
HTTP
HTTPS

SERVER_ADDR
Optional Parameter

Fullword containing IP address of the TCP/IP stack on which the HTTP request was received

SERVER_PORT
Optional Parameter

Fullword containing port number on which the HTTP request was received

SSL_TYPE
Optional Parameter

Indicates what level of SSL support applies to the incoming HTTP request.

Values for the parameter are:
CLIENTAUTH
NO
YES

URIMAP
Optional Parameter

WBAP gate, QUERY_URIMAP function

The QUERY_URIMAP function passes back URIMAP information that applies to an HTTP request.

Output Parameters

URIMAP
URIMAP name.

REASON
The value for the parameter is returned when RESPONSE is EXCEPTION:

NON_WEB_TRANSACTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBAP gate, READ function

Retrieve either a specific HTTP header value from the TS queue containing the HTTP request header data or a specific form field from the fields in the HTML form for the current HTTP request.

Input Parameters

DATA_TYPE
Indicates whether the request is a browse operation on HTTP forms data or HTTP headers.

Values for the parameter are:
FORMFIELD
HEADER

HTTP_BUFFER_NAME
Optional Parameter

A block containing a character string that contains the name of the header or form field, and the length of that string.

HTTP_BUFFER_VALUE
The value of the header or form field.

CLIENT_CODEPAGE
Optional Parameter

ASCII code page into which the data is to be converted before being passed back to the caller

CONVERT
Optional Parameter

Indicates whether or not data is to undergo code page conversion.

Values for the parameter are:
DEFAULT
NO
YES

PRIVATE_DATA
Optional Parameter

A binary value indicating whether the data is private. Private data is not exposed in trace entries.

Values for the parameter are:
NO
YES

SERVER_CODEPAGE
Optional Parameter

EBCDIC code page of the data to be passed back

Output Parameters

REASON
The values for the parameter are:
CLIENT_CODEPAGE_UNSUPP
CODEPAGE_NOT_FOUND
FORMFIELD_CANNOT_GET_BODY
FORMFIELD_CORRUPT_HEADER

FORMFIELD_NO_BOUNDARY_STR
FORMFIELD_NO_CONTENT_HDR
FORMFIELD_STRUCT_CORRUPT
FORMFIELD_STRUCT_FORM_ERR
FORMFIELD_UNKNOWN_FORMTYPE
FORMFLD_NOT_FOUND
FORMFLD_VALUE_LENGTH_ERROR
HEADER_NOT_FOUND
INVALID_CODEPAGE_COMBIN
INVALID_REQUEST_FORMAT
NO_CONVERT_PARM
NO_FORMS_DATA
NON_WEB_TRANSACTION
SERVER_CODEPAGE_UNSUPP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SET_BLOCK
Optional Parameter

A block for returning the pointer and data length when the SET option is specified.

WBAP gate, READ_NEXT function

The READ_NEXT function returns the next HTTP header in a browse of HTTP headers.

Input Parameters

DATA_TYPE
Indicates whether the request is a browse operation on HTTP forms data or HTTP headers.

Values for the parameter are:
FORMFIELD
HEADER

HTTP_BUFFER_NAME
Optional Parameter

A block containing a character string that contains the name of the header or form field, and the length of that string.

HTTP_BUFFER_VALUE
The value of the header or form field.

Output Parameters

REASON
The values for the parameter are:
BROWSE_END
FORMFIELD_CORRUPT_HEADER
FORMFIELD_STRUCT_CORRUPT
FORMFLD_BROWSE_NOT_ACTIVE
FORMFLD_NAME_LENGTH_ERROR
FORMFLD_VALUE_LENGTH_ERROR
HEADER_BROWSE_NOT_ACTIVE
HEADER_NAME_LENGTH_ERROR
HEADER_VALUE_LENGTH_ERROR
INVALID_FORMFLD
INVALID_HEADER
NO_CONVERT_PARM

NO_FORMS_DATA
NON_WEB_TRANSACTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBAP gate, SEND_RESPONSE function

The SEND_RESPONSE function identifies a CICS Document which is to be used as the body of a HTTP response, and the HTTP reason code with which that response is to be returned.

Input Parameters

ACTION
Optional Parameter

Values for the parameter are:
EVENTUAL
IMMEDIATE

CHUNKING
Optional Parameter

Specifies whether the data is to be chunked.

Values for the parameter are:
YES
NO
END

CLIENT_CODEPAGE
Optional Parameter

ASCII Code page into which the data is to be converted before being passed back to the caller

CLOSESTATUS
Optional Parameter

Controls sending of the `connect: close` header. If the session is not to persist then send the header. The default action is to not to send the `connect: close` header unless the client has indicated that it wants to close the connection after the response has been received.

Values for the parameter are:
CLOSE
NOCLOSE

CONVERSION
Optional Parameter

A binary parameter indicating whether the data is to undergo code page conversion.

Values for the parameter are:
NO
YES

DOCUMENT_TOKEN
Optional Parameter

The 8 byte field into which CICS places the document token identifying the document which contains the body of the HTTP response.

FROM
Optional Parameter

The block containing the data to be sent.

MEDIATYPE
Optional Parameter

SERVER_CODEPAGE
Optional Parameter

EBCDIC Code page of the data to be passed back

STATUS_CODE
Optional Parameter

HTTP response code with which the HTTP response is returned

STATUS_TEXT
Optional Parameter

Text to accompany HTTP response code with which the HTTP response is returned.

Output Parameters

REASON
The values for the parameter are:

```
CHUNK_INCOMPLETE
CHUNKING_NOT_SUPPORTED
CHUNKLENGTH_INVAL_HTTP10
CHUNKLENGTH_INVAL_NONHTTP
CLOSESTATUS_INVAL_NONHTTP
CONNECTION_CLOSED
DOCUMENT_NOT_FOUND
HEADER_MISSED_THE_BUS
INVALID_CHUNKSIZE
INVALID_CODEPAGE_COMBIN
INVALID_MEDIATYPE
INVALID_SEND_SEQUENCE
MSG_BODY_NOT_ALLOWED
NON_WEB_TRANSACTION
PREVIOUS_SEND_FAILED
SOCKETS_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBAP gate, START_BROWSE function

The START_BROWSE function starts a browse of the HTTP headers or the HTML forms data in an HTTP request.

Input Parameters

DATA_TYPE
Indicates whether the request is a browse operation on HTTP forms data or HTTP headers.

Values for the parameter are:

```
FORMFIELD
HEADER
```

CLIENT_CODEPAGE
Optional Parameter

ASCII code page into which the data is to be converted before being passed back to the caller

CONVERT
Optional Parameter

Indicates whether or not data is to undergo code page conversion.

Values for the parameter are:

DEFAULT
NO
YES

HTTP_BUFFER_NAME

Optional Parameter

A block containing a character string that contains the name of the header or form field, and the length of that string.

SERVER_CODEPAGE

Optional Parameter

EBCDIC code page of the data to be passed back

Output Parameters

REASON

The values for the parameter are:

CLIENT_CODEPAGE_UNSUPP
FORMFIELD_CANNOT_GET_BODY
FORMFIELD_CORRUPT_HEADER
FORMFIELD_NO_BOUNDARY_STR
FORMFIELD_NO_CONTENT_HDR
FORMFIELD_STRUCT_CORRUPT
FORMFIELD_STRUCT_FORM_ERR
FORMFIELD_UNKNOWN_FORMTYPE
FORMFLD_BROWSE_ACTIVE
FORMFLD_NAME_LENGTH_ERROR
HEADER_BROWSE_ACTIVE
INVALID_CODEPAGE_COMBIN
INVALID_FORMFLD
INVALID_REQUEST_FORMAT
NO_CONVERT_PARM
NO_FORMS_DATA
NON_WEB_TRANSACTION
SERVER_CODEPAGE_UNSUPP

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBAP gate, WRITE_HEADER function

The WRITE_HEADER function causes a HTTP response header to be stored by CICS.

Input Parameters

HTTP_BUFFER_NAME

Optional Parameter

A block containing a character string that contains the name of the header or form field, and the length of that string.

HTTP_BUFFER_VALUE

The value of the header.

Output Parameters

REASON

The values for the parameter are:

INVALID_TRAILER_HEADER

NON_WEB_TRANSACTION
TRAILER_NOT_SUPPORTED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBCL gate, CLOSE_SESSION function

The CLOSE_SESSION function ends the connection to the server by closing the socket and releasing the session control block.

Input Parameters

SESSION_TOKEN

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED
BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED
CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR
EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR
HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN
INVALID_URL
MEDIATYPE_INVALID
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS

NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR
URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is DISASTER or EXCEPTION
UNEXPECTED_SOCK_RESPONSE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
OMITTED_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBCL gate, END_BROWSE_HEADERS function

The END_BROWSE_HEADERS function ends a browse of the HTTP headers for an HTTP response that has been received.

Input Parameters

SESSION_TOKEN

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED
BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED
CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR

EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR
HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN
INVALID_URL
MEDIATYPE_INVALID
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS
NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR
URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
OMITTED_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBCL gate, INQUIRE_SESSION function

The INQUIRE_SESSION function returns information about the specified connection to a server, represented by the session token.

Input Parameters

SESSION_TOKEN

HOST_BUFFER
Optional Parameter

PATH_BUFFER
Optional Parameter

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED
BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED
CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR
EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR
HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN
INVALID_URL
MEDIATYPE_INVALID
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS
NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR

URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
OMITTED_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

HTTP_RNUM
Optional Parameter

HTTP_VNUM
Optional Parameter

PORT
Optional Parameter

SCHEME
Optional Parameter

Values for the parameter are:
HTTP
HTTPS
OTHER

BIN_IP_ADDRESS
Binary format IP address

IP_ADDRESS
Character format IP address

IP_ADDRESS_LEN
Length of IP_ADDRESS

IP_ADDRESS_TYPE
Values for the parameter are:
IPV4_HOST
IPV6_HOST

HOSTTYPE
Values for the parameter are:
IPV4_HOST
IPV6_HOST

CHUNKED_REQUEST
Values for the parameter are:
YES
NO

URIMAP
Optional Parameter

WBCL gate, OPEN_SESSION function

The OPEN_SESSION function opens a session with the HTTP server.

Input Parameters

HOST

PORT

SCHEME

Values for the parameter are:
HTTP
HTTPS
OTHER

CERTIFICATE_LABEL
Optional Parameter

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the CIPHER_SUITES parameter.

CIPHER_SUITES
Optional Parameter

A string of up to 56 hexadecimal digits that encodes a list of up to 28 2-digit cipher suite codes.

HOST_CODEPAGE
Optional Parameter

PROXY_URL
Optional Parameter

URIMAP
Optional Parameter

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED
BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED
CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR
EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR
HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN
INVALID_URL
```

MEDIATYPE_INVALID
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS
NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR
URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is DISASTER or EXCEPTION:

UNEXPECTED_DSAT_RESPONSE
UNEXPECTED_SOCK_RESPONSE

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
OMITTED_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SESSION_TOKEN

HTTP_RNUM
Optional Parameter

HTTP_VNUM
Optional Parameter

WBCL gate, PARSE_URL function

The PARSE_URL function parses a URL into its constituent components.

Input Parameters

URL

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED

BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED
CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR
EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR
HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN
INVALID_URL
MEDIATYPE_INVALID
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS
NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR
URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
OMITTED_PARAMETER

HOST

Optional Parameter

HOSTTYPE

Values for the parameter are:

IPV4_HOST

IPV6_HOST

PATH
Optional Parameter

PORT
Optional Parameter

QUERY_STRING
Optional Parameter

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SCHEME

Values for the parameter are:
HTTP
HTTPS
OTHER

BIN_IP_ADDRESS
Binary IP address

SCHEME_NAME
Optional Parameter

WBCL gate, READ_HEADER function

The READ_HEADER function reads a specific HTTP header from the HTTP response that has been received.

Input Parameters

NAME

SESSION_TOKEN

VALUE_BUFFER

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:
ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED
BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED
CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR
EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR

HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN
INVALID_URL
MEDIATYPE_INVALID
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS
NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR
URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
OMITTED_PARAMETER

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBCL gate, READ_NEXT_HEADER function

The READ_NEXT_HEADER function reads the next HTTP header in the browse operation for an HTTP response that has been received.

Input Parameters

NAME_BUFFER
SESSION_TOKEN
VALUE_BUFFER

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED

LOOP

The following values are returned when RESPONSE is EXCEPTION:

BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED
BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED
CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR
EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR
HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN
INVALID_URL
MEDIATYPE_INVALID
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS
NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR
URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
OMITTED_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBCL gate, READ_RESPONSE function

The READ_RESPONSE function waits for and then reads the HTTP response that is expected from the HTTP server.

Input Parameters

SESSION_TOKEN
An 8-byte binary value that uniquely identifies the connection between CICS as an HTTP client, and an HTTP server.

BODY
Optional Parameter

A buffer that will receive the HTTP response.

CONTAINER_NAME
Optional Parameter

The name of the container that will receive the message body.

CONTAINER_POOL
Optional Parameter

The container pool of which the named container is a member.

HOST_CODEPAGE
Optional Parameter

MAX_DATA_LENGTH
Optional Parameter

STATUS_TEXT
Optional Parameter

Text to accompany HTTP response code with which the HTTP response is returned.

TIME_OUT_VALUE
Optional Parameter

TRANSLATE
Optional Parameter

Values for the parameter are:
NO
YES

TRUNCATE
Optional Parameter

indicates whether or not data is to be truncated if the buffer is too small.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED
BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED
CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR
EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR
HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN
INVALID_URL
MEDIATYPE_INVALID
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS
NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR
URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is DISASTER or EXCEPTION:
UNEXPECTED_SOCK_RESPONSE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

OMITTED_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATUS_CODE

CHARSET
Optional Parameter

MEDIATYPE
Optional Parameter

SET_BUFFER
Optional Parameter

WBCL gate, START_BROWSE_HEADERS function

The START_BROWSE_HEADERS function starts a browse of the HTTP headers for a response that has been received.

Input Parameters

SESSION_TOKEN

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED
BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED
CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR
EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR
HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN

INVALID_URL
MEDIATYPE_INVALID
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS
NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR
URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
OMITTED_PARAMETER

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBCL gate, WRITE_HEADER function

The WRITE_HEADER function adds one HTTP header to the HTTP request being composed. It can be called multiple times to add multiple headers.

Input Parameters

NAME
SESSION_TOKEN
VALUE

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED
BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED

CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR
EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR
HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN
INVALID_URL
MEDIATYPE_INVALID
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS
NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR
URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is DISASTER or EXCEPTION
UNEXPECTED_SOCK_RESPONSE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
OMITTED_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBCL gate, WRITE_REQUEST function

The WRITE_REQUEST function appends the request body to the HTTP request being composed, and schedules it to be sent. It also handles sending a chunk of data.

Input Parameters

METHOD

Values for the parameter are:

```
DELETE
GET
HEADS
LINK
OPTIONS
POST
PUT
REQUEUE
TRACE
UNLINK
```

SESSION_TOKEN

ACTION

Optional Parameter

Values for the parameter are:

```
EVENTUAL
EXPECT
IMMEDIATE
```

ACTION_PARAMETER

Optional Parameter

BODY

Optional Parameter

A buffer that contains the HTTP request.

CHARSET

Optional Parameter

CHUNK

Optional Parameter

A block that contains chunked data.

CLOSE

Optional Parameter

Values for the parameter are:

```
NO
YES
```

CONTAINER_NAME

Optional Parameter

The name of the container that contains the request.

CONTAINER_POOL

Optional Parameter

The container pool of which the named container is a member.

CONVERSE

Optional Parameter

Values for the parameter are:

```
NO
YES
```

DOCUMENT_TOKEN
Optional Parameter

The 8 byte field into which CICS places the document token identifying the document which contains the body of the HTTP response

HOST_CODEPAGE
Optional Parameter

MEDIATYPE
Optional Parameter

PATH
Optional Parameter

QUERY_STRING
Optional Parameter

TRANSLATE
Optional Parameter

Values for the parameter are:
NO
YES

URIMAP
Optional Parameter

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
EXIT_LINKAGE_ERROR
FREEMAIN_FAILED
GETMAIN_FAILED
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BODY_NOT_ALLOWED
BODY_REQUIRED
BODY_TRUNCATED
BROWSE_ERROR
CHUNKING_ERROR
CHUNKING_NOT_SUPPORTED
COMBINATION_UNSUPPORTED
CONNECT_FAILED
CONNECTION_CLOSE_SENT
CONNECTION_CLOSED
CONTAINER_NOT_FOUND
END_HEADERS
ESCAPE_ERROR
EXPECT_REJECTED
HEADER_NAME_LENGTH_ERROR
HEADER_NOT_FOUND
HEADER_VALUE_LENGTH_ERROR
HTTP_ERROR
INVALID_CHARSET
INVALID_CHUNK
INVALID_CLIENT_CERTIFICATE
INVALID_DOCUMENT_TOKEN
INVALID_HOST
INVALID_HOST_CODEPAGE
INVALID_MEDIATYPE

INVALID_PATH
INVALID_RESPONSE_HEADER
INVALID_SCHEME
INVALID_SESSION_TOKEN
INVALID_URL
MEDIATYPE_NOT_ALLOWED
MEDIATYPE_REQUIRED
METHOD_NOT_ALLOWED
NO_RESPONSE_HEADERS
NOT_AUTHORIZED
PARTIAL_BODY
PIPELINING_ERROR
PROXY_ERROR
SOCKET_ERROR
STATUS_TEXT_TRUNCATED
TIMED_OUT
TRANSLATE_ERROR
UNKNOWN_HOST
UNKNOWN_PROXY
URIMAP_DISABLED
URIMAP_HOST_ERROR
URIMAP_NOT_FOUND
URIMAP_PATH_ERROR
XWBOPEN_ERROR

The following values are returned when RESPONSE is DISASTER or EXCEPTION

UNEXPECTED_SOCK_RESPONSE

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
OMITTED_PARAMETER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBFM gate, PARSE_MULTIPART_FORM function

This function takes a form encoded as multipart form data (with media type multipart/form-data) and converts it into a formfield structure.

Input parameters

SOURCE_DATA
This is the area containing the source data, consisting of a multipart/form-data message body.

CLIENT_CCSID
Optional parameter. This is used to interpret the value strings for multipart forms data.

SOURCE_CCSID
Optional parameter. This is the source CCSID in which the form boundary string is provided. If omitted, it defaults to the client CCSID.

TARGET_CCSID
Optional parameter. This is the target CCSID in which output data is required. If omitted, it defaults to the source CCSID.

FORM_BOUNDARY
This specifies the boundary string in the source CCSID. If the source CCSID is

not 819, the boundary string is copied and converted from the source CCSID to CCSID 819 before being used to scan the message body.

SUBPOOL_TOKEN
Optional parameter. This specifies the token for the subpool from which the form field structure storage is to be allocated. If this is omitted, the storage is obtained from STORAGE_CLASS(TASK31).

Output Parameters

FORM_STRUCTURE
This is the returned address and length of the allocated form field structure.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CCSID_CONVERSION_ERROR
CCSID_NOT_SUPPORTED
FORMS_DECODE_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETER
```

The following values are returned when RESPONSE is DISASTER:

```
ABEND
INTERNAL_ERROR
FAILURE
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBFM gate, PARSE_URL_ENCODED_FORM function

This function takes a URL-encoded forms data stream and converts it into a form field structure.

Input parameters

SOURCE_DATA
This is the area containing the source data, consisting of URL-encoded forms data.

CLIENT_CCSID
Optional parameter. This is used to interpret percent-encoded escape sequences for URL-encoded data. If omitted, the default value is 819.

SOURCE_CCSID
Optional parameter. This is the source CCSID in which URL-encoded input data is provided. If omitted, it defaults to the client CCSID.

TARGET_CCSID
Optional parameter. This is the target CCSID in which output data is required. If omitted, it defaults to the source CCSID.

SUBPOOL_TOKEN
Optional parameter. This specifies the token for the subpool from which the form field structure storage is to be allocated. If this is omitted, the storage is obtained from STORAGE_CLASS(TASK31).

Output Parameters

FORM_STRUCTURE
This is the returned address and length of the allocated form field structure.

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
CCSID_CONVERSION_ERROR
```

CCSID_NOT_SUPPORTED
FORMS_DECODE_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETER

The following values are returned when RESPONSE is DISASTER:
ABEND
INTERNAL_ERROR
FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBFM gate, PARSE_URL_ENCODED_LIST function

This function takes a URL-encoded forms data stream and converts it into a form field structure.

Input parameters

SOURCE_DATA
This is the area containing the source data, consisting of URL-encoded forms data.

CLIENT_CCSID
Optional parameter. This is used to interpret percent-encoded escape sequences for URL-encoded data. If omitted, the default value is 819.

SOURCE_CCSID
Optional parameter. This is the source CCSID in which URL-encoded input data is provided. If omitted, it defaults to the client CCSID.

TARGET_CCSID
Optional parameter. This is the target CCSID in which output data is required. If omitted, it defaults to the source CCSID.

NAME_DELIMITER
Optional parameter. This delimiter separates names from values in the source CCSID. The default value is "=".

FIELD_DELIMITER
Optional parameter. This delimiter separates name-value pairs in the source CCSID. The default value is "&".

PRIV_DELIMITER
Optional parameter. This is an alternative delimiter for name-value pairs in the source CCSID. If omitted, no alternative delimiter is used.

UNESCAPE
Optional parameter. This specifies whether percent-encoded escape sequences should be resolved during PARSE_URL_ENCODED_FORM processing. The default value is YES.

SUBPOOL_TOKEN
Optional parameter. This specifies the token for the subpool from which the form field structure storage is to be allocated. If this is omitted, the storage is obtained from STORAGE_CLASS(TASK31).

Output Parameters

FORM_STRUCTURE
This is the returned address and length of the form field structure allocated for forms requests.

REASON
The following values are returned when RESPONSE is EXCEPTION:

CCSID_CONVERSION_ERROR
CCSID_NOT_SUPPORTED
FORMS_DECODE_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETER

The following values are returned when RESPONSE is DISASTER:
ABEND
INTERNAL_ERROR
FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBFM gate, URL_DECODE function

This function processes a URL-encoded value in the source CCSID and converts it to a standard character string in the target CCSID.

Input parameters

SOURCE_DATA
This is the area containing the source data, consisting of URL-encoded forms data.

TARGET_BUFFER
This specifies the target buffer. If the specified buffer size is too small but other processing is successful, an OUTPUT_BUFFER_OVERFLOW exception is indicated and the required total size is returned as the actual length.

CLIENT_CCSID
Optional parameter. This is used to interpret percent-encoded escape sequences for URL-encoded data. If omitted, the default value is 819.

SOURCE_CCSID
Optional parameter. This is the source CCSID in which URL-encoded input data is provided. If omitted, it defaults to the client CCSID.

TARGET_CCSID
Optional parameter. This is the target CCSID in which output data is required. If omitted, it defaults to the source CCSID.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
OUTPUT_BUFFER_OVERFLOW
CCSID_CONVERSION_ERROR
CCSID_NOT_SUPPORTED
FORMS_DECODE_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_PARAMETER

The following values are returned when RESPONSE is DISASTER:
ABEND
INTERNAL_ERROR
FAILURE

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBSR gate, PEEK function

The PEEK function peeks an HTTP request's header to determine whether the request can directly attach the user transaction or not.

Input Parameters

SOCKET_TOKEN
Associated socket token of the request.

Output Parameters

TOKEN
Token uniquely identifying the WebRequestBlock associated with this HTTP request.

ATTACH_TRANID
Transaction ID of web alias transaction to be attached to continue processing the HTTP request.

REASON
The following values are returned when RESPONSE is DISASTER:

```
LOGIC_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
NOT_ELIGIBLE
PEEK_ERROR
METHOD_NOT_IMPLEMENTED
HEADER_CONVERSTION_ERROR
UNAVAILABLE
URIMAP_LOOKUP_FAILED
CLIENT_AUTHENTICATE_ERROR
BASIC_AUTHENTICATE_ERROR
CONNECTION_CLOSED
```

RESPONSE
Indicates whether the domain call was successful. For more information, see RESPONSE parameter on domain interfaces.

WBSR gate, RECEIVE function

The RECEIVE function receives an HTTP Request off a socket, and parses it in order to determine what to do with it.

Input Parameters

INITIAL_RECEIVE
Indicates whether this is the first receive issued by the caller.

Values for the parameter are:

```
NO
YES
```

INTOKEN
Optional: Token Uniquely identifying the **WebRequestBlock** associated with this HTTP request.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
LOGIC_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

ANALYZER_ABEND
ANALYZER_DATALENG_ERROR
ANALYZER_ERROR
ANALYZER_LINK_ERROR
BASIC_AUTHENTICATE_ERROR
CHARACTERSET_ERROR
CHUNKED_CONTENT_CONFLICT
CLIENT_AUTHENTICATE_ERROR
CLIENT_ERROR
CODEPAGE_CONVERSION_ERROR
CONNECTION_CLOSED
DATA_LENGTH_EXCEEDED
GETMAIN_FAILED
HDR_LENGTH_ERROR
HEADER_CONVERSION_ERROR
HOSTCODEPAGE_ERROR
HTTP10_INVALID_EXPECT
INSUFFICIENT_THREADS
INVALID_CHUNK
INVALID_CHUNK_SIZE_HEADER
INVALID_EXPECT_HEADER
INVALID_STATIC_TYPE
METHOD_NOT_IMPLEMENTED
NO_ANALYZER_SPECIFIED
NO_DATA
NO_HOST_HEADER
NON_HTTP_DATA
PRECONDITION_FAILED
RECEIVE_ERROR
REQUEST_TIMEOUT
SEND_ERROR
SSL_HANDSHAKE_ERROR
STATIC_DATA_NOT_FOUND
STATIC_DATA_NOTAUTH
STATIC_DATA_READ_ERROR
TRAILER_LENGTH_ERROR
UNAVAILABLE
USER_DATA_CONVERSION_ERROR
VERSION_NOT_SUPPORTED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_SESSION_TOKEN

ATTACH_TRANSID

Optional: Transaction ID of Web alias transaction to be attached to continue processing the HTTP request.

CONNECTION_PERSIST

Indicates whether the HTTP Request included the HTTP 1.0 Keepalive header.

Values for the parameter are:

NO
YES

FAILING_PROGRAM

Name of program which returned an error in the course of receiving the HTTP request.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TOKEN
Optional: Token uniquely identifying the **WebRequestBlock** associated with this HTTP request.

WBSR gate, SEND function

The SEND function returns the response constructed following receipt of an HTTP request.

Input Parameters

TOKEN
Token identifying WebRequestBlock with which this SEND is associated

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
LOGIC_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
ANALYZER_ABEND
ANALYZER_DATALENG_ERROR
ANALYZER_ERROR
ANALYZER_LINK_ERROR
BASIC_AUTHENTICATE_ERROR
CHARACTERSET_ERROR
CHUNKED_CONTENT_CONFLICT
CLIENT_AUTHENTICATE_ERROR
CLIENT_ERROR
CODEPAGE_CONVERSION_ERROR
CONNECTION_CLOSED
DATA_LENGTH_EXCEEDED
GETMAIN_FAILED
HDR_LENGTH_ERROR
HEADER_CONVERSION_ERROR
HOSTCODEPAGE_ERROR
HTTP10_INVALID_EXPECT
INSUFFICIENT_THREADS
INVALID_CHUNK
INVALID_CHUNK_SIZE_HEADER
INVALID_EXPECT_HEADER
INVALID_STATIC_TYPE
METHOD_NOT_IMPLEMENTED
NO_ANALYZER_SPECIFIED
NO_DATA
NO_HOST_HEADER
NON_HTTP_DATA
PRECONDITION_FAILED
RECEIVE_ERROR
REQUEST_TIMEOUT
SEND_ERROR
SSL_HANDSHAKE_ERROR
STATIC_DATA_NOT_FOUND
STATIC_DATA_NOTAUTH
STATIC_DATA_READ_ERROR
TRAILER_LENGTH_ERROR
```

UNAVAILABLE
USER_DATA_CONVERSION_ERROR
VERSION_NOT_SUPPORTED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_SESSION_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBSR gate, SEND_STATIC_RESPONSE function

The SEND_STATIC_RESPONSE function returns a static response specified by a URIMAP definition following receipt of an HTTP request.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ANALYZER_ABEND
ANALYZER_DATALENG_ERROR
ANALYZER_ERROR
ANALYZER_LINK_ERROR
BASIC_AUTHENTICATE_ERROR
CHARACTERSET_ERROR
CHUNKED_CONTENT_CONFLICT
CLIENT_AUTHENTICATE_ERROR
CLIENT_ERROR
CODEPAGE_CONVERSION_ERROR
CONNECTION_CLOSED
DATA_LENGTH_EXCEEDED
GETMAIN_FAILED
HDR_LENGTH_ERROR
HEADER_CONVERSION_ERROR
HOSTCODEPAGE_ERROR
HTTP10_INVALID_EXPECT
INSUFFICIENT_THREADS
INVALID_CHUNK
INVALID_CHUNK_SIZE_HEADER
INVALID_EXPECT_HEADER
INVALID_STATIC_TYPE
METHOD_NOT_IMPLEMENTED
NO_ANALYZER_SPECIFIED
NO_DATA
NO_HOST_HEADER
NON_HTTP_DATA
PRECONDITION_FAILED
RECEIVE_ERROR
REQUEST_TIMEOUT
SEND_ERROR
SSL_HANDSHAKE_ERROR
STATIC_DATA_NOT_FOUND
STATIC_DATA_NOTAUTH
STATIC_DATA_READ_ERROR

TRAILER_LENGTH_ERROR
UNAVAILABLE
USER_DATA_CONVERSION_ERROR
USERID_NOTAUTH
VERSION_NOT_SUPPORTED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_SESSION_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBSV gate, READ_REQUEST function

The READ_REQUEST function receives the HTTP message body.

Input parameters

SESSION_TOKEN
This is the Web server session token.

BODY_BUFFER
This is the buffer to be received as part of the HTTP message.

RECEIVE_TYPE
This is the type of receive. Values for the parameter are:

SYNC
ASYNC

TRUNCATE
Optional parameter. This specifies whether the HTTP message is to be truncated. Values for the parameter are

NO
YES

:

TIME_OUT_VALUE
Optional parameter. This is the timeout period, in seconds, when receiving an HTTP message.

Output parameters

MEDIATYPE
Optional parameter. This is the mediatype of the HTTP message.

CONTENT_LENGTH
Optional parameter. This is the length in the HTTP Content-Length header.

REASON
The following values are returned when RESPONSE is EXCEPTION:

PARTIAL_BODY
BODY_TRUNCATED
HEADERS_MAXLEN_EXCEEDED
NO_CONTENT_LENGTH
FIRST_LINE_INVALID
SCHEDULED
HEADERS_PARTLY_PEEKED
ASYNC_TRUNCATE_INVALID
BODY_ALREADY_RECEIVED
TIMED_OUT
INVALID_SESSION_TOKEN
CONNECTION_CLOSED

CONNECTION_CLOSED_ASYNC0
SOCKET_ERROR
SOCKET_ERROR_ASYNC0

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

WBSV gate, WRITE_RESPONSE function

The WRITE_RESPONSE function sends the HTTP message.

Input parameters

SESSION_TOKEN
This is the Web server session token.

BODY | BODY_LIST
This is the body list to be sent as the body of an HTTP message.

MEDIATYPE
This is the media type of the HTTP message.

STATUS_TEXT
This is the status text to be sent with the HTTP message. The default is "OK".

STATUS_CODE
This is the status code to be sent with the HTTP message. The default is 200.

HEADER1_NAME
This is the name of the first additional HTTP header to be sent with the HTTP message.

HEADER2_NAME
This is the name of the second HTTP header to be sent with the HTTP message.

HEADER2_VALUE
This is the value of the second HTTP header to be sent with the HTTP message.

HEADER3_NAME
This is the name of the third HTTP header to be sent with the HTTP message.

HEADER3_VALUE
This is the value of the third HTTP header to be sent with the HTTP message.

HEADER_NAME_LIST
This is the list of header names to be sent with the HTTP message.

HEADER_VALUE_LIST
This is the list of header values to be sent with the HTTP message.

TIME_OUT_VALUE
This is the timeout value to be applied to the socket SEND for the response.

Output parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
HEADER1_NAME_NOTALLOWED
HEADER2_NAME_NOTALLOWED
HEADER3_NAME_NOTALLOWED
HEADERLIST_NAME_NOTALLOWED
MAX_LIST_SIZE_EXCEEDED

NO_CLIENT_CHARSET
HEADERS_MAXLEN_EXCEEDED
INVALID_SESSION_TOKEN
INVALID_MEDIATYPE
CONNECTION_CLOSED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBSV gate, PEEK_HEADERS function

The PEEK_HEADERS function peeks the HTTP headers.

Input parameters

SESSION_TOKEN
This is the Web server session token.

HEADER1_NAME
Optional parameter. This is the name of the first additional HTTP header to be sent with the HTTP message.

HEADER2_NAME
Optional parameter. This is the name of the second HTTP header to be sent with the HTTP message.

HEADER3_NAME
Optional parameter. This is the name of the third HTTP header to be sent with the HTTP message.

HEADER_NAME_LIST
Optional parameter. This is the list of header names to be sent with the HTTP message.

HEADER_VALUE_LIST
This is the list of header values to be sent with the HTTP message.

HEADERS_OPTIONAL
Optional parameter. This specifies whether an exception response should be suppressed if specified headers are missing. Values for the parameter are:
YES
NO

Output parameters

HEADER1_VALUE_SETBUF
Optional parameter. This is the block for HEADER1_VALUE_SETDATA. It sets the buffer to be returned to the caller.

HEADER2_VALUE_SETBUF
Optional parameter. This is the block for HEADER2_VALUE_SETDATA. It sets the buffer to be returned to the caller.

HEADER3_VALUE_SETBUF
Optional parameter. This is the block for HEADER3_VALUE_SETDATA. It sets the buffer to be returned to the caller.

CONTENT_LENGTH
This is the length in the HTTP Content-Length header.

REASON
The following values are returned when RESPONSE is EXCEPTION:
SCHEDULED
HEADERS_MAXLEN_EXCEEDED
NO_CONTENT_LENGTH
NO_CONTENT_TYPE
FIRST_LINE_INVALID
NO_STORAGE_AVAILABLE
HEADER1_TOO_LONG
HEADER2_TOO_LONG
HEADER3_TOO_LONG
UNSUPPORTED_VERSION
INVALID_CONTENT_LENGTH
NO_HEADER1
NO_HEADER2
NO_HEADER3
METHOD_NOT_SUPPORTED
HEADER1_INVALID
HEADER2_INVALID
HEADER3_INVALID
HEADER1_EQ_HEADER2
LAST_BODY_NOT_RECEIVED
INVALID_SESSION_TOKEN
CONNECTION_CLOSED
CONNECTION_CLOSED_ASYNC0
SOCKET_ERROR
SOCKET_ERROR_ASYNC0
INVALID_PARAMETERS
LIST_HEADER_MISSING
HEADER_NAME_TOO_LONG
HEADER_INVALID

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

WBSV gate, INQUIRE_CURRENT_SESSION function

The INQUIRE_CURRENT_SESSION function inquires on the current session.

Output parameters

SESSION_TOKEN
Optional parameter. This is the Web server session token.

USER_TOKEN
Optional parameter. This is the ISC user token.

SERVER_BIN_IP_ADDRESS
Optional parameter. This is the server IP address returned by a SOCKETS_INQUIRE.

SERVER_IP_ADDRESS
Optional parameter. This is the IP address of the server.

SERVER_IP_ADDRESS_LEN
Optional parameter. This is the length of the server IP address.
SERVER_IP_ADDRESS_TYPE
Optional parameter. This is address type of the server IP address.
CLIENT_BIN_IP_ADDRESS
Optional parameter. This is the binary form of the client IP address.
CLIENT_IP_ADDRESS
Optional parameter. This is the IP address of the client.
CLIENT_IP_ADDRESS_LEN
Optional parameter. This is the length of the client IP address.
CLIENT_IP_ADDRESS_TYPE
Optional parameter. This is address type of the client IP address.
TCPIPSERVICE_NAME
Optional parameter. This is the TCPIP service name returned by SOCKETS_INQUIRE.
TRANSID
Optional parameter. This is the transaction ID returned by SOCKETS_INQUIRE.
LISTENER_PORT
Optional parameter. This is the listener port number returned by SOCKETS_INQUIRE.
SSLTYPE
Optional parameter. This is the SSL type returned by SOCKETS_INQUIRE.
PROTOCOL
Optional parameter. This is the protocol returned by SOCKETS_INQUIRE.
SOCKET_TOKEN
Optional parameter. This is the socket token returned by SOCKETS_INQUIRE.
CLUSTER_TYPE
Optional parameter. This is the cluster type returned by SOCKETS_INQUIRE. Values for the parameter are:
NONE
SAME_SYSPLEX
SAME_IMAGE
SAME_STACK
REASON
The following values are returned when RESPONSE is EXCEPTION:
NON_WEB_TRANSACTION
SESSION_CLOSED
INFO_NOT_AVAILABLE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

WBSV gate, SET_SESSION function

The SET_SESSION function sets the current session.

Input parameters

SESSION_TOKEN
This is the Web server session token.

USER_TOKEN
Optional parameter. This is the ISC user token.
APPLDATA_SFX
Optional parameter. This is the Application Data Suffix.
TRACE_SUPPRESSION
Optional parameter. This specifies whether tracing of the HTTP body is to be suppressed by the Socket Domain. Values for the parameter are:
YES
NO

Output parameters
REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

WBSV gate, CLOSE_SESSION function
The CLOSE_SESSION function sets the current session.

Input parameters
SESSION_TOKEN
This is the Web server session token.

Output parameters
REASON
The following values are returned when RESPONSE is EXCEPTION:
INVALID_SESSION_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

WBSV gate, INQUIRE_SESSION function
The INQUIRE_SESSION function inquires on the session.

Input parameters
SESSION_TOKEN
This is the Web server session token.

Output parameters

SOCKET_TOKEN
Optional parameter.

REASON
The following values are returned when RESPONSE is EXCEPTION:
```
INVALID_SESSION_TOKEN
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```

The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7

WBUR gate, ADD_REPLACE_URIMAP function

The ADD_REPLACE_URIMAP function adds or replaces a URIMAP resource into the Web domain. The parameters correspond to attributes specified on the URIMAP definition.

Input Parameters

HOST
The host name of the URI to which the URIMAP resource applies, or its IPv4 or IPv6 address.

PATH
The path component of the URI to which the URIMAP resource applies.

URIMAP
The name of the URIMAP resource.

ANALYZER
Optional Parameter

A binary value that specifies whether an analyzer program is to be used in processing HTTP requests.

Values for the parameter are:
```
NO
YES
```

CERTIFICATE_LABEL
Optional Parameter

The label of the X.509 certificate that is to be used as the SSL client certificate during the SSL handshake.

CHARACTERSET
Optional Parameter

The character set into which CICS converts the entity body of the response that is sent to the web client.

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the **CIPHER_SUITES** parameter.

CIPHER_SUITES
Optional Parameter

A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

CONVERTER
Optional Parameter

The name of a converter program that is to be run to perform conversion or other processing on the request and response.

HFSFILE
Optional Parameter

The fully qualified or relative name of a zFS file that forms the body of the static response that is sent to the HTTP request from the web client.

HOSTCODEPAGE
Optional Parameter

The EBCDIC code page in which the text document that forms the static response is encoded.

MEDIATYPE
Optional Parameter

The media type of the static response that CICS provides to the HTTP request, for example `image/jpg`, `text/html`, or `text/xml`.

PIPELINE_NAME
Optional Parameter

The PIPELINE resource used by web service requests for the URIMAP resource.

PORT
Optional Parameter

The decimal number of the port used by a CICS application when it communicates with a server using the URIMAP resource.

PROGRAM
Optional Parameter

The name of the user application program that composes the HTTP response for the URIMAP resource.

REDIRECTION_LOCATION
Optional Parameter

A URL to which the client's request should be redirected.

REDIRECTION_TYPE
Optional Parameter

The type of redirection for requests that match the URIMAP resource. When redirection is required, the REDIRECTION_LOCATION parameter specifies the URL to which the request should be redirected.

Values for the parameter are:
`NONE`
`PERMANENT`
`TEMPORARY`

SCHEME
Optional Parameter

The scheme component of the URI to which the URIMAP resource applies.

Values for the parameter are:
`HTTP`
`HTTPS`
`WMQ`

SOCKETCLOSE
Optional Parameter

The length of time for which CICS keeps a client HTTP connection open after the CICS application has finished using it.

STATUS
Optional Parameter

The enabled or disabled state of the URIMAP resource.

Values for the parameter are:
DISABLED
DISABLEDHOST
ENABLED

TCPIPSERVICE
Optional Parameter

The name of the TCPIPSERVICE resource that defines the inbound port to which the URIMAP resource relates.

TEMPLATENAME
Optional Parameter

The name of a CICS document template that forms the body of the static response that is sent to the HTTP request from the web client.

TRANSACTION
Optional Parameter

The name of an alias transaction that is to be used to run the user application that composes the HTTP response, or to start the pipeline.

USAGE
Optional Parameter

Specifies how the URIMAP resource is used.

Values for the parameter are:
ATOM
CLIENT
PIPELINE
SERVER

USERID
Optional Parameter

The user ID under which requests for the URIMAP resource are initially processed.

WEBSERVICE_NAME
Optional Parameter

The name of a WEBSERVICE resource associated with the URIMAP resource.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET

INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
CONFLICTING_ATTRIBUTES

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

DUPLICATE_URIMAP
Optional Parameter

The name of a URIMAP resource that already exists and would be duplicated by this request.

WBUR gate, DELETE_URIMAP function

The DELETE_URIMAP function deletes a URIMAP definition from the Web domain.

Input Parameters

URIMAP

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBUR gate, END_BROWSE_HOST function

The END_BROWSE_HOST function is used to end a browse of the virtual host names in the Web domain.

Input Parameters

BROWSE_TOKEN

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBUR gate, END_BROWSE_URIMAP function

The END_BROWSE_URIMAP function is used to end a browse through the URIMAP resources in the Web domain.

Input Parameters

BROWSE_TOKEN

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

LOOP

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBUR gate, FIND_URIMAP function

The FIND_URIMAP function is used to locate a URIMAP definition that is associated with a specified HOST and PATH.

Input Parameters

HOST

The host name of the URI to which the URIMAP resource applies, or its IPv4 or IPv6 address.

PATH

The path component of the URI to which the URIMAP resource applies.

HFSFILE

Optional Parameter

The fully qualified or relative name of a zFS file that forms the body of the static response that is sent to the HTTP request from the web client.

TCPIPSERVICE

Optional Parameter

The name of the TCPIPSERVICE resource that defines the inbound port to which the URIMAP resource relates.

Output Parameters

URIMAP

The name of the URIMAP resource.

ANALYZER

Optional Parameter

A binary value that specifies whether an analyzer program is to be used in processing HTTP requests.

Values for the parameter are:

NO
YES

CERTIFICATE_LABEL
Optional Parameter

The label of the X.509 certificate that is to be used as the SSL client certificate during the SSL handshake.

CHARACTERSET
Optional Parameter

The character set into which CICS converts the entity body of the response that is sent to the web client.

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the **CIPHER_SUITES** parameter.

CIPHER_SUITES
Optional Parameter

A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

CONVERTER
Optional Parameter

The name of a converter program that is to be run to perform conversion or other processing on the request and response.

HOSTCODEPAGE
Optional Parameter

The EBCDIC code page in which the text document that forms the static response is encoded.

MEDIATYPE
Optional Parameter

The media type of the static response that CICS provides to the HTTP request, for example `image/jpg`, `text/html`, or `text/xml`.

PIPELINE_NAME
Optional Parameter

The PIPELINE resource used by web service requests for the URIMAP resource.

PROGRAM
Optional Parameter

The name of the user application program that composes the HTTP response for the URIMAP resource.

REDIRECTION_TYPE
Optional Parameter

The type of redirection for requests that match the URIMAP resource. When redirection is required, the REDIRECTION_LOCATION parameter specifies the URL to which the request should be redirected.

Values for the parameter are:

NONE
PERMANENT
TEMPORARY

SCHEME
Optional Parameter

The scheme component of the URI to which the URIMAP resource applies.

Values for the parameter are:
HTTP
HTTPS
WMQ

SOCKETCLOSE
Optional Parameter

The length of time for which CICS keeps a client HTTP connection open after the CICS application has finished using it.

STATUS
Optional Parameter

The enabled or disabled state of the URIMAP resource.

Values for the parameter are:
DISABLED
DISABLEDHOST
ENABLED

TEMPLATENAME
Optional Parameter

The name of a CICS document template that forms the body of the static response that is sent to the HTTP request from the web client.

TRANSACTION
Optional Parameter

The name of an alias transaction that is to be used to run the user application that composes the HTTP response, or to start the pipeline.

UME_TOKEN
Optional parameter

The address of the URIMAP.

USAGE
Optional Parameter

Specifies how the URIMAP resource is used.

Values for the parameter are:
ATOM
CLIENT
PIPELINE
SERVER

USERID
Optional Parameter

The user ID under which requests for the URIMAP resource are initially processed.

WEBSERVICE_NAME
Optional Parameter

The name of a WEBSERVICE resource associated with the URIMAP resource.

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

LOOP

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT

INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBUR gate, GET_NEXT_HOST function

The GET_NEXT_HOST function is used to continue a browse through the virtual host names in the Web domain.

Input Parameters

BROWSE_TOKEN
HOST

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION

NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATUS

Values for the parameter are:

DISABLED
DISABLEDHOST
ENABLED

TCPIPSERVICE

WBUR gate, GET_NEXT_URIMAP function

The GET_NEXT_URIMAP function is used to continue a browse through the URIMAP resources in the Web domain.

Input Parameters

BROWSE_TOKEN
See "The **BROWSE_TOKEN** parameter on domain interfaces" on page 7

HFSFILE
Optional Parameter

The fully qualified or relative name of a zFS file that forms the body of the static response that is sent to the HTTP request from the web client.

HOST
The host name of the URI to which the URIMAP resource applies, or its IPv4 or IPv6 address.

PATH
The path component of the URI to which the URIMAP resource applies.

REDIRECTION_LOCATION
Optional Parameter

A URL to which the client's request should be redirected.

RESOURCE_SIGNATURE
Optional Parameter

The resource signature for the URIMAP resource definition.

APPLICATION_NAME
Optional Parameter

The name of the application for which the URIMAP is declared as an application entry point.

PLATFORM_NAME
Optional Parameter

The name of the platform where the application is deployed for which the URIMAP is declared as an application entry point.

OPERATION_NAME
Optional Parameter

The name of the application operation for which the URIMAP is declared as an application entry point.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

URIMAP

The name of the URIMAP resource.

ANALYZER

Optional Parameter

A binary value that specifies whether an analyzer program is to be used in processing HTTP requests.

Values for the parameter are:

NO
YES

AVAILSTATUS

Optional parameter

A 1-byte enumeration expressing the availability status of the CICS bundle.

Values for the parameter are:

AVAILABLE
UNAVAILABLE
NONE

CERTIFICATE_LABEL

Optional Parameter

The label of the X.509 certificate that is to be used as the SSL client certificate during the SSL handshake.

CHARACTERSET

Optional Parameter

The character set into which CICS converts the entity body of the response that is sent to the web client.

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the **CIPHER_SUITES** parameter.

CIPHER_SUITES
Optional Parameter

A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

CONVERTER
Optional Parameter

The name of a converter program that is to be run to perform conversion or other processing on the request and response.

HOSTCODEPAGE
Optional Parameter

The EBCDIC code page in which the text document that forms the static response is encoded.

MEDIATYPE
Optional Parameter

The media type of the static response that CICS provides to the HTTP request, for example `image/jpg`, `text/html`, or `text/xml`.

PIPELINE_NAME
Optional Parameter

The PIPELINE resource used by web service requests for the URIMAP resource.

PROGRAM
Optional Parameter

The name of the user application program that composes the HTTP response for the URIMAP resource.

REDIRECTION_TYPE
Optional Parameter

The type of redirection for requests that match the URIMAP resource. When redirection is required, the REDIRECTION_LOCATION parameter specifies the URL to which the request should be redirected.

Values for the parameter are:

```
NONE
PERMANENT
TEMPORARY
```

SCHEME
Optional Parameter

The scheme component of the URI to which the URIMAP resource applies.

Values for the parameter are:

```
HTTP
HTTPS
WMQ
```

SOCKETCLOSE
Optional Parameter

The length of time for which CICS keeps a client HTTP connection open after the CICS application has finished using it.

SOCKPOOLSIZE
Optional Parameter

The number of client HTTP connections made using this URIMAP resource that CICS is currently holding in a pool in a dormant state.

SOCKETPOOL_TKN
Optional Parameter

The unique identifier for the pool of opened client HTTP connections made using this URIMAP resource.

STATUS
Optional Parameter

The enabled or disabled state of the URIMAP resource.

Values for the parameter are:
DISABLED
DISABLEDHOST
ENABLED

TCPIPSERVICE
Optional Parameter

The name of the TCPIPSERVICE resource that defines the inbound port to which the URIMAP resource relates.

TEMPLATENAME
Optional Parameter

The name of a CICS document template that forms the body of the static response that is sent to the HTTP request from the web client.

TRANSACTION
Optional Parameter

The name of an alias transaction that is to be used to run the user application that composes the HTTP response, or to start the pipeline.

USAGE
Optional Parameter

Specifies how the URIMAP resource is used.

Values for the parameter are:
ATOM
CLIENT
PIPELINE
SERVER

USERID
Optional Parameter

The user ID under which requests for the URIMAP resource are initially processed.

WEBSERVICE_NAME
Optional Parameter

The name of a WEBSERVICE resource associated with the URIMAP resource.

WBUR gate, INITIALIZE_URIMAPS function

The INITIALIZE_URIMAPS function initializes the Web domain state required by the URIMAP support.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND

LOOP

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBUR gate, INQUIRE_HOST function

The INQUIRE_HOST function is used to inquire on the attributes of a virtual host.

Input Parameters

HOST

TCPIPSERVICE

Optional Parameter

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND

NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

STATUS

Values for the parameter are:
DISABLED
DISABLEDHOST
ENABLED

WBUR gate, INQUIRE_URIMAP function

The INQUIRE_URIMAP function is used to inquire on the attributes of a URIMAP resource.

Input Parameters

APPLICATION_NAME
Optional Parameter

The 64-character buffer to receive the name of the application if the URIMAP is declared as an application entry point.

HFSFILE
Optional Parameter

The fully qualified or relative name of a zFS file that forms the body of the static response that is sent to the HTTP request from the web client.

HOST
The host name of the URI to which the URIMAP resource applies, or its IPv4 or IPv6 address.

OPERATION_NAME
Optional Parameter

The 64-character buffer to receive the name of the application operation if the URIMAP is declared as an application entry point.

PATH
The path component of the URI to which the URIMAP resource applies.

PLATFORM_NAME
Optional Parameter

The 64-character buffer to receive the name of the platform if the URIMAP is declared as an application entry point.

REDIRECTION_LOCATION
Optional Parameter

A URL to which the client's request should be redirected.

RESOURCE_SIGNATURE
Optional Parameter

The resource signature for the URIMAP resource definition.

URIMAP
The name of the URIMAP resource.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ANALYZER

Optional Parameter

A binary value that specifies whether an analyzer program is to be used in processing HTTP requests.

Values for the parameter are:

NO
YES

AVAILSTATUS

Optional parameter

A 1-byte enumeration expressing the availability status of the CICS bundle.

Values for the parameter are:

AVAILABLE
UNAVAILABLE
NONE

BUNDLE_TOKEN

Optional Parameter

An 8-byte token that represents the CICS bundle in which the URIMAP is declared as an application entry point.

CERTIFICATE_LABEL

Optional Parameter

The label of the X.509 certificate that is to be used as the SSL client certificate during the SSL handshake.

CHARACTERSET
Optional Parameter

The character set into which CICS converts the entity body of the response that is sent to the web client.

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the **CIPHER_SUITES** parameter.

CIPHER_SUITES
Optional Parameter

A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

CONVERTER
Optional Parameter

The name of a converter program that is to be run to perform conversion or other processing on the request and response.

ENTRYPOINT_RES_TOKEN
Optional Parameter

An 8-byte token that represents the application entry point.

HOSTCODEPAGE
Optional Parameter

The EBCDIC code page in which the text document that forms the static response is encoded.

MAJOR_VERSION
Optional Parameter

The fullword binary value that is the major version of the application for which the URIMAP is declared as an application entry point.

MICRO_VERSION
Optional Parameter

The fullword binary value that is the micro version of the application for which the URIMAP is declared as an application entry point.

MINOR_VERSION
Optional Parameter

The fullword binary value that is the minor version of the application for which the URIMAP is declared as an application entry point.

MEDIATYPE
Optional Parameter

The media type of the static response that CICS provides to the HTTP request, for example `image/jpg`, `text/html`, or `text/xml`.

PIPELINE_NAME
Optional Parameter

The PIPELINE resource used by web service requests for the URIMAP resource.

PORT
Optional Parameter

The decimal number of the port used by a CICS application when it communicates with a server using the URIMAP resource.

PROGRAM
Optional Parameter

The name of the user application program that composes the HTTP response for the URIMAP resource.

REDIRECTION_TYPE

Optional Parameter

The type of redirection for requests that match the URIMAP resource. When redirection is required, the REDIRECTION_LOCATION parameter specifies the URL to which the request should be redirected.

Values for the parameter are:

```
NONE
PERMANENT
TEMPORARY
```

SCHEME

Optional Parameter

The scheme component of the URI to which the URIMAP resource applies.

Values for the parameter are:

```
HTTP
HTTPS
WMQ
```

SOCKETCLOSE

Optional Parameter

The length of time for which CICS keeps a client HTTP connection open after the CICS application has finished using it.

SOCKPOOLSIZE

Optional Parameter

The number of client HTTP connections made using this URIMAP resource that CICS is currently holding in a pool in a dormant state.

SOCKETPOOL_TKN

Optional Parameter

The unique identifier for the pool of opened client HTTP connections made using this URIMAP resource.

STATUS

Optional Parameter

The enabled or disabled state of the URIMAP resource.

Values for the parameter are:

```
DISABLED
DISABLEDHOST
ENABLED
```

TCPIPSERVICE

Optional Parameter

The name of the TCPIPSERVICE resource that defines the inbound port to which the URIMAP resource relates.

TEMPLATENAME

Optional Parameter

The name of a CICS document template that forms the body of the static response that is sent to the HTTP request from the web client.

TRANSACTION

Optional Parameter

The name of an alias transaction that is to be used to run the user application that composes the HTTP response, or to start the pipeline.

USAGE
Optional Parameter

Specifies how the URIMAP resource is used.

Values for the parameter are:
ATOM
CLIENT
PIPELINE
SERVER

USERID
Optional Parameter

The user ID under which requests for the URIMAP resource are initially processed.

WEBSERVICE_NAME
Optional Parameter

The name of a WEBSERVICE resource associated with the URIMAP resource.

WBUR gate, LOCATE_URIMAP function

The LOCATE_URIMAP function is used to locate a URIMAP definition associated with a specified HOST and PATH.

Input Parameters

HOST
The host name of the URI to which the URIMAP resource applies, or its IPv4 or IPv6 address.

PATH
The path component of the URI to which the URIMAP resource applies.

HFSFILE
Optional Parameter

The fully qualified or relative name of a zFS file that forms the body of the static response that is sent to the HTTP request from the web client.

PATH
The path component of the URI to which the URIMAP resource applies.

PORT
Optional Parameter

The decimal number of the port used by a CICS application when it communicates with a server using the URIMAP resource.

REDIRECTION_LOCATION
Optional Parameter

A URL to which the client's request should be redirected.

TCPIPSERVICE
Optional Parameter

The name of the TCPIPSERVICE resource that defines the inbound port to which the URIMAP resource relates.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
CCNV_ERROR

CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

URIMAP
The name of the URIMAP resource.

ANALYZER
Optional Parameter

A binary value that specifies whether an analyzer program is to be used in processing HTTP requests.

Values for the parameter are:
NO
YES

CERTIFICATE_LABEL
Optional Parameter

The label of the X.509 certificate that is to be used as the SSL client certificate during the SSL handshake.

CHARACTERSET
Optional Parameter

The character set into which CICS converts the entity body of the response that is sent to the web client.

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the **CIPHER_SUITES** parameter.

CIPHER_SUITES
Optional Parameter

A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

CONVERTER
Optional Parameter

The name of a converter program that is to be run to perform conversion or other processing on the request and response.

HOSTCODEPAGE
Optional Parameter

The EBCDIC code page in which the text document that forms the static response is encoded.

MEDIATYPE
Optional Parameter

The media type of the static response that CICS provides to the HTTP request, for example `image/jpg`, `text/html`, or `text/xml`.

PIPELINE_NAME
Optional Parameter

The PIPELINE resource used by web service requests for the URIMAP resource.

PROGRAM
Optional Parameter

The name of the user application program that composes the HTTP response for the URIMAP resource.

REDIRECTION_TYPE
Optional Parameter

The type of redirection for requests that match the URIMAP resource. When redirection is required, the REDIRECTION_LOCATION parameter specifies the URL to which the request should be redirected.

Values for the parameter are:
- `NONE`
- `PERMANENT`
- `TEMPORARY`

SCHEME
Optional Parameter

The scheme component of the URI to which the URIMAP resource applies.

Values for the parameter are:
- `HTTP`
- `HTTPS`
- `WMQ`

SOCKETCLOSE
Optional Parameter

The length of time for which CICS keeps a client HTTP connection open after the CICS application has finished using it.

STATUS
Optional Parameter

The enabled or disabled state of the URIMAP resource.

Values for the parameter are:
- `DISABLED`
- `DISABLEDHOST`
- `ENABLED`

TEMPLATENAME
Optional Parameter

The name of a CICS document template that forms the body of the static response that is sent to the HTTP request from the web client.

TRANSACTION
Optional Parameter

The name of an alias transaction that is to be used to run the user application that composes the HTTP response, or to start the pipeline.

UME_TOKEN
Optional parameter

The address of the URIMAP.

USAGE
Optional Parameter

Specifies how the URIMAP resource is used.

Values for the parameter are:
- ATOM
- CLIENT
- PIPELINE
- SERVER

USERID
Optional Parameter

The user ID under which requests for the URIMAP resource are initially processed.

WEBSERVICE_NAME
Optional Parameter

The name of a WEBSERVICE resource associated with the URIMAP resource.

WBUR gate, SET_HOST function

The SET_HOST function is used to set the attributes of a virtual host.

Input Parameters

HOST

STATUS

Values for the parameter are:
- DISABLED
- DISABLEDHOST
- ENABLED

TCPIPSERVICE
Optional Parameter

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:
- ABEND
- LOOP

The following values are returned when RESPONSE is EXCEPTION:
- BROWSE_END
- CCNV_ERROR
- CONFLICTING_ATTRIBUTES
- DIRECTORY_ERROR
- DUPLICATE_MAPPING
- GETMAIN_FAILED
- INVALID_BROWSE_TOKEN
- INVALID_CHARACTERSET
- INVALID_HOSTCODEPAGE
- INVALID_HOSTNAME
- INVALID_PATHNAME
- LOCATION_INVALID
- NO_REDIRECTION_LOCATION
- NOT_FOUND
- NOT_POSSIBLE

SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBUR gate, SET_URIMAP function

The SET_URIMAP function is used to set the attributes of a URIMAP resource.

Input Parameters

URIMAP
The name of the URIMAP resource.

ANALYZER
Optional Parameter

A binary value that specifies whether an analyzer program is to be used in processing HTTP requests.

Values for the parameter are:
NO
YES

AVAILSTATUS
Optional parameter

A 1-byte enumeration expressing the availability status of the CICS bundle.

Values for the parameter are:
AVAILABLE
UNAVAILABLE
NONE

ENTRYP_BUNDLE_TOKEN
Optional Parameter

An 8-byte token that represents the CICS bundle in which the URIMAP is declared as an application entry point.

CERTIFICATE_LABEL
Optional Parameter

The label of the X.509 certificate that is to be used as the SSL client certificate during the SSL handshake.

CHARACTERSET
Optional Parameter

The character set into which CICS converts the entity body of the response that is sent to the web client.

CIPHER_COUNT
Optional Parameter

The number of cipher suites encoded in the **CIPHER_SUITES** parameter.

CIPHER_SUITES
Optional Parameter

A string of up to 56 hexadecimal digits that is interpreted as a list of up to 28 2-digit cipher suite codes.

CONVERTER
Optional Parameter

The name of a converter program that is to be run to perform conversion or other processing on the request and response.

HFSFILE
Optional Parameter

The fully qualified or relative name of a zFS file that forms the body of the static response that is sent to the HTTP request from the web client.

HOST
The host name of the URI to which the URIMAP resource applies, or its IPv4 or IPv6 address.

HOSTCODEPAGE
Optional Parameter

The EBCDIC code page in which the text document that forms the static response is encoded.

MEDIATYPE
Optional Parameter

The media type of the static response that CICS provides to the HTTP request, for example `image/jpg`, `text/html`, or `text/xml`.

PATH
The path component of the URI to which the URIMAP resource applies.

PIPELINE_NAME
Optional Parameter

The PIPELINE resource used by web service requests for the URIMAP resource.

PROGRAM
Optional Parameter

The name of the user application program that composes the HTTP response for the URIMAP resource.

REDIRECTION_LOCATION
Optional Parameter

A URL to which the client's request should be redirected.

REDIRECTION_TYPE
Optional Parameter

The type of redirection for requests that match the URIMAP resource. When redirection is required, the REDIRECTION_LOCATION parameter specifies the URL to which the request should be redirected.

Values for the parameter are:
- `NONE`
- `PERMANENT`
- `TEMPORARY`

SCHEME
Optional Parameter

The scheme component of the URI to which the URIMAP resource applies.

Values for the parameter are:
- `HTTP`
- `HTTPS`
- `WMQ`

SOCKETCLOSE
Optional Parameter

The length of time for which CICS keeps a client HTTP connection open after the CICS application has finished using it.

STATUS
Optional Parameter

The enabled or disabled state of the URIMAP resource.

Values for the parameter are:
DISABLED
DISABLEDHOST
ENABLED

TCPIPSERVICE
Optional Parameter

The name of the TCPIPSERVICE resource that defines the inbound port to which the URIMAP resource relates.

TEMPLATENAME
Optional Parameter

The name of a CICS document template that forms the body of the static response that is sent to the HTTP request from the web client.

TRANSACTION
Optional Parameter

The name of an alias transaction that is to be used to run the user application that composes the HTTP response, or to start the pipeline.

USAGE
Optional Parameter

Specifies how the URIMAP resource is used.

Values for the parameter are:
ATOM
CLIENT
PIPELINE
SERVER

USERID
Optional Parameter

The user ID under which requests for the URIMAP resource are initially processed.

WEBSERVICE_NAME
Optional Parameter

The name of a WEBSERVICE resource associated with the URIMAP resource.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME

INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBUR gate, START_BROWSE_HOST function

The START_BROWSE_HOST function is used to begin a browse through the virtual host names in the Web domain.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

BROWSE_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

WBUR gate, START_BROWSE_URIMAP function

The START_BROWSE_URIMAP function is used to begin a browse through the URIMAP resources in the Web domain.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
CCNV_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
DUPLICATE_MAPPING
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
INVALID_CHARACTERSET
INVALID_HOSTCODEPAGE
INVALID_HOSTNAME
INVALID_PATHNAME
LOCATION_INVALID
NO_REDIRECTION_LOCATION
NOT_FOUND
NOT_POSSIBLE
SECURITY_FAILED
SSL_INACTIVE
URIMAP_ENABLED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

BROWSE_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Web domain's generic gates

Table 59 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 59. Web domain's generic gates

Gate	Trace	Functions	Format
DMDM	WB 0100 WB 0101	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
XMAC	WB 0600 WB 0601	INIT_XM_CLIENT BIND_XM_CLIENT TRANSACTION_HANG RELEASE_XM_CLIENT	XMAC
RLCB	WB 4700 WB 4707	CREATE DISCARD INQUIRE INQUIRE_BY_NAME SET	RLCB

In initialization, quiesce, and termination processing, the Web domain performs only internal routines.

Web domain's call-back gates

Table 60 summarizes the domain's call-back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 60. Web domain's call-back gates

Gate	Trace	Function	Format
WBXM	WB 0600 WB 0601	INIT_XM_CLIENT BIND_XM_CLIENT RELEASE_XM_CLIENT	XMAC

Modules

Module	Function
DFHWBAP	Handles the following requests: START_BROWSE READ_NEXT END_BROWSE GET_MESSAGE_BODY GET_HTTP_RESPONSE SEND_RESPONSE READ WRITE_HEADER INQUIRE QUERY_URIMAP
DFHWBAPF	Handles forms processing for: START_BROWSE READ_NEXT END_BROWSE READ
DFHWBCL	Functions for HTTP client processing.
DFHWBDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHWBQM	Domain subroutine which writes Web data to TS. Handles the following requests: PUT_QUEUE GET_QUEUE DELETE_QUEUE GET_TOKEN
DFHWBRP	Web domain recovery program.
DFHWBSR	Handles the following requests: SEND RECEIVE SEND_STATIC_RESPONSE PEEK
DFHWBUR	Functions for handling the URIMAP resource, including virtual hosts.

Module	Function
DFHWBXM	Handles the following requests: INIT_XM_CLIENT BIND_XM_CLIENT TRANSACTION_HANG RELEASE_XM_CLIENT

Exits

Three global user exit points are provided in CICS Web support for HTTP client requests.

XWBAUTH, HTTP client send exit
XWBAUTH is called during processing of an EXEC CICS WEB SEND or EXEC CICS WEB CONVERSE command. It allows you to specify basic authentication credentials (username and password) for a target server. XWBAUTH passes these to CICS on request, to create an Authorization header. The host name and path information are passed to the user exit, with an optional qualifying realm.

XWBOPEN, HTTP client open exit
XWBOPEN is called during processing of an EXEC CICS WEB OPEN command, which is used by an application program to open a connection with a server. It allows you to specify proxy servers that should be used for HTTP requests by CICS as an HTTP client, and to apply a security policy to the host name specified for those requests.

XWBSNDO, HTTP client send exit
XWBSNDO is called during processing of an EXEC CICS WEB SEND or EXEC CICS WEB CONVERSE command. It allows you to specify a security policy for HTTP requests, in particular for the path component of the request.

Web 2.0 domain (W2)

The Web 2.0 domain manages Atom feeds that CICS serves to web clients. The other actions of CICS as an HTTP server and as an HTTP client are managed by the web (WB) domain.

For more information about Atom feeds from CICS, see How CICS supports Atom feeds.

Web 2.0 domain's specific gates

The specific gates provide access for other domains to functions that are provided by the W2 domain.

Table 61. Web 2.0 Domain's specific gates

Gate	Trace	Function	XPI
W2AT	W2 0201	ADD_ATOMSERVICE	No
	W2 0202	ADD_REPLACE_ATOMSERVICE	No
	W2 0203	DELETE_ATOMSERVICE	No
	W2 0204	END_BROWSE_ATOMSERVICE	No
	W2 0205	GET_NEXT_ATOMSERVICE	No
	W2 0206	INITIALIZE_ATOMSERVICES	No
		INQUIRE_ATOMSERVICE	No
		SET_ATOMSERVICE	No
		START_BROWSE_ATOMSERVICE	No
W2W2	W2 0401	HANDLE_ATOM_REQUEST	No
	W2 0402	SET_PARAMETERS	No
	W2 0403		
	W2 0404		
	W2 0405		
	W2 0406		

W2AT gate, ADD_ATOMSERVICE function

The ADD_ATOMSERVICE function is used to add a new ATOMSERVICE resource into the Web 2.0 domain. If an ATOMSERVICE resource with the same name already exists, this function fails with reason code ATOMSERVICE_EXISTS.

Input Parameters

ATOM_TYPE

Type of Atom document associated with this ATOMSERVICE resource.

Values for the parameter are:

```
CATEGORY
COLLECTION
FEED
SERVICE
```

ATOMSERVICE

Name of the ATOMSERVICE resource to be installed.

BINDFILE

Optional Parameter

Name of the XSD bind file for this ATOMSERVICE resource.

CONFIGFILE

Optional Parameter

Name of the Atom configuration file for this ATOMSERVICE resource.

MESSAGE

Optional Parameter

Specifies whether installation messages will be issued.

Values for the parameter are:

```
NO
YES
```

RESOURCE_NAME

Optional Parameter

Name of the CICS resource associated with this ATOMSERVICE resource.

RESOURCE_SIGNATURE

Optional Parameter

The INSTALL resource signature for the new ATOMSERVICE resource.

RESOURCE_TYPE
Optional Parameter

The type of the CICS resource associated with this ATOMSERVICE resource.

Values for the parameter are:
FILE
PROGRAM
TSQUEUE

STATUS
Optional Parameter

Specifies the state in which the new ATOMSERVICE resource is installed.

Values for the parameter are:
DISABLED
ENABLED

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
ATOMSERVICE_ENABLED
ATOMSERVICE_EXISTS
BINDFILE_ERROR
BINDFILE_NOT_FOUND
BINDFILE_NOTAUTH
BROWSE_END
CONFIGFILE_NOT_FOUND
CONFIGFILE_NOTAUTH
CONFIGURATION_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
NOT_AUTH
NOT_FOUND
NOT_POSSIBLE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SIGNATURE

RESPONSE
Standard domain response values.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

W2AT gate, ADD_REPLACE_ATOMSERVICE function

The ADD_REPLACE_ATOMSERVICE function is used to add or replace an ATOMSERVICE resource in the Web 2.0 domain. If an ATOMSERVICE resource

with the same name already exists, it is replaced if it is disabled; otherwise, this function fails with reason code ATOMSERVICE_ENABLED.

Input Parameters

ATOM_TYPE

Type of Atom document associated with this ATOMSERVICE resource.

Values for the parameter are:

```
CATEGORY
COLLECTION
FEED
SERVICE
```

ATOMSERVICE

Name of the ATOMSERVICE resource to be installed.

RESOURCE_SIGNATURE

The INSTALL resource signature for the new ATOMSERVICE resource.

BINDFILE

Optional Parameter

Name of the XSD bind file for this ATOMSERVICE resource.

CONFIGFILE

Optional Parameter

Name of the Atom configuration file for this ATOMSERVICE resource.

MESSAGE

Optional Parameter

Specifies whether installation messages will be issued.

Values for the parameter are:

```
NO
YES
```

RESOURCE_NAME

Optional Parameter

Name of the CICS resource associated with this ATOMSERVICE resource.

RESOURCE_TYPE

Optional Parameter

The type of the CICS resource associated with this ATOMSERVICE resource.

Values for the parameter are:

```
FILE
PROGRAM
TSQUEUE
```

STATUS

Optional Parameter

Specifies the state in which the new ATOMSERVICE resource is installed.

Values for the parameter are:

```
DISABLED
ENABLED
```

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

ATOMSERVICE_ENABLED
ATOMSERVICE_EXISTS
BINDFILE_ERROR
BINDFILE_NOT_FOUND
BINDFILE_NOTAUTH
BROWSE_END
CONFIGFILE_NOT_FOUND
CONFIGFILE_NOTAUTH
CONFIGURATION_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
NOT_AUTH
NOT_FOUND
NOT_POSSIBLE

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_SIGNATURE

RESPONSE

Standard domain response values.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

W2AT gate, DELETE_ATOMSERVICE function

The DELETE_ATOMSERVICE function is used to delete an ATOMSERVICE resource from the Web 2.0 domain.

Input Parameters

ATOMSERVICE

Name of the ATOMSERVICE resource to be deleted.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

ATOMSERVICE_ENABLED
ATOMSERVICE_EXISTS
BINDFILE_ERROR
BINDFILE_NOT_FOUND
BINDFILE_NOTAUTH
BROWSE_END
CONFIGFILE_NOT_FOUND
CONFIGFILE_NOTAUTH
CONFIGURATION_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR

GETMAIN_FAILED
INVALID_BROWSE_TOKEN
NOT_AUTH
NOT_FOUND
NOT_POSSIBLE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SIGNATURE

RESPONSE
Standard domain response values.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

W2AT gate, END_BROWSE_ATOMSERVICE function

The END_BROWSE_ATOMSERVICE function is used to end a browse through the ATOMSERVICE resources in the Web 2.0 Domain.

Input Parameters

BROWSE_TOKEN
Token representing the current browse in progress.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
ATOMSERVICE_ENABLED
ATOMSERVICE_EXISTS
BINDFILE_ERROR
BINDFILE_NOT_FOUND
BINDFILE_NOTAUTH
BROWSE_END
CONFIGFILE_NOT_FOUND
CONFIGFILE_NOTAUTH
CONFIGURATION_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
NOT_AUTH
NOT_FOUND
NOT_POSSIBLE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SIGNATURE

RESPONSE
Standard domain response values.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

W2AT gate, GET_NEXT_ATOMSERVICE function

The GET_NEXT_ATOMSERVICE function is used to continue a browse through the ATOMSERVICE resources in the Web 2.0 Domain.

Input Parameters

BROWSE_TOKEN
Token representing the current browse in progress.

BINDFILE
Optional Parameter

Name of the XSD bind file for this ATOMSERVICE resource.

CONFIGFILE
Optional Parameter

Name of the Atom configuration file for this ATOMSERVICE resource.

RESOURCE_SIGNATURE
Optional Parameter

The INSTALL resource signature for the new ATOMSERVICE resource.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
ATOMSERVICE_ENABLED
ATOMSERVICE_EXISTS
BINDFILE_ERROR
BINDFILE_NOT_FOUND
BINDFILE_NOTAUTH
BROWSE_END
CONFIGFILE_NOT_FOUND
CONFIGFILE_NOTAUTH
CONFIGURATION_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
NOT_AUTH
NOT_FOUND
NOT_POSSIBLE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SIGNATURE

ATOMSERVICE
Name of the ATOMSERVICE resource located in the browse.

RESPONSE
Standard domain response values.

Values for the parameter are:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

ATOM_TYPE
Optional Parameter

Type of Atom document associated with this ATOMSERVICE resource.

Values for the parameter are:
- CATEGORY
- COLLECTION
- FEED
- SERVICE

RESOURCE_NAME
Optional Parameter

Name of the CICS resource associated with this ATOMSERVICE resource.

RESOURCE_TYPE
Optional Parameter

The type of the CICS resource associated with this ATOMSERVICE resource.

Values for the parameter are:
- FILE
- PROGRAM
- TSQUEUE

STATUS
Optional Parameter

Specifies the current state of the ATOMSERVICE resource.

Values for the parameter are:
- DISABLED
- ENABLED

W2AT gate, INITIALIZE_ATOMSERVICES function

The INITIALIZE_ATOMSERVICES function is used to initialize the Web 2.0 domain state required by the ATOMSERVICE support.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
- ABEND
- LOOP

The following values are returned when RESPONSE is EXCEPTION:
- ATOMSERVICE_ENABLED
- ATOMSERVICE_EXISTS
- BINDFILE_ERROR
- BINDFILE_NOT_FOUND
- BINDFILE_NOTAUTH
- BROWSE_END

CONFIGFILE_NOT_FOUND
CONFIGFILE_NOTAUTH
CONFIGURATION_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
NOT_AUTH
NOT_FOUND
NOT_POSSIBLE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SIGNATURE

RESPONSE
Standard domain response values.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

W2AT gate, INQUIRE_ATOMSERVICE function

The INQUIRE_ATOMSERVICE function is used to inquire on the attributes of an ATOMSERVICE resource.

Input Parameters

ATOMSERVICE
The INQUIRE_ATOMSERVICE function is used to inquire on the attributes of an ATOMSERVICE resource.

BINDFILE
Optional Parameter

Name of the XSD bind file for this ATOMSERVICE resource.

CONFIGFILE
Optional Parameter

Name of the Atom configuration file for this ATOMSERVICE resource.

RESOURCE_SIGNATURE
Optional Parameter

The INSTALL resource signature for the new ATOMSERVICE resource.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
ATOMSERVICE_ENABLED
ATOMSERVICE_EXISTS
BINDFILE_ERROR
BINDFILE_NOT_FOUND
BINDFILE_NOTAUTH

BROWSE_END
CONFIGFILE_NOT_FOUND
CONFIGFILE_NOTAUTH
CONFIGURATION_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
NOT_AUTH
NOT_FOUND
NOT_POSSIBLE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SIGNATURE

RESPONSE
Standard domain response values.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

ATOM_TYPE
Optional Parameter

Type of Atom document associated with this ATOMSERVICE resource.

Values for the parameter are:
CATEGORY
COLLECTION
FEED
SERVICE

RESOURCE_NAME
Optional Parameter

Name of the CICS resource associated with this ATOMSERVICE resource.

RESOURCE_TYPE
Optional Parameter

The type of the CICS resource associated with this ATOMSERVICE resource.

Values for the parameter are:
FILE
PROGRAM
TSQUEUE

STATUS
Optional Parameter

Specifies the current state of the ATOMSERVICE resource.

Values for the parameter are:
DISABLED
ENABLED

W2AT gate, SET_ATOMSERVICE function

The SET_ATOMSERVICE function is used to set the attributes of an ATOMSERVICE resource.

Input Parameters

ATOMSERVICE

Name of the ATOMSERVICE resource with the attributes that are being changed.

STATUS

Specifies the required state of the ATOMSERVICE resource.

Values for the parameter are:

DISABLED
ENABLED

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

ATOMSERVICE_ENABLED
ATOMSERVICE_EXISTS
BINDFILE_ERROR
BINDFILE_NOT_FOUND
BINDFILE_NOTAUTH
BROWSE_END
CONFIGFILE_NOT_FOUND
CONFIGFILE_NOTAUTH
CONFIGURATION_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
NOT_AUTH
NOT_FOUND
NOT_POSSIBLE

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION
INVALID_SIGNATURE

RESPONSE

Standard domain response values.

Values for the parameter are:

OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

W2AT gate, START_BROWSE_ATOMSERVICE function

The START_BROWSE_ATOMSERVICE function is used to start a browse through the ATOMSERVICE resources in the Web 2.0 Domain.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND

LOOP

The following values are returned when RESPONSE is EXCEPTION:
ATOMSERVICE_ENABLED
ATOMSERVICE_EXISTS
BINDFILE_ERROR
BINDFILE_NOT_FOUND
BINDFILE_NOTAUTH
BROWSE_END
CONFIGFILE_NOT_FOUND
CONFIGFILE_NOTAUTH
CONFIGURATION_ERROR
CONFLICTING_ATTRIBUTES
DIRECTORY_ERROR
GETMAIN_FAILED
INVALID_BROWSE_TOKEN
NOT_AUTH
NOT_FOUND
NOT_POSSIBLE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION
INVALID_SIGNATURE

BROWSE_TOKEN
Token representing the browse being started

RESPONSE
Standard domain response values.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

W2W2 gate, HANDLE_ATOM_REQUEST function

The HANDLE_ATOM_REQUEST function processes an inbound HTTP request for an Atom document. It examines the request and calls an appropriate response handling routine.

Input Parameters

CHECK_ACCESS
Optional Parameter

Specifies whether the authority of the user to access the ATOMSERVICE resource is to be checked.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
ABEND
DIRECTORY_ERROR
INITIALIZATION_ERROR

INVALID_FORMAT
INVALID_FUNCTION
LOOP
NON_WEB_TRANSACTION
NOT_FOUND

RESPONSE
Standard domain response values.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

W2W2 gate, SET_PARAMETERS function

The SET_PARAMETERS function specifies system initialization parameters for the domain.

Input Parameters

HOME_DIRECTORY
The CICS home directory in the Unix System Services file system, as specified by the **USSHOME** system initialization parameter.

Output Parameters

REASON
The values for the parameter are:
ABEND
DIRECTORY_ERROR
INITIALIZATION_ERROR
INVALID_FORMAT
INVALID_FUNCTION
LOOP
NON_WEB_TRANSACTION
NOT_FOUND

RESPONSE
Standard domain response values.

Values for the parameter are:
OK
EXCEPTION
DISASTER
INVALID
KERNERROR
PURGED

Modules

The W2 domain modules handle requests for Atom documents.

Module	Function
DFHW2A	Application program run for the CW2A transaction, which is the alias transaction for servicing Atom requests.
DFHW2AC	Reads and parses the Atom configuration file, as input for DFHW2AT.
DFHW2AS	Atom stringid generator. Called by XMLSS to tokenize text strings.

Module	Function
DFHW2AT	Manages the ATOMSERVICE resource.
DFHW2DM	Handles initialization and termination of the W2 domain.
DFHW2DUF	Dump formatter for the Web 2.0 domain.
DFHW2FD	Main feed document handler. Receives the Atom HTTP requests and forwards them to the appropriate service routine.
DFHW2FI	Atom service routine for CICS file requests.
DFHW2FR	Remote file handler. Communicates file requests to a File Owning Region.
DFHW2RP	ATOMSERVICE recovery program. Restores Atom feed support on CICS restart.
DFHW2SD	Atom Service Document handler. Returns Atom Publishing Protocol service documents and category documents.
DFHW2ST	Statistics manager.
DFHW2TRI	Trace interpretation routine.
DFHW2TS	Atom service routine for Temporary Storage requests.
DFHW2TT	Translate tables.
DFHW2UE	User exit manager.
DFHW2W2	Router module for Atom requests. Communicates between DFHW2A and DFHW2FD or DFHW2SD.

Exits

The Web 2.0 domain (W2) has no specific global user exit points. The general resource install and discard exit XRSINDI is called by the Web 2.0 domain to log the installation and discarding of ATOMSERVICE resource definitions.

Transaction manager domain (XM)

The transaction manager domain (also sometimes known as "transaction manager") provides transaction-related services.

The services provided by the domain are used to:

- Create tasks
- Terminate tasks
- Purge tasks
- Inquire on tasks
- Manage transaction definitions
- Manage tranclass definitions

The transaction manager domain also provides a transaction environment to enable other CICS components to implement transaction-related services.

Transaction manager domain's specific gates

The specific gates provide access for other domains to functions that are provided by the XM domain.

XMAT gate, ATTACH function

The ATTACH function of the XMAT gate is used to attach a new transaction.

Input Parameters

RETURN_NOT_FOUND
Indicates whether the attacher wants to receive the NOT_FOUND exception. Default is to attach CSAC in place of the requested transaction.

Values for the parameter are:
NO
YES

TPNAME
Alternative means of specifying the transaction identifier to attach.

TRANSACTION_ID
The transaction identifier to attach.

ATTACH_PARMS
Optional Parameter

Parameters to be passed to the attached transaction.

EXTERNAL_UOW_ID
Optional Parameter

An externally created unit-of-work identifier to be associated with the attached transaction.

FACILITY_TYPE
Optional Parameter

The type of principal facility to be associated with the attached transaction.

Values for the parameter are:
NONE
TERMINAL

PRIMARY_CLIENT_REQ_BLOCK
Optional Parameter

A data block containing information associated with the primary client.

PRIMARY_CLIENT_TYPE
Optional Parameter

The type of client for which the transaction is being attached.

Values for the parameter are:
APPC_SESSION
BRIDGE
IIRR
IP_ECI
LU61_SESSION
MRO_SESSION
NONE
RRS_UR
RZ_INSTORE_TRPORT
SCHEDULER
SOCKET
START
START_TERMINAL
TERMINAL
TRANDATA
WEB
XM_RUN_TRANSACTION

PRIORITY
Optional Parameter

Combined user and terminal priority to be added to that of the transaction definition to determine the total priority of the attached transaction.

RESTART_COUNT
Optional Parameter

If the attach is for a restarted transaction then this count indicates the number of this restart attempt.

START_ATTACH
Optional Parameter

Indicates if the attach is in response to a START command.

Values for the parameter are:
YES

START_CODE
Optional Parameter

Indicates the reason for the attach.

Values for the parameter are:
C
DF
QD
S
SD
SZ
T
TT

SUSPEND
Optional Parameter

Indicates whether the attacher is willing to suspend during the attach.

Values for the parameter are:
NO
YES

SYSTEM_ATTACH
Optional Parameter

Indicates whether the transaction should be attached as a system transaction.

Values for the parameter are:
YES

TD_TOKEN
Optional Parameter

Token identifying a TDQ to be associated with the transaction.

TF_TOKEN
Optional Parameter

Token identifying a terminal to be associated with the transaction.

TOTAL_PRIORITY
Optional Parameter

The overriding priority to be associated with the attached transaction.

TRANSACTION_GROUP
Optional Parameter

Indicates whether the newly attached transaction should be in the same monitoring group as the current transaction.

Values for the parameter are:
NEW

SAME

US_TOKEN
Optional Parameter

Token identifying a user to be associated with the transaction.

USE_DTRTRAN
Optional Parameter

If the named transaction-id or tpname cannot be found then indicates whether the DTRTRAN, if installed, should be used instead.

Values for the parameter are:
NO
YES

Output Parameters

CURRENT_ACTIVE
Optional Parameter

Is the number of transactions that are currently active within the region.

CURRENT_QUEUED
Optional Parameter

Is the number of transactions that are currently queued waiting on MXT.

REASON
The values for the parameter are:
ABEND
DISABLED
INSUFFICIENT_STORAGE
INVALID_FUNCTION
INVALID_RETURN_NOT_FOUND
INVALID_START_CODE
INVALID_SYSTEM_ATTACH
LOOP
NOT_ENABLED_FOR_SHUTDOWN
NOT_FOUND
STATE_SYSTEM_ATTACH
STATE_TASKDATAKEY
STATE_TASKDATALOC

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANDEF_TOKEN
Optional Parameter

The token representing the returned transaction definition.

TRANNUM
Optional Parameter

Is the transaction number assigned to the newly attached transaction.

TRANSACTION_TOKEN
Optional Parameter

Is the token identifying the newly attached transaction.

XMAT gate, REATTACH function

A variation of the ATTACH function that is used by Recovery Manager to attach a task that will unshunt a specific UOW.

Input Parameters

FACILITY_TYPE
Optional Parameter

The type of principal facility to be associated with the transaction.

Values for the parameter are:
NONE

RETURN_NOT_FOUND
Indicates whether the attacher wants to receive the NOT_FOUND exception. Default is to attach CSAC in place of the requested transaction.

Values for the parameter are:
NO
YES

START_CODE
Optional Parameter

Indicates the reason for the attach.

Values for the parameter are:
C

TRANSACTION_ID
The transaction identifier to attach.

UOW_TOKEN
A token representing the unit of work that is to be unshunted.

PRIORITY
Optional Parameter

Combined user and terminal priority to be added to that of the transaction definition to determine the total priority of the attached transaction.

SUSPEND
Optional Parameter

Indicates whether the attacher is willing to suspend during the attach.

Values for the parameter are:
NO
YES

SYSTEM_ATTACH
Optional Parameter

Indicates whether the transaction should be attached as a system transaction.

Values for the parameter are:
YES

TCLASS
Optional Parameter

The transaction class of the attched transaction.

Values for the parameter are:
NONE

TOTAL_PRIORITY
Optional Parameter

The overriding priority to be associated with the attached transaction.

TRANSACTION_GROUP
Optional Parameter

Indicates whether the newly attached transaction should be in the same monitoring group as the current transaction.

Values for the parameter are:
NEW
SAME

Output Parameters

REASON
The values for the parameter are:
ABEND
INSUFFICIENT_STORAGE
INVALID_FUNCTION
LOOP
NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANNUM
Optional Parameter

Is the transaction number assigned to the newly attached transaction.

TRANSACTION_TOKEN
Optional Parameter

Is the token identifying the newly attached transaction.

XMAT gate, BUILD_TRANSACTION function

The BUILD_TRANSACTION function of the XMAT gate builds a transaction environment.

Input Parameters

RETURN_NOT_FOUND
Indicates whether the attacher wants to receive the NOT_FOUND exception. Default is to attach CSAC in place of the requested transaction.

Values for the parameter are:
YES

TPNAME
Alternative means of specifying the transaction identifier to attach.

TRANSACTION_ID
The transaction identifier to attach.

ATTACH_PARMS
Optional Parameter

Parameters to be passed to the attached transaction.

EXTERNAL_UOW_ID
Optional Parameter

An externally created unit-of-work identifier to be associated with the attached transaction.

FACILITY_TYPE
Optional Parameter

The type of principal facility to be associated with the attached transaction.

Values for the parameter are:
NONE
TERMINAL

PRIMARY_CLIENT_REQ_BLOCK
Optional Parameter

A data block containing information associated with the primary client.

PRIMARY_CLIENT_TYPE
Optional Parameter

The type of client for which the transaction is being attached.

Values for the parameter are:
APPC_SESSION
BRIDGE
IIRR
IP_ECI
LU61_SESSION
MRO_SESSION
NONE
RRS_UR
RZ_INSTORE_TRPORT
SCHEDULER
SOCKET
START
START_TERMINAL
TERMINAL
TRANDATA
WEB
XM_RUN_TRANSACTION

PRIORITY
Optional Parameter

Combined user and terminal priority to be added to that of the transaction definition to determine the total priority of the attached transaction.

RESTART_COUNT
Optional Parameter

If the attach is for a restarted transaction then this count indicates the number of this restart attempt.

START_ATTACH
Optional Parameter

Indicates if the attach is in response to a START command.

Values for the parameter are:
YES

START_CODE
Optional Parameter

Indicates the reason for the attach.

Values for the parameter are:
C
DF
QD
S
SD
SZ
T
TT

SUSPEND
Optional Parameter

Indicates whether the attacher is willing to suspend during the attach.

Values for the parameter are:
NO

YES

SYSTEM_ATTACH
Optional Parameter

Indicates whether the transaction should be attached as a system transaction.

Values for the parameter are:
YES

TD_TOKEN
Optional Parameter

Token identifying a TDQ to be associated with the transaction.

TF_TOKEN
Optional Parameter

Token identifying a terminal to be associated with the transaction.

TOTAL_PRIORITY
Optional Parameter

The overriding priority to be associated with the attached transaction.

TRANSACTION_GROUP
Optional Parameter

Indicates whether the newly attached transaction should be in the same monitoring group as the current transaction.

Values for the parameter are:
NEW
SAME

US_TOKEN
Optional Parameter

Token identifying a user to be associated with the transaction.

USE_DTRTRAN
Optional Parameter

If the named transaction-id or tpname cannot be found then indicates whether the DTRTRAN, if installed, should be used instead.

Values for the parameter are:
NO
YES

Output Parameters

CURRENT_ACTIVE
Optional Parameter

Is the number of transactions that are currently active within the region.

CURRENT_QUEUED
Optional Parameter

Is the number of transactions that are currently queued waiting on MXT.

REASON
The values for the parameter are:
ABEND
DISABLED
INSUFFICIENT_STORAGE
INVALID_FUNCTION
INVALID_RETURN_NOT_FOUND
INVALID_START_CODE
INVALID_SYSTEM_ATTACH
LOOP

NOT_ENABLED_FOR_SHUTDOWN
NOT_FOUND
STATE_SYSTEM_ATTACH
STATE_TASKDATAKEY
STATE_TASKDATALOC

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANDEF_TOKEN
Optional Parameter

The token representing the returned transaction definition.

TRANNUM
Optional Parameter

Is the transaction number assigned to the newly attached transaction.

TRANSACTION_TOKEN
Optional Parameter

Is the token identifying the newly attached transaction.

XMAT gate, TERMINATE_TRANSACTION function

The TERMINATE_TRANSACTION function of the XMAT gate terminates the transaction environment that was built by BUILD_TRANSACTION.

Input Parameters

TERMINATION_TYPE
The type of termination.

Values for the parameter are:
NORMAL
ABNORMAL

Output Parameters

REASON
The values for the parameter are:
CHANGE_MODE_ERROR
ABEND
LOOP
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMBD gate, END_BROWSE_TRANDEF function

The END_BROWSE_TRANDEF function of the XMBD gate is used to terminate a browse of installed transaction definitions.

Input Parameters

BROWSE_TOKEN
Token identifying this browse of the transaction definitions.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMBD gate, GET_NEXT_TRANDEF function

The GET_NEXT_TRANDEF function of the XMBD gate is used to return information about the next transaction definition in the browse.

Input Parameters

BROWSE_TOKEN
Token identifying this browse of the transaction definitions.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:
BROWSE_END_TRANDEF

The following values are returned when RESPONSE is INVALID:
INVALID_BROWSE_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BREXIT
Optional Parameter

The name of the default bridge exit associated with the transaction.

CMDSEC
Optional Parameter

Whether command security checking is active.

Values for the parameter are as follows:
NO
YES

CONFDATA
Optional Parameter

The value of the CONFDATA attribute specified in the transaction definition.

Values for the parameter are as follows:
NO
YES

DTIMEOUT
Optional Parameter

The deadlock timeout value for the transaction.

DUMP
Optional Parameter

Whether transaction dumps are to be taken.

Values for the parameter are as follows:
NO
YES

DYNAMIC
Optional Parameter

Whether the transaction is defined to be dynamic.

Values for the parameter are as follows:
NO
YES

INDOUBT
Optional Parameter

The action to take if work performed by the transaction becomes indoubt.

Values for the parameter are as follows:
BACKOUT
COMMIT

INDOUBT_WAIT
Optional Parameter

Indicates whether an indoubt unit of work (UOW) is to wait, pending recovery from a failure that occurs after the UOW has entered the indoubt state.

Values for the parameter are as follows:
NO
YES

INDOUBT_WAIT_TIME
Optional Parameter

Indicates how long the transaction is to wait before taking an arbitrary decision about an indoubt unit of work.

INITIAL_PROGRAM
Optional Parameter

Initial program of transaction.

ISOLATE
Optional Parameter

Whether the transaction runs in its own subspace.

Values for the parameter are as follows:
NO
YES

LOCAL_QUEUING
Optional Parameter

Whether the transaction is eligible to queue locally when it is started on the remote system.

Values for the parameter are as follows:
NO
YES

OTSTIMEOUT
Optional Parameter

The value of the OTSTIMEOUT attribute in the transaction definition.

PARTITIONSET
Optional Parameter

The partitionset defined for the transaction.

Values for the parameter are as follows:
KEEP
NAMED
NONE
OWN

PARTITIONSET_NAME
Optional Parameter

The name of the user defined partitionset used by the transaction.

PROFILE_NAME
Optional Parameter

Profile of transaction.

REMOTE
Optional Parameter

Whether the transaction is remote.

Values for the parameter are as follows:
NO
YES

REMOTE_NAME
Optional Parameter

The name of a remote transaction on the remote system.

REMOTE_SYSTEM
Optional Parameter

The system that a remote transaction is to be routed to.

RESSEC
Optional Parameter

Whether resource security checking is active.

Values for the parameter are as follows:
NO
YES

RESTART
Optional Parameter

Whether the transaction is restartable.

Values for the parameter are as follows:
NO
YES

ROUTABLE_STATUS
Optional Parameter

Specifies whether, if the transaction is the subject of an eligible EXEC CICS START command, it will be routed using the enhanced routing method.

Values for the parameter are as follows:
NOTROUTABLE
ROUTABLE

RUNAWAY_LIMIT
Optional Parameter

The runaway limit associated with the transaction.

SHUTDOWN
Optional Parameter

Whether the transaction can be run during shutdown.

Values for the parameter are as follows:
DISABLED
ENABLED

SPURGE
Optional Parameter

Whether the transaction is system-purgeable.

Values for the parameter are as follows:

NO
YES

STATUS
Optional Parameter

The status of the transaction.

Values for the parameter are as follows:
DISABLED
ENABLED

STORAGE_CLEAR
Optional Parameter

Whether task-lifetime storage is to be cleared before it is released by using a freemain request.

Values for the parameter are as follows:
NO
YES

STORAGE_FREEZE
Optional Parameter

Whether storage freeze is on for the transaction.

Values for the parameter are as follows:
NO
YES

SYSTEM_RUNAWAY
Optional Parameter

Whether the transaction uses the default system runaway limit.

Values for the parameter are as follows:
NO
YES

TASKDATAKEY
Optional Parameter

The storage key that task-lifetime storage is allocated in.

Values for the parameter are as follows:
CICS
USER

TASKDATALOC
Optional Parameter

The location of task-lifetime storage.

Values for the parameter are as follows:
ANY
BELOW

TCLASS
Optional Parameter

Whether the transaction belongs to a tclass.

TCLASS_NAME
Optional Parameter

The name of the tclass that the transaction belongs to.

TPURGE
Optional Parameter

Whether the transaction can be purged after a terminal error.

Values for the parameter are as follows:
NO
YES

TRACE
Optional Parameter

The level of tracing associated with the transaction.

Values for the parameter are as follows:
SPECIAL
STANDARD
SUPPRESSED

TRAN_PRIORITY
Optional Parameter

Transaction priority

TRAN_ROUTING_PROFILE
Optional Parameter

Profile to be used to route a remote transaction to a remote system.

TRANSACTION_ID
Optional Parameter

Transaction identifier

TWASIZE
Optional Parameter

Size of Transaction Work Area.

XMBD gate, START_BROWSE_TRANDEF function

The START_BROWSE_TRANDEF function of the XMBD gate is used to initiate a browse of installed transaction definitions.

Input Parameters

START_AT
Optional Parameter

Identifies a transaction identifier that the browse is to start at.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

BROWSE_TOKEN
Token identifying this transaction definition browse.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMCL gate, ADD_REPLACE_TCLASS function

The ADD_REPLACE_TCLASS function of the XMCL gate is used to install a tclass definition.

Input Parameters

MAX_ACTIVE
The max-active limit of the tclass.

TCLASS_NAME
The name of the tclass.

PURGE_THRESHOLD
Optional Parameter

The purge-threshold limit of the tclass.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
LOGIC_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_MAX_ACTIVE
INVALID_PURGE_THRESHOLD
INVALID_TCLASS_NAME
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TCLASS_TOKEN

Optional Parameter

Token identifying the tclass.

XMCL gate, ADD_TCLASS function

The ADD_TCLASS function of the XMCL gate is used to add an internal tclass definition.

Input Parameters

MAX_ACTIVE

The max-active limit of the tclass.

PURGE_THRESHOLD

Optional Parameter

The purge-threshold limit of the tclass.

TCLASS_NAME

Optional Parameter

The name of the tclass.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
LOGIC_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
DUPLICATE_TCLASS_NAME
INVALID_MAX_ACTIVE
INVALID_PURGE_THRESHOLD
INVALID_TCLASS_NAME
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TCLASS_TOKEN

Token identifying the tclass.

XMCL gate, DELETE_TCLASS function

The DELETE_TCLASS function of the XMCL gate is used to discard an installed tclass definition.

Input Parameters

TCLASS_NAME

The name of the tclass.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:

TCLASS_BUSY
UNKNOWN_TCLASS

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMCL gate, DEREGISTER_TCLASS_USAGE function

The DEREGISTER_TCLASS_USAGE function of the XMCL gate is used to deregister usage of a tclass by a transaction definition.

Input Parameters

TCLASS_TOKEN

Token identifying tclass being inquired upon.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOGIC_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_TCLASS_TOKEN
NOT_IN_USE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMCL gate, END_BROWSE_TCLASS function

The END_BROWSE_TCLASS function of the XMCL gate is used to terminate a browse of installed tclass definitions.

Input Parameters

BROWSE_TOKEN

Token identifying this browse of the transaction definitions.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOGIC_ERROR

The following values are returned when RESPONSE is INVALID:

INVALID_BROWSE_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMCL gate, GET_NEXT_TCLASS function

The GET_NEXT_TCLASS function of the XMCL gate is used to return information about the next tclass definition in the browse.

Input Parameters

BROWSE_TOKEN

Token identifying this browse of the transaction definitions.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END_TCLASS

The following values are returned when RESPONSE is INVALID:

INVALID_BROWSE_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CURRENT_ACTIVE

Optional Parameter

The number of active transactions in the tclass.

CURRENT_QUEUED

Optional Parameter

The number of queuing transactions in the tclass.

MAX_ACTIVE

Optional Parameter

The max-active limit of the tclass.

PURGE_THRESHOLD

Optional Parameter

The purge-threshold limit of the tclass.

TCLASS_NAME

Optional Parameter

The name of the tclass that the transaction belongs to.

XMCL gate, INQUIRE_ALL_TCLASSES function

The INQUIRE_ALL_TCLASSES function of the XMCL gate is used to inquire about the current state of all the tclasses in the system.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOGIC_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TOTAL_ACTIVE

Optional Parameter

The number of transactions active in a tclass.

TOTAL_QUEUED

Optional Parameter

The number of transactions queueing for a tclass.

XMCL gate, INQUIRE_TCLASS function

The INQUIRE_TCLASS function of the XMCL gate is used to inquire upon a tclass.

Input Parameters

INQ_TCLASS_NAME

The name of the tclass being inquired upon.

TCLASS_TOKEN
Token identifying tclass being inquired upon.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
`LOGIC_ERROR`

The following values are returned when RESPONSE is EXCEPTION:
`UNKNOWN_TCLASS`

The following values are returned when RESPONSE is INVALID:
`INVALID_TCLASS_TOKEN`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CURRENT_ACTIVE
Optional Parameter

The number of active transactions in the tclass.

CURRENT_QUEUED
Optional Parameter

The number of queuing transactions in the tclass.

MAX_ACTIVE
Optional Parameter

The max-active limit of the tclass.

PURGE_THRESHOLD
Optional Parameter

The purge-threshold limit of the tclass.

TCLASS_NAME
Optional Parameter

The name of the tclass that the transaction belongs to.

XMCL gate, LOCATE_AND_LOCK_TCLASS function

The LOCATE_AND_LOCK_TCLASS function of the XMCL gate is used to locate a named tclass and lock it against delete.

Input Parameters

TCLASS_NAME
The name of the tclass.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
`LOGIC_ERROR`

The following values are returned when RESPONSE is EXCEPTION:
`UNKNOWN_TCLASS`

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TCLASS_TOKEN
Token identifying the tclass.

XMCL gate, REGISTER_TCLASS_USAGE function

The REGISTER_TCLASS_USAGE function of the XMCL gate is used to register usage of a tclass by a transaction definition.

Input Parameters

TCLASS_NAME
The name of the tclass.

UNKNOWN_ACTION
Specifies the action to perform if the TCLASS hasn't been installed by the user.

Values for the parameter are:
CREATE
ERROR

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:
UNKNOWN_TCLASS

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TCLASS_TOKEN
Token identifying the tclass.

XMCL gate, SET_TCLASS function

The SET_TCLASS function of the XMCL gate is used to modify a tclass definition.

Input Parameters

TCLASS_NAME
The name of the tclass.

TCLASS_TOKEN
Token identifying tclass being inquired upon.

MAX_ACTIVE
Optional Parameter

The max-active limit of the tclass.

PURGE_THRESHOLD
Optional Parameter

The purge-threshold limit of the tclass.

RESET_STATISTICS
Optional Parameter

Indicates whether the statistics for the tclass are to be reset.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:
INVALID_MAX_ACTIVE

INVALID_PURGE_THRESHOLD
UNKNOWN_TCLASS

The following values are returned when RESPONSE is INVALID:
INVALID_TCLASS_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMCL gate, START_BROWSE_TCLASS function

The START_BROWSE_TCLASS function of the XMCL gate is used to initiate a browse of installed tclass definitions.

Input Parameters

START_AT
Optional Parameter

Identifies a transaction identifier that the browse is to start at.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

BROWSE_TOKEN
Token identifying this transaction definition browse.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMCL gate, UNLOCK_TCLASS function

The UNLOCK_TCLASS function of the XMCL gate is used to unlock a previously locked tclass.

Input Parameters

TCLASS_TOKEN
Token identifying tclass being inquired upon.

XM_LOCK_HELD
Optional Parameter

A binary parameter that indicates whether the caller already holds the transaction manager lock.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_TCLASS_TOKEN
NOT_LOCKED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMDD gate, DELETE_TRANDEF function

The DELETE_TRANDEF function of the XMDD gate is used to discard an installed transaction definition.

Input Parameters

TRANSACTION_ID

The transaction identifier to attach.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
LOGIC_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
AID_PENDING
ICE_PENDING
SIT_PARAMETER
UNKNOWN_TRANSACTION_ID
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMER gate, ABEND_TRANSACTION function

The ABEND_TRANSACTION function of the XMER gate is used abend a transaction whose attach has failed.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
INVALID_FUNCTION
LOOP
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMER gate, INQUIRE_DEFERRED_ABEND function

The INQUIRE_DEFERRED_ABEND function of the XMER gate is used to retrieve the abend that is to be issued for the transaction whose attach has failed.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
DEFERRED_ABEND_NOT_FOUND
INVALID_FUNCTION
LOOP
```

DEFERRED_ABEND_CODE

The abend code.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANSACTION_DUMP

Optional Parameter

Indicates whether a transaction dump is to be taken for the abend.

Values for the parameter are:

NO
YES

XMER gate, INQUIRE_DEFERRED_MESSAGE function

The INQUIRE_DEFERRED_MESSAGE function of the XMER gate is used to retrieve the message that is to be issued which will indicate the cause of a transaction attach failure.

Output Parameters

REASON

The values for the parameter are:

ABEND
INVALID_FUNCTION
LOOP
MESSAGE_NOT_FOUND

MESSAGE

The message that is to be issued.

Values for the parameter are:

ALL_SESSIONS_BUSY
CONSOLE_AUTOINSTALL_FAILED
CONSOLE_AUTOINSTALL_REJECT
CONSOLE_NOT_DEFINED
CONSOLE_SIGNON_FAILED
CONV_RESTART_REQUESTED
DBA_NOT_SUPPORTED
INVALID_ASIF_LENGTH
INVALID_ATTACH_PARAMETER
INVALID_CONV_TYPE
INVALID_FMH_LENGTH
INVALID_SYNC_LEVEL
INVALID_TERMINAL_FOR_TRANS
INVALID_UOW_IN_ATTACH
IO_ERROR_DURING_WRITE
LAST_MESSAGE
NULL_MESSAGE
PROFILE_UNAVAILABLE
PROGRAM_UNAVAILABLE
REMOTE_CONN_OOS
REMOTE_CONN_OOS_SYS_CHGD
SEC_VIOLATION_DETECTED
SECURITY_NOT_VALID
SECURITY_PROTOCOL_ERROR
SYNC_LEVEL_NOT_SUPPORTED
TRANID_NOT_FOUND
TRANSACTION_DISABLED
TRANSACTION_REMOTE
TXN_UNAVAIL_DURING_QUIESCE
UNRECOGNIZED_PIP
USER_NOT_AUTHORISED
XRF_RECOVERY_NOT_COMPLETE
ZNAC_DETECTED_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMER gate, REPORT_MESSAGE function

The REPORT_MESSAGE function of the XMER gate is used send a deferred message if the attach of a transaction has failed.

Input Parameters

MESSAGE

The message that is to be issued.

Values for the parameter are:

ALL_SESSIONS_BUSY
CONSOLE_AUTOINSTALL_FAILED
CONSOLE_AUTOINSTALL_REJECT
CONSOLE_NOT_DEFINED
CONSOLE_SIGNON_FAILED
CONV_RESTART_REQUESTED
DBA_NOT_SUPPORTED
INVALID_ASIF_LENGTH
INVALID_ATTACH_PARAMETER
INVALID_CONV_TYPE
INVALID_FMH_LENGTH
INVALID_SYNC_LEVEL
INVALID_TERMINAL_FOR_TRANS
INVALID_UOW_IN_ATTACH
IO_ERROR_DURING_WRITE
LAST_MESSAGE
NULL_MESSAGE
PROFILE_UNAVAILABLE
PROGRAM_UNAVAILABLE
REMOTE_CONN_OOS
REMOTE_CONN_OOS_SYS_CHGD
SEC_VIOLATION_DETECTED
SECURITY_NOT_VALID
SECURITY_PROTOCOL_ERROR
SYNC_LEVEL_NOT_SUPPORTED
TRANID_NOT_FOUND
TRANSACTION_DISABLED
TRANSACTION_REMOTE
TXN_UNAVAIL_DURING_QUIESCE
UNRECOGNIZED_PIP
USER_NOT_AUTHORISED
XRF_RECOVERY_NOT_COMPLETE
ZNAC_DETECTED_ERROR

Output Parameters

REASON

The values for the parameter are:

ABEND
INVALID_FUNCTION
LOOP
TRANSACTION_ABEND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMER gate, SET_DEFERRED_ABEND function

The SET_DEFERRED_ABEND function of the XMER gate is used to schedule an abend to be issued if the attach of a transaction fails.

Input Parameters

DEFERRED_ABEND_CODE

The abend code that is to be used.

TRANSACTION_DUMP

Optional Parameter

Indicates whether a transaction dump is to be taken for the abend.

Values for the parameter are:

NO
YES

TRANSACTION_TOKEN

Optional Parameter

Optional token to identify the transaction that the message is to be sent to. Defaults to the current transaction.

Output Parameters

REASON

The values for the parameter are:

ABEND
DEFERRED_ABEND_ALREADY_SET
INVALID_ABEND_CODE
INVALID_FUNCTION
INVALID_TRANSACTION_TOKEN
LOOP
MESSAGE_ALREADY_SET

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMER gate, SET_DEFERRED_MESSAGE function

The SET_DEFERRED_MESSAGE function of the XMER gate is used to store a message to be issued if the attach of a transaction fails.

Input Parameters

MESSAGE

The message that is to be issued.

Values for the parameter are:

ALL_SESSIONS_BUSY
CONSOLE_AUTOINSTALL_FAILED
CONSOLE_AUTOINSTALL_REJECT
CONSOLE_NOT_DEFINED
CONSOLE_SIGNON_FAILED
CONV_RESTART_REQUESTED
DBA_NOT_SUPPORTED
INVALID_ASIF_LENGTH
INVALID_ATTACH_PARAMETER
INVALID_CONV_TYPE
INVALID_FMH_LENGTH
INVALID_SYNC_LEVEL
INVALID_TERMINAL_FOR_TRANS
INVALID_UOW_IN_ATTACH

IO_ERROR_DURING_WRITE
LAST_MESSAGE
NULL_MESSAGE
PROFILE_UNAVAILABLE
PROGRAM_UNAVAILABLE
REMOTE_CONN_OOS
REMOTE_CONN_OOS_SYS_CHGD
SEC_VIOLATION_DETECTED
SECURITY_NOT_VALID
SECURITY_PROTOCOL_ERROR
SYNC_LEVEL_NOT_SUPPORTED
TRANID_NOT_FOUND
TRANSACTION_DISABLED
TRANSACTION_REMOTE
TXN_UNAVAIL_DURING_QUIESCE
UNRECOGNIZED_PIP
USER_NOT_AUTHORISED
XRF_RECOVERY_NOT_COMPLETE
ZNAC_DETECTED_ERROR

TRANSACTION_TOKEN

Optional Parameter

Optional token to identify the transaction that the message is to be sent to. Defaults to the current transaction.

Output Parameters

REASON

The values for the parameter are:

ABEND
DEFERRED_ABEND_ALREADY_SET
INVALID_FUNCTION
INVALID_TRANSACTION_TOKEN
LOOP
MESSAGE_ALREADY_SET

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMFD gate, FIND_PROFILE function

The FIND_PROFILE function of the XMFD gate is used to check whether the given profile is in use by a transaction definition.

Input Parameters

PROFILE_NAME

The profile that is to be found.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:

PROFILE_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANSACTION_ID
Optional Parameter

Transaction identifier

XMIQ gate, END_BROWSE_TRANSACTION function

The END_BROWSE_TRANSACTION function of the XMIQ gate is used to terminate a browse of all transactions in the system.

Input Parameters

BROWSE_TOKEN
Token identifying this browse of the transaction definitions.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BROWSE_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMIQ gate, END_BROWSE_TXN_TOKEN function

The END_BROWSE_TXN_TOKEN function of the XMIQ gate is used to terminate a browse of transaction tokens.

Input Parameters

BROWSE_TOKEN
Token identifying this browse of the transaction definitions.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BROWSE_TOKEN
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMIQ gate, GET_NEXT_TRANSACTION function

The GET_NEXT_TRANSACTION function of the XMIQ gate is used to inquire upon the next transaction in a transaction browse.

Input Parameters

BROWSE_TOKEN
Token identifying this browse of the transaction definitions.

ATTACH_PARMS
Optional Parameter

Parameters to be passed to the attached transaction.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:

BROWSE_END

The following values are returned when RESPONSE is INVALID:

INVALID_BROWSE_TOKEN

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ATTACH_TIME

Optional Parameter

The time when the transaction was attached.

CICS_UOW_ID

Optional Parameter

The CICS Unit Of Work Identifier associated with the transaction.

CONFDATA

Optional Parameter

The value of the CONFDATA attribute specified in the transaction definition.

Values for the parameter are:

NO
YES

DS_TASK_TOKEN

Optional Parameter

A token that identifies the dispatcher task associated with the transaction.

DTIMEOUT

Optional Parameter

The deadlock timeout value for the transaction.

DYNAMIC

Optional Parameter

Whether the transaction is defined to be dynamic.

Values for the parameter are:

NO
YES

ENTRYPOINT_RES_TOKEN

Optional Parameter

An 8-byte token that represents the application entry point.

FACILITY_NAME

Optional Parameter

The name of the principal facility associated with the transaction.

FACILITY_TOKEN

Optional Parameter

A token that represents the principal facility associated with the transaction.

FACILITY_TYPE

Optional Parameter

The type of the principal facility associated with the transaction.

Values for the parameter are:

IPECI
NONE
START
TD
TERMINAL

INITIAL_PROGRAM
Optional Parameter

Initial program of transaction.

NETNAME
Optional Parameter

The network name of a terminal principal facility.

OPERATION_NAME
Optional Parameter

The name of the application operation for which the TRANSACTION is declared as an application entry point.

ORIGINAL_TRANSACTION_ID
Optional Parameter

The transid that was used to attach the transaction.

OUT_TRANSACTION_TOKEN
Optional Parameter

The token that represents this transaction.

PHASE
Optional Parameter

The phase of the transaction.

Values for the parameter are:

BIND
INIT
PRE_INIT
TERM

PRIMARY_CLIENT_TOKEN
Optional Parameter

A token representing the client for which the transaction was attached.

PRIMARY_CLIENT_TYPE
Optional Parameter

The type of client for which the transaction was attached.

Values for the parameter are:

APPC_SESSION
BRIDGE
IIRR
IP_ECI
LU61_SESSION
MRO_SESSION
NONE
RRS_UR
RZ_INSTORE_TRPORT
SCHEDULER
SOCKET
START
START_TERMINAL
TERMINAL

TRANDATA
WEB
XM_RUN_TRANSACTION

RE_ATTACHED_TRANSACTION
Optional Parameter

Indicates if the transaction was reattached.

Values for the parameter are:
NO
YES

REMOTE
Optional Parameter

Whether the transaction is remote.

Values for the parameter are:
NO
YES

REMOTE_NAME
Optional Parameter

The name of a remote transaction on the remote system.

REMOTE_SYSTEM
Optional Parameter

The system that a remote transaction is to be routed to.

RESOURCE_NAME
Optional Parameter

The name of a resource that a suspended transaction is waiting for.

RESOURCE_TYPE
Optional Parameter

The type of resource that a suspended transaction is waiting for.

RESTART
Optional Parameter

Whether the transaction is restartable.

Values for the parameter are:
NO
YES

RESTART_COUNT
Optional Parameter

Contains the number of times this transaction instance has been restarted.

SPURGE
Optional Parameter

Whether the transaction is system-purgeable.

Values for the parameter are:
NO
YES

START_CODE
Optional Parameter

Indicates the reason for the attach of the transaction.

Values for the parameter are:
C
DF

QD
S
SD
SZ
T
TT

STATUS
Optional Parameter

The status of the transaction.

Values for the parameter are:
READY
RUNNING
SUSPENDED

SUSPEND_TIME
Optional Parameter

Contains the length of time that the transaction has currently been suspended for.

SYSTEM_TRANSACTION
Optional Parameter

Whether the transaction has been attached by CICS.

Values for the parameter are:
NO
YES

TASK_PRIORITY
Optional Parameter

The combined priority of the transaction.

TCLASS
Optional Parameter

Whether the transaction belongs to a tclass.

TCLASS_NAME
Optional Parameter

The name of the tclass that the transaction belongs to.

TPURGE
Optional Parameter

Whether the transaction can be purged after a terminal error.

Values for the parameter are:
NO
YES

TRAN_PRIORITY
Optional Parameter

Transaction priority

TRAN_ROUTING_PROFILE
Optional Parameter

Profile to be used to route a remote transaction to a remote system.

TRANDEF_TOKEN
Optional Parameter

The token representing the returned transaction definition.

TRANNUM
Optional Parameter

Is the transaction number assigned to the newly attached transaction.

TRANSACTION_GROUP_ID
Optional Parameter

The identifier of the transaction's monitoring group.

TRANSACTION_ID
Optional Parameter

Transaction identifier

USERID
Optional Parameter

The userid of the user associated with the transaction.

XMIQ gate, GET_NEXT_TXN_TOKEN function

The GET_NEXT_TXN_TOKEN function of the XMIQ gate is used to return the transaction token that is associated with the next transaction in the system.

Input Parameters

BROWSE_TOKEN
A token that identifies this browse of the transaction definitions.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
BROWSE_END
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_BROWSE_TOKEN
```

OWNERS_TOKEN | OWNERS_TOKEN_64

OWNERS_TOKEN
The 31-bit transaction token that is associated with the current transaction.

OWNERS_TOKEN_64
The 64-bit transaction token that is associated with the current transaction.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANNUM
Optional Parameter

The transaction number assigned to the newly attached transaction.

XMIQ gate, INQUIRE_TRANSACTION function

The INQUIRE_TRANSACTION function of the XMIQ gate is used to inquire upon a particular transaction.

Input Parameters

ATTACH_PARMS
Optional Parameter

Parameters to be passed to the attached transaction.

OPERATION_NAME
Optional Parameter

The 64-character buffer to receive the name of the application operation if the TRANSACTION is declared as an application entry point.

TRANSACTION_NUMBER
Optional Parameter

The number of the transaction being inquired upon.

TRANSACTION_TOKEN
Optional Parameter

Optional token to identify the transaction that the message is to be sent to. Defaults to the current transaction.

Output Parameters

ATTACH_TIME
Optional Parameter

The time when the transaction was attached.

CICS_UOW_ID
Optional Parameter

The CICS Unit Of Work Identifier associated with the transaction.

CONFDATA
Optional Parameter

The value of the CONFDATA attribute specified in the transaction definition.

Values for the parameter are:
- NO
- YES

DS_TASK_TOKEN
Optional Parameter

A token that identifies the dispatcher task associated with the transaction.

DTIMEOUT
Optional Parameter

The deadlock timeout value for the transaction.

DYNAMIC
Optional Parameter

Whether the transaction is defined to be dynamic.

Values for the parameter are:
- NO
- YES

ENTRYPOINT_RES_TOKEN
Optional Parameter

An 8-byte token that represents the application entry point.

FACILITY_NAME
Optional Parameter

The name of the principal facility associated with the transaction.

FACILITY_TOKEN
Optional Parameter

A token representing the principal facility associated with the transaction.

FACILITY_TYPE
Optional Parameter

The type of the principal facility associated with the transaction.

Values for the parameter are:
- IPECI
- NONE

START
TD
TERMINAL

INITIAL_PROGRAM
Optional Parameter

Initial program of transaction.

NETNAME
Optional Parameter

The network name of a terminal principal facility.

ORIGINAL_TRANSACTION_ID
Optional Parameter

The transid that was used to attach the transaction.

OUT_TRANSACTION_TOKEN
Optional Parameter

The token that represents this transaction.

PHASE
Optional Parameter

The phase of the transaction.

Values for the parameter are:
BIND
INIT
PRE_INIT
TERM

PRIMARY_CLIENT_TOKEN
Optional Parameter

A token representing the client for which the client was attached.

PRIMARY_CLIENT_TYPE
Optional Parameter

The type of client for which the transaction was attached.

Values for the parameter are:
APPC_SESSION
BRIDGE
IIRR
IP_ECI
LU61_SESSION
MRO_SESSION
NONE
RRS_UR
RZ_INSTORE_TRPORT
SCHEDULER
SOCKET
START
START_TERMINAL
TERMINAL
TRANDATA
WEB
XM_RUN_TRANSACTION

RE_ATTACHED_TRANSACTION
Optional Parameter

Indicates if the transaction was reattached.

Values for the parameter are:
NO
YES

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
BUFFER_TOO_SMALL
INVALID_TRANSACTION_TOKEN
NO_TRANSACTION_ENVIRONMENT
UNKNOWN_TRANSACTION_NUMBER

REMOTE
Optional Parameter

Whether the transaction is remote.

Values for the parameter are:
NO
YES

REMOTE_NAME
Optional Parameter

The name of a remote transaction on the remote system.

REMOTE_SYSTEM
Optional Parameter

The system that a remote transaction is to be routed to.

RESOURCE_NAME
Optional Parameter

The name of a resource that a suspended transaction is waiting for.

RESOURCE_TYPE
Optional Parameter

The type of resource that a suspended transaction is waiting for.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

RESTART
Optional Parameter

Whether the transaction is restartable.

Values for the parameter are:
NO
YES

RESTART_COUNT
Optional Parameter

Contains the number of times this transaction instance has been restarted.

SPURGE
Optional Parameter

Whether the transaction is system-purgeable.

Values for the parameter are:
NO
YES

START_CODE
Optional Parameter

Indicates the reason for the attach of the transaction.

Values for the parameter are:

```
C
DF
QD
S
SD
SZ
T
TT
```

STATUS

Optional Parameter

The status of the transaction.

Values for the parameter are:

```
READY
RUNNING
SUSPENDED
```

SUSPEND_TIME

Optional Parameter

Contains the length of time that the transaction has currently been suspended for.

SYSTEM_TRANSACTION

Optional Parameter

Whether the transaction has been attached by CICS.

Values for the parameter are:

```
NO
YES
```

TASK_PRIORITY

Optional Parameter

The combined priority of the transaction.

TCLASS

Optional Parameter

Whether the transaction belongs to a tclass.

TCLASS_NAME

Optional Parameter

The name of the tclass that the transaction belongs to.

TPURGE

Optional Parameter

Whether the transaction can be purged after a terminal error.

Values for the parameter are:

```
NO
YES
```

TRAN_PRIORITY

Optional Parameter

Transaction priority

TRAN_ROUTING_PROFILE

Optional Parameter

Profile to be used to route a remote transaction to a remote system.

TRANDEF_TOKEN
Optional Parameter

The token representing the returned transaction definition.

TRANNUM
Optional Parameter

Is the transaction number assigned to the newly attached transaction.

TRANSACTION_GROUP_ID
Optional Parameter

The identifier of the transaction's monitoring group.

TRANSACTION_ID
Optional Parameter

Transaction identifier

USERID
Optional Parameter

The userid of the user associated with the transaction.

XMIQ gate, INQUIRE_TRANSACTION_TOKEN function

The INQUIRE_TRANSACTION_TOKEN function of the XMIQ gate is used to return a transaction token that is associated with a specific transaction.

Input Parameters

TOKEN_OWNER | TOKEN_OWNER_64

TOKEN_OWNER
Identifies the 31-bit transaction token to retrieve for the transaction.

The parameter can take the following values:

AP
BR
DD
DP
EP
EJ
IE
IS
LG
ML
MN
PG
PI
RL
RM
RZ
SM
SO
TD
TF
US
WB
XM
XS

TOKEN_OWNER_64
Identifies the 64-bit transaction token to retrieve for the transaction.

The parameter can take the following value:

AD

TRANSACTION_TOKEN
Optional Parameter

An optional token that identifies the transaction to send the message to. The default is the current transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NO_TRANSACTION_ENVIRONMENT

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

OWNERS_TOKEN | OWNERS_TOKEN_64
OWNERS_TOKEN
The 31-bit transaction token that is associated with the current transaction.
OWNERS_TOKEN_64
The 64-bit transaction token that is associated with the current transaction.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMIQ gate, PURGE_TRANSACTION function

The PURGE_TRANSACTION function of the XMIQ gate is used to purge a particular transaction in the system.

Input Parameters

PURGE_TYPE
The type of purge that is to be attempted.

Values for the parameter are:
FORCE
KILL
NORMAL

TRANSACTION_NUMBER
The number of the transaction being inquired upon.

TRANSACTION_TOKEN
Optional token to identify the transaction that the message is to be sent to. Defaults to the current transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
FORCEPURGE_NOT_ATTEMPTED
INVALID_STATE
INVALID_TRANSACTION_TOKEN
PURGE_ABENDING_TRANSACTION
PURGE_DEFERRED
PURGE_INHIBITED
PURGE_JAVA_PROGRAM
PURGE_SYSTEM_TRANSACTION
SPURGE_PROTECTED
TRANSACTION_INITIALIZING
TRANSACTION_TERMINATING

UNKNOWN_TRANSACTION_NUMBER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMIQ gate, SET_TRANSACTION function

The SET_TRANSACTION function of the XMIQ gate is used to change some attributes associated with a particular transaction.

Input Parameters

FACILITY_TOKEN
Optional Parameter

A token representing the principal facility associated with the transaction.

FACILITY_TYPE
Optional Parameter

The type of principal facility to be associated with the attached transaction.

Values for the parameter are:
IPECI
NONE
START
TD
TERMINAL

REMOTE_NAME
Optional Parameter

The name of a remote transaction on the remote system.

REMOTE_SYSTEM
Optional Parameter

The system that a remote transaction is to be routed to.

REPORT_CONDITION
Optional Parameter

An indicator that provides a means of communicating the fact that an abend message has already been reported to the principal facility terminal or destination.

Values for the parameter are:
NO
YES

RESTART
Optional Parameter

Whether the transaction is restartable.

Values for the parameter are:
NO
YES

START_CODE
Optional Parameter

Indicates the reason for the attach.

Values for the parameter are:
C
QD
S
SD
SZ

T
TT

STORAGE_VIOLATIONS
Optional Parameter

Set to indicate that the transaction has suffered a storage violation.

Values for the parameter are:
INCREMENT

TASK_PRIORITY
Optional Parameter

The combined priority of the transaction.

TCLASS_NAME
Optional Parameter

The name of the tclass.

TRANSACTION_NUMBER
Optional Parameter

The number of the transaction being inquired upon.

TRANSACTION_TOKEN
Optional Parameter

Optional token to identify the transaction that the message is to be sent to. Defaults to the current transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
INVALID_TRANSACTION_TOKEN
NO_TRANSACTION_ENVIRONMENT
UNKNOWN_TCLASS
UNKNOWN_TRANSACTION_NUMBER

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMIQ gate, SET_TRANSACTION_TOKEN function

The SET_TRANSACTION_TOKEN function of the XMIQ gate is used to modify a transaction token that is associated with a specific transaction.

Input Parameters

OWNERS_TOKEN | OWNERS_TOKEN_64
OWNERS_TOKEN
The new value for the 31-bit transaction token.
OWNERS_TOKEN_64
The new value for the 64-bit transaction token.

TOKEN_OWNER | TOKEN_OWNER_64
TOKEN_OWNER
Identifies the 31-bit transaction token to set for the transaction.

The parameter can take the following values:
AP
BR
DD

DP
EP
EJ
IE
IS
LG
ML
MN
PG
PI
RL
RM
RZ
SM
SO
TD
TF
US
WB
XM
XS

TOKEN_OWNER_64
Identifies the particular 64-bit transaction token to set for the transaction.

The parameter can take the following value:
AD

TRANSACTION_TOKEN
Optional Parameter

An optional token that identifies the transaction to send the message to. The default is the current transaction.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
NO_TRANSACTION_ENVIRONMENT

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMIQ gate, START_BROWSE_TRANSACTION function

The START_BROWSE_TRANSACTION function of the XMIQ gate is used to initiate a browse of all transactions in the system.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

BROWSE_TOKEN
Token identifying this transaction definition browse.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMIQ gate, START_BROWSE_TXN_TOKEN function

The START_BROWSE_TXN_TOKEN function of the XMIQ gate is used to initiate a browse of the transaction token of a specific component in all transactions in the system.

Input Parameters

TOKEN_OWNER | TOKEN_OWNER_64

TOKEN_OWNER

Identifies the 31-bit transaction token that is to be browsed in the transactions.

The parameter can take the following values:

AP
BR
DD
DP
EP
EJ
IE
IS
LG
ML
MN
PG
PI
RL
RM
RZ
SM
SO
TD
TF
US
WB
XM
XS

TOKEN_OWNER_64

Identifies the 64-bit transaction token that is to be browsed in the transactions.

The parameter can take the following value:

AD

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

ABEND
LOOP

BROWSE_TOKEN

A token that identifies this transaction definition browse.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMLD gate, LOCATE_AND_LOCK_TRANDEF function

The LOCATE_AND_LOCK_TRANDEF function of the XMLD gate is used to locate a particular transaction definition instance.

Input Parameters

TPNAME
Alternative means of specifying the transaction identifier to attach.

TRANSACTION_ID
The transaction identifier to attach.

USE_DTRTRAN
Optional Parameter

If the named transaction-id or tpname cannot be found then indicates whether the DTRTRAN, if installed, should be used instead.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:
NOT_FOUND

The following values are returned when RESPONSE is INVALID:
INVALID_TPNAME

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANDEF_TOKEN
The token representing the returned transaction definition.

PRIMARY_TRANSACTION_ID
Optional Parameter

The primary transaction identifier of the returned transaction. definition.

XMLD gate, UNLOCK_TRANDEF function

The UNLOCK_TRANDEF function of the XMLD gate is used to unlock a previously located transaction definition instance.

Input Parameters

TRANDEF_TOKEN
Transaction definition instance to unlock.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

The following values are returned when RESPONSE is INVALID:
INVALID_TOKEN
NOT_LOCKED

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMRU gate, RUN_TRANSACTION function

Run a BTS transaction.

Input Parameters

TRANID
The transaction identifier.

CLIENT_DATA_BLOCK
Optional Parameter

Client data associated with the request.

CLIENT_TYPE
Optional Parameter

A string that indicates the type of client.

PROGRAM
Optional Parameter

The program associated with the transaction.

USERID
Optional Parameter

the user ID under which the transaction runs.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:

```
BIND_FAILURE
NOTAUTH
TASK_ABENDED
TRANSACTION_HANG
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ABEND_CODE
Optional Parameter

The abend code if an abend occurred in the BTS transaction.

ABEND_PROGRAM
Optional Parameter

The name of the program that ended abnormally if an abend occurred in the BTS transaction.

XMSR gate, INQUIRE_DTRTRAN function

The INQUIRE_DTRTRAN function of the XMSR gate returns the name of the dynamic transaction routing transaction.

Output Parameters

DTRTRAN
The name of the dynamic transaction routing transaction definition.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON
Optional Parameter

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOGIC_ERROR
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_MXT_LIMIT
```

LIMIT_TOO_HIGH

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

XMSR gate, INQUIRE_MXT function

The INQUIRE_MXT function of the XMSR gate is used to inquire upon the state of MXT in the system.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CURRENT_ACTIVE

Optional Parameter

The number of active transactions in the tclass.

MXT_LIMIT

Optional Parameter

The maximum number of transactions in the transaction class that are allowed to be active.

MXT_QUEUED

Optional Parameter

The number of user transactions queued for MXT.

TCLASS_QUEUED

Optional Parameter

The number of transactions queued for tclass membership.

XMSR gate, SET_DTRTRAN function

The SET_DTRTRAN function of the XMSR gate changes the dynamic transaction routing transaction definition.

Input Parameters

DTRTRAN

The name of the dynamic transaction routing transaction definition.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REASON

Optional Parameter

The following values are returned when RESPONSE is DISASTER:
ABEND
LOGIC_ERROR
LOOP

The following values are returned when RESPONSE is EXCEPTION:
INVALID_MXT_LIMIT
LIMIT_TOO_HIGH

The following values are returned when RESPONSE is INVALID:
INVALID_FUNCTION

XMSR gate, SET_MXT function

The SET_MXT function of the XMSR gate is used to change MXT in the system.

Input Parameters

MXT_LIMIT

The requested setting for MXT.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:

INVALID_MXT_LIMIT
LIMIT_TOO_HIGH

MXT_LIMIT_SET

The MXT limit that could be set.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMXD gate, ADD_REPLACE_TRANDEF function

The ADD_REPLACE_TRANDEF function of the XMXD gate is used to install a transaction definition.

Input Parameters

PROFILE_NAME

The profile that is to be found.

TRAN_PRIORITY

Transaction priority.

TRANSACTION_ID

The transaction identifier to attach.

ALIAS

Optional Parameter

Alternative name for transaction definition.

BREXIT

Optional Parameter

The name of the default bridge exit to be associated with this transaction.

CATALOGUED_EXTERNALS

Optional Parameter

Block of data specified as an alternative to these parameters when a transaction definition is being installed from the catalog.

CMDSEC

Optional Parameter

Whether command security checking is active.

Values for the parameter are as follows:

NO
YES

CONFDATA

Optional Parameter

The value of the CONFDATA attribute specified in the TRANSACTION definition.

Values for the parameter are as follows:

NO
YES

DTIMEOUT
Optional Parameter

The deadlock timeout value for the transaction.

DUMP
Optional Parameter

Whether transaction dumps are to be taken.

Values for the parameter are as follows:
NO
YES

DYNAMIC
Optional Parameter

Whether the transaction is defined to be dynamic.

Values for the parameter are as follows:
NO
YES

INDOUBT
Optional Parameter

The action to take if work performed by the transaction becomes indoubt.

Values for the parameter are as follows:
BACKOUT
COMMIT

INDOUBT_WAIT
Optional Parameter

Indicates whether an indoubt unit of work (UOW) is to wait, pending recovery from a failure that occurs after the UOW has entered the indoubt state.

Values for the parameter are as follows:
NO
YES

INDOUBT_WAIT_TIME
Optional Parameter

Indicates how long the transaction is to wait before taking an arbitrary decision about an indoubt unit of work.

INITIAL_PROGRAM
Optional Parameter

Initial program of transaction.

ISOLATE
Optional Parameter

Whether the transaction runs in its own subspace.

Values for the parameter are as follows:
NO
YES

LOCAL_QUEUING
Optional Parameter

Whether the transaction is eligible to queue locally when it is started on the remote system.

Values for the parameter are as follows:

NO
YES

OTSTIMEOUT
Optional Parameter

The value of the OTSTIMEOUT attribute in the transaction definition.

PARTITIONSET
Optional Parameter

The partitionset defined for the transaction.

PARTITIONSET_NAME
Optional Parameter

The name of the user defined partitionset used by the transaction.

REMOTE_NAME
Optional Parameter

The name of a remote transaction on the remote system.

REMOTE_SYSTEM
Optional Parameter

The system that a remote transaction is to be routed to.

RESSEC
Optional Parameter

Whether resource security checking is active.

Values for the parameter are as follows:
NO
YES

RESTART
Optional Parameter

Whether the transaction is restartable.

Values for the parameter are as follows:
NO
YES

ROUTABLE_STATUS
Optional Parameter

Specifies whether, if the transaction is the subject of an eligible EXEC CICS START command, it will be routed using the enhanced routing method.

Values for the parameter are as follows:
NOTROUTABLE
ROUTABLE

RUNAWAY_LIMIT
Optional Parameter

The runaway limit associated with the transaction.

SHUTDOWN
Optional Parameter

Whether the transaction can be run during shutdown.

Values for the parameter are as follows:
DISABLED
ENABLED

SPURGE
Optional Parameter

Whether the transaction is system-purgeable.

Values for the parameter are as follows:
NO
YES

STATUS
Optional Parameter

The status of the transaction.

Values for the parameter are as follows:
DISABLED
ENABLED

STORAGE_CLEAR
Optional Parameter

Whether task-lifetime storage is to be cleared before it is released by using a freemain request.

Values for the parameter are as follows:
NO
YES

STORAGE_FREEZE
Optional Parameter

Whether storage freeze is on for the transaction.

Values for the parameter are as follows:
NO
YES

SYSTEM_DEFINITION
Optional Parameter

A binary value that indicates whether the transaction is defined by the system.

Values for the parameter are as follows:
NO
YES

SYSTEM_RUNAWAY
Optional Parameter

Whether the transaction uses the default system runaway limit.

TASKDATAKEY
Optional Parameter

The storage key that task-lifetime storage is allocated in.

Values for the parameter are as follows:
CICS
USER

TASKDATALOC
Optional Parameter

The location of task-lifetime storage.

Values for the parameter are as follows:
ANY
BELOW

TASKREQ
Optional Parameter

Alternative name for transaction definition so that it can be invoked by PF/PA key, light pen, etc.

TCLASS
Optional Parameter

Whether the transaction belongs to a tclass.

TCLASS_NAME

Optional Parameter

The name of the tclass.

TPNAME

Optional Parameter

Alternative means of specifying the transaction identifier to attach.

TPURGE

Optional Parameter

Whether the transaction can be purged after a terminal error.

Values for the parameter are as follows:

NO
YES

TRACE

Optional Parameter

The level of tracing associated with the transaction.

Values for the parameter are as follows:

SPECIAL
STANDARD
SUPPRESSED

TRAN_ROUTING_PROFILE

Optional Parameter

Profile to be used to route a remote transaction to a remote system.

TWASIZE

Optional Parameter

Size of Transaction Work Area.

XTRANID

Optional Parameter

Alternative name for transaction definition originally specified in hexadecimal notation.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:

ALIAS_INVALID
RECOVERY_NOT_COMPLETE
RUNAWAY_LIMIT_INVALID
TASKREQ_INVALID
TPNAME_INVALID
TRANSACTION_ID_INVALID
TWASIZE_INVALID
XTRANID_INVALID

The following values are returned when RESPONSE is INVALID:

INITIAL_PROGRAM_EXPECTED
PARTITIONSET_NAME_EXPECTED
REMOTE_NAME_EXPECTED
REMOTE_SYSTEM_EXPECTED
RUNAWAY_LIMIT_EXPECTED

TCLASS_NAME_EXPECTED
TRAN_ROUTING_PROF_EXPECTED

The values for the parameter are as follows:

ALIAS_EXISTS_AS_PRIMARY

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANDEF_TOKEN

Optional Parameter

The token representing the returned transaction definition.

XMXD gate, INQUIRE_REMOTE_TRANDEF function

The INQUIRE_REMOTE_TRANDEF function of the XMXD gate is used to inquire upon a remote transaction definition.

Input Parameters

REMOTENAME_KEY

Remote name of remote transaction definition to be found.

REMOTESYSTEM_KEY

Remote system of remote transaction definition to be found.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:

REMOTE_NOT_FOUND

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BREXIT

Optional Parameter

The name of the default bridge exit to be associated with this transaction.

CMDSEC

Optional Parameter

Whether command security checking is active.

Values for the parameter are as follows:

NO
YES

CONFDATA

Optional Parameter

The value of the CONFDATA attribute specified in the TRANSACTION definition.

Values for the parameter are as follows:

NO
YES

DTIMEOUT

Optional Parameter

The deadlock timeout value for the transaction.

DTRTRAN

Optional Parameter

The name of the dynamic transaction routing transaction definition.

Values for the parameter are as follows:

NO
YES

DUMP

Optional Parameter

Whether transaction dumps are to be taken.

Values for the parameter are as follows:

NO
YES

DYNAMIC

Optional Parameter

Whether the transaction is defined to be dynamic.

Values for the parameter are as follows:

NO
YES

INDOUBT

Optional Parameter

The action to take if work performed by the transaction becomes indoubt.

Values for the parameter are as follows:

BACKOUT
COMMIT

INDOUBT_WAIT

Optional Parameter

Indicates whether an indoubt unit of work (UOW) is to wait, pending recovery from a failure that occurs after the UOW has entered the indoubt state.

Values for the parameter are as follows:

NO
YES

INDOUBT_WAIT_TIME

Optional Parameter

Indicates how long the transaction is to wait before taking an arbitrary decision about an indoubt unit of work.

INITIAL_PROGRAM

Optional Parameter

Initial program of transaction.

ISOLATE

Optional Parameter

Whether the transaction runs in its own subspace.

Values for the parameter are as follows:

NO
YES

LOCAL_QUEUING

Optional Parameter

Whether the transaction is eligible to queue locally when it is started on the remote system.

Values for the parameter are as follows:

NO

YES

OTSTIMEOUT
Optional Parameter

The value of the OTSTIMEOUT attribute in the transaction definition.

PARTITIONSET
Optional Parameter

The partitionset defined for the transaction.

Values for the parameter are as follows:
KEEP
NAMED
NONE
OWN

PARTITIONSET_NAME
Optional Parameter

The name of the user defined partitionset used by the transaction.

PROFILE_NAME
Optional Parameter

Profile of transaction.

REMOTE
Optional Parameter

Whether the transaction is remote.

Values for the parameter are as follows:
NO
YES

REMOTE_NAME
Optional Parameter

The name of a remote transaction on the remote system.

REMOTE_SYSTEM
Optional Parameter

The system that a remote transaction is to be routed to.

RESSEC
Optional Parameter

Whether resource security checking is active.

Values for the parameter are as follows:
NO
YES

RESTART
Optional Parameter

Whether the transaction is restartable.

Values for the parameter are as follows:
NO
YES

ROUTABLE_STATUS
Optional Parameter

Specifies whether, if the transaction is the subject of an eligible EXEC CICS START command, it will be routed using the enhanced routing method.

Values for the parameter are as follows:
NOTROUTABLE

ROUTABLE

RUNAWAY_LIMIT
Optional Parameter

The runaway limit associated with the transaction.

SHUTDOWN
Optional Parameter

Whether the transaction can be run during shutdown.

Values for the parameter are as follows:
DISABLED
ENABLED

SPURGE
Optional Parameter

Whether the transaction is system-purgeable.

Values for the parameter are as follows:
NO
YES

STATUS
Optional Parameter

The status of the transaction.

Values for the parameter are as follows:
DISABLED
ENABLED

STORAGE_CLEAR
Optional Parameter

Whether task-lifetime storage is to be cleared before it is released by using a freemain request.

Values for the parameter are as follows:
NO
YES

STORAGE_FREEZE
Optional Parameter

Whether storage freeze is on for the transaction.

Values for the parameter are as follows:
NO
YES

SYSTEM_ATTACH
Optional Parameter

Indicates whether the transaction should be attached as a system transaction.

Values for the parameter are as follows:
NO
YES

SYSTEM_RUNAWAY
Optional Parameter

Whether the transaction uses the default system runaway limit.

Values for the parameter are as follows:
NO
YES

TASKDATAKEY
Optional Parameter

The storage key that task-lifetime storage is allocated in.

Values for the parameter are as follows:
CICS
USER

TASKDATALOC
Optional Parameter

The location of task-lifetime storage.

Values for the parameter are as follows:
ANY
BELOW

TCLASS
Optional Parameter

Whether the transaction belongs to a tclass.

TCLASS_NAME
Optional Parameter

The name of the tclass that the transaction belongs to.

TPURGE
Optional Parameter

Whether the transaction can be purged after a terminal error.

Values for the parameter are as follows:
NO
YES

TRACE
Optional Parameter

The level of tracing associated with the transaction.

Values for the parameter are as follows:
SPECIAL
STANDARD
SUPPRESSED

TRAN_PRIORITY
Optional Parameter

Transaction priority

TRAN_ROUTING_PROFILE
Optional Parameter

Profile to be used to route a remote transaction to a remote system.

TRANSACTION_ID
Optional Parameter

Transaction identifier

TWASIZE
Optional Parameter

Size of Transaction Work Area.

XMXD gate, INQUIRE_TRANDEF function

The INQUIRE_TRANDEF function of the XMXD gate is used to inquire upon a named transaction definition.

Input Parameters

APPLICATION_NAME
Optional Parameter

The 64-character buffer to receive the name of the application associated with the transaction.

INQ_TRANSACTION_ID
Transaction-id to inquire upon.

OPERATION_NAME
Optional Parameter

The 64-character buffer to receive the name of the operation associated with the transaction.

PLATFORM_NAME
Optional Parameter

The 64-character buffer to receive the name of the platform associated with the transaction.

TRANDEF_TOKEN
Transaction definition instance to unlock.

USE_DTRTRAN
Optional Parameter

If the named transaction-id or tpname cannot be found then indicates whether the DTRTRAN, if installed, should be used instead.

Values for the parameter are as follows:
NO
YES

Output Parameters

AVAILSTATUS
Optional Parameter

A 1-byte enumeration expressing the availability status of the application entry point.

Values for the parameter are:
- AVAILABLE
- UNAVAILABLE
- NONE

REASON
The following values are returned when RESPONSE is DISASTER:
LOGIC_ERROR

The following values are returned when RESPONSE is EXCEPTION:
UNKNOWN_TRANSACTION_ID

The following values are returned when RESPONSE is INVALID:
INVALID_TOKEN

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

BREXIT
Optional Parameter

The name of the bridge exit defined by the BREXIT parameter of the transaction resource definition.

CMDSEC
Optional Parameter

Whether command security checking is active.

Values for the parameter are as follows:
NO
YES

CONFDATA
Optional Parameter

A binary value that indicates whether CICS should clear storage that is released from a task executing this transaction, to prevent other tasks accidentally viewing confidential data.

Values for the parameter are as follows:
NO
YES

DTIMEOUT
Optional Parameter

The deadlock timeout value for the transaction.

DTRTRAN
Optional Parameter

The name of the dynamic transaction routing transaction definition.

Values for the parameter are as follows:
NO
YES

DUMP
Optional Parameter

Whether transaction dumps are to be taken.

Values for the parameter are as follows:
NO
YES

DYNAMIC
Optional Parameter

Whether the transaction is defined to be dynamic.

Values for the parameter are as follows:
NO
YES

ENTRYPOINT_BUNDLE_TOKEN
Optional Parameter

An 8-byte token that represents the bundle that defined the application entry point.

ENTRYPOINT_RES_TOKEN
Optional Parameter

An 8-byte token that represents the application entry point.

INDOUBT
Optional Parameter

The action to take if work performed by the transaction becomes indoubt.

Values for the parameter are as follows:
BACKOUT
COMMIT

INDOUBT_WAIT
Optional Parameter

A binary value that indicates whether CICS wait to determine whether recoverable resources are to be backed out or committed if a failure occurs while the unit of work associated with the transaction is in an indoubt state.

Values for the parameter are as follows:

NO
YES

INDOUBT_WAIT_TIME

Optional Parameter

The length of time for which CICS should wait to for resolution if a failure occurs while the unit of work associated with the transaction is in an indoubt state.

INITIAL_PROGRAM

Optional Parameter

Initial program of transaction.

ISOLATE

Optional Parameter

Whether the transaction runs in its own subspace.

Values for the parameter are as follows:

NO
YES

LOCAL_QUEUING

Optional Parameter

Whether the transaction is eligible to queue locally when it is started on the remote system.

Values for the parameter are as follows:

NO
YES

MAJOR_VERSION

Optional Parameter

The fullword binary value that is the major version of the application associated with the transaction.

MICRO_VERSION

Optional Parameter

The fullword binary value that is the micro version of the application associated with the transaction.

MINOR_VERSION

Optional Parameter

The fullword binary value that is the minor version of the application associated with the transaction.

OTSTIMEOUT

Optional Parameter

The time for which an OTS transaction, created in an EJB environment executing under this CICS transaction, is allowed to execute before syncpoint.

PARTITIONSET

Optional Parameter

The partitionset defined for the transaction.

Values for the parameter are as follows:

KEEP
NAMED

NONE
OWN

PARTITIONSET_NAME
Optional Parameter

The name of the user defined partitionset used by the transaction.

PROFILE_NAME
Optional Parameter

Profile of transaction.

REMOTE
Optional Parameter

Whether the transaction is remote.

Values for the parameter are as follows:
NO
YES

REMOTE_NAME
Optional Parameter

The name of a remote transaction on the remote system.

REMOTE_SYSTEM
Optional Parameter

The system that a remote transaction is to be routed to.

RESSEC
Optional Parameter

Whether resource security checking is active.

Values for the parameter are as follows:
NO
YES

RESTART
Optional Parameter

Whether the transaction is restartable.

Values for the parameter are as follows:
NO
YES

ROUTABLE_STATUS
Optional Parameter

Specifies whether, if the transaction is the subject of an eligible EXEC CICS START command, it will be routed using the enhanced routing method.

Values for the parameter are as follows:
NOTROUTABLE
ROUTABLE

RUNAWAY_LIMIT
Optional Parameter

The runaway limit associated with the transaction.

SHUTDOWN
Optional Parameter

Whether the transaction can be run during shutdown.

Values for the parameter are as follows:
DISABLED
ENABLED

SPURGE
Optional Parameter

Whether the transaction is system-purgeable.

Values for the parameter are as follows:
NO
YES

STATUS
Optional Parameter

The status of the transaction.

Values for the parameter are as follows:
DISABLED
ENABLED

STORAGE_CLEAR
Optional Parameter

Whether task-lifetime storage is to be cleared before it is released by a freemain request.

Values for the parameter are as follows:
NO
YES

STORAGE_FREEZE
Optional Parameter

Whether storage freeze is on for the transaction.

Values for the parameter are as follows:
NO
YES

SYSTEM_ATTACH
Optional Parameter

Whether a system task will be attached using this transaction definition.

Values for the parameter are as follows:
NO
YES

SYSTEM_RUNAWAY
Optional Parameter

Whether the transaction uses the default system runaway limit.

Values for the parameter are as follows:
NO
YES

TASKDATAKEY
Optional Parameter

The storage key that task-lifetime storage is allocated in.

Values for the parameter are as follows:
CICS
USER

TASKDATALOC
Optional Parameter

The location of task-lifetime storage.

Values for the parameter are as follows:
ANY

BELOW

TCLASS
Optional Parameter

Whether the transaction belongs to a tclass.

TCLASS_NAME
Optional Parameter

The name of the tclass that the transaction belongs to.

TPURGE
Optional Parameter

Whether the transaction can be purged after a terminal error.

Values for the parameter are as follows:
NO
YES

TRACE
Optional Parameter

The level of tracing associated with the transaction.

Values for the parameter are as follows:
SPECIAL
STANDARD
SUPPRESSED

TRAN_PRIORITY
Optional Parameter

Transaction priority

TRAN_ROUTING_PROFILE
Optional Parameter

Profile to be used to route a remote transaction to a remote system.

TRANSACTION_ID
Optional Parameter

Transaction identifier

TWASIZE
Optional Parameter

Size of Transaction Work Area.

XMXD gate, SET_TRANDEF function

The SET_TRANDEF function of the XMXD gate is used to modify transaction definition creating a new transaction. definition instance.

Input Parameters

TRANSACTION_ID
The transaction identifier to attach.

BUNDLE_CALLER
Optional Parameter

Defines whether the caller is either the RL client (YES) or a CICS internal module (CICSOVERRIDE) and therefore allowed to modify the state of a transaction that is defined in a CICS bundle.

Values for the parameter are as follows:
CICSOVERRIDE
NO
YES

DUMP
Optional Parameter

Whether transaction dumps are to be taken.

Values for the parameter are:
NO
YES

RUNAWAY_LIMIT
Optional Parameter

The runaway limit associated with the transaction.

SHUTDOWN
Optional Parameter

Whether the transaction can be run during shutdown.

Values for the parameter are:
DISABLED
ENABLED

SHUTDOWN_DISABLEOVERRIDE
Optional Parameter

Whether to override a SHUTDOWN setting of DISABLED for the transaction definition.

Values for the parameter are:
NO
YES

SPURGE
Optional Parameter

Whether the transaction is system-purgeable.

Values for the parameter are:
NO
YES

STATUS
Optional Parameter

The status of the transaction.

Values for the parameter are:
DISABLED
ENABLED

STORAGE_FREEZE
Optional Parameter

Whether storage freeze is on for the transaction.

Values for the parameter are:
NO
YES

SYSTEM_ATTACH
Optional Parameter

Indicates whether the transaction should be attached as a system transaction.

Values for the parameter are:
NO
YES

SYSTEM_RUNAWAY
Optional Parameter

Whether the transaction uses the default system runaway limit.

TCLASS

Optional Parameter

Whether the transaction belongs to a tclass.

TCLASS_NAME

Optional Parameter

The name of the tclass.

TRACE

Optional Parameter

The level of tracing associated with the transaction.

Values for the parameter are:

```
SPECIAL
STANDARD
SUPPRESSED
```

TRAN_PRIORITY

Optional Parameter

Transaction priority

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
LOGIC_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
RUNAWAY_LIMIT_INVALID
UNKNOWN_TCLASS
UNKNOWN_TRANSACTION_ID
```

The following values are returned when RESPONSE is INVALID:

```
RUNAWAY_LIMIT_EXPECTED
TCLASS_NAME_EXPECTED
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

TRANDEF_TOKEN

Optional Parameter

The token representing the returned transaction definition.

XMXE gate, FREE_TXN_ENVIRONMENT function

The FREE_TXN_ENVIRONMENT function of the XMXE gate is used to release a transaction environment for a task that was DS instead XM attached.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
ATTACHED_TRANSACTION
CALL_NOT_MADE_ON_QR
INVALID_FUNCTION
LOOP
NO_ENVIRONMENT
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMXE gate, GET_TXN_ENVIRONMENT function

The GET_TXN_ENVIRONMENT function of the XMXE gate is used to acquire a transaction environment for a task that was DS instead XM attached.

Output Parameters

REASON

The values for the parameter are:

```
ABEND
ATTACHED_TRANSACTION
CALL_NOT_MADE_ON_QR
DUPLICATE_ENVIRONMENT
INVALID_FUNCTION
LOOP
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Transaction manager domain's generic gates

Table 62 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 62. Transaction manager domain's generic gates

Gate	Trace	Functions	Format
XMDM	XM 0101 XM 0102	PRE_INITIALIZE INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM
XMST	XM 0C01 XM 0C02	COLLECT_STATISTICS COLLECT_RESOURCE_STATS	STST
XMRL	XM 1701 XM 1702 XM 1703 XM 1704 XM 1705	CREATE DISCARD INQUIRE INQUIRE_BY_NAME SET	RLCB

Transaction Manager domain's callback formats

Table 63 describes the call-back formats owned by the domain and shows the functions performed on the calls.

Table 63. Transaction Manager domain's call-back formats

Format	Calling module	Function
XMAC	DFHXMTA DFHXMXE	INIT_XM_CLIENT BIND_XM_CLIENT TRANSACTION_HANG ABEND_TERMINATE RELEASE_XM_CLIENT

Note: In the descriptions of the formats, the input parameters are input not to the transaction Manager domain, but to the domain being called by the transaction Manager domain. Similarly, the output parameters are output by the domain that was called by the transaction Manager domain, in response to the call.

XMAC gate, ABEND_TERMINATE function

Clean up after a deferred abend has been noted during transaction initialization.

Input Parameters

CLIENT_REQUEST_BLOCK
A block that refers to data that defines the context of the request.

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMAC gate, BIND_XM_CLIENT function

Initialize primary resources and client recoverable resources, and optionally set the program to be called after initialization is complete.

Input Parameters

CLIENT_REQUEST_BLOCK
A block that refers to data that defines the context of the request.

Output Parameters

APPLICATION_PROGRAM
The application program to be called after initialization is complete.

LINK_APPLICATION_PROGRAM
A binary value that indicates whether an application program is to be called after initialization is complete.

Values for the parameter are:
NO
YES

ROUTABLE
A binary value that indicates whether the application program request can be routed.

Values for the parameter are:
NO
YES

REASON
The values for the parameter are:
BAD_ENVIRONMENT

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMAC gate, INIT_XM_CLIENT function

Initialize the Transaction Manager client and return the user token extracted from the client token. Also return whether this user token should be used to set up the transaction user.

Input Parameters

CLIENT_REQUEST_BLOCK
A block that refers to data that defines the context of the request.

Output Parameters

USER_TOKEN
A token that is used to manage interactions between the transaction manager and the client.

SET_USER_TOKEN
A binary value that indicates whether the user token is set.

Values for the parameter are:
NO
YES

REASON
The values for the parameter are:
INVALID_FORMAT
INVALID_FUNCTION
ABEND
BAD_ENVIRONMENT
RESTART_FAILURE
REMOTE_TRANSACTION
TRANSACTION_ABEND
INVALID_USERID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMAC gate, RELEASE_XM_CLIENT function

Clean up resources acquired by INIT_XM_CLIENT and .BIND_XM_CLIENT during Transaction Manager tear-down of the transaction environment.

Input Parameters

TERMINATION_TYPE
Indicates whether the transaction was terminated normally or abnormally.

Values for the parameter are:
NORMAL
ABNORMAL

RESTART_REQUESTED
Optional parameter

A binary value that indicates whether the transaction should be restarted.

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
RESTART_FAILURE
TRANSACTION_ABEND
BAD_ENVIRONMENT
ABEND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMAC gate, TRANSACTION_HANG function

Clean up after a severe error has taken place during transaction initialization.

Input Parameters

CLIENT_REQUEST_BLOCK
A block that refers to data that defines the context of the request.

Output Parameters

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Transaction manager domain's generic formats

Table 64 describes the generic formats owned by the domain and shows the functions performed on the calls.

Table 64. Transaction manager domain's generic formats

Format	Calling modules	Functions
XMNT	DFHXMSR DFHXMAT DFHXMTA DFHXMCL	MXT_NOTIFY MXT_CHANGE_NOTIFY
XMDN	DFHXMXD DFHXMQD DFHXMDD	TRANDEF_NOTIFY TRANDEF_DELETE_QUERY
XMPP	DFHXMIQ	FORCE_PURGE_INHIBIT_QUERY

Note: In the descriptions of the formats, the input parameters are input not to the transaction manager domain, but to the domain being called by the transaction manager domain. Similarly, the output parameters are output by the domain that was called by the transaction manager domain, in response to the call.

XMDN gate, TRANDEF_DELETE_QUERY function

The TRANDEF_DELETE_QUERY function of the XMDN format allows other domains to object to the deletion of the named transaction. definition.

Input Parameters

TRANSACTION_ID

The transaction definition subject to the delete request.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
LOGIC_ERROR
```

The values for the parameter are:

```
AID_PENDING
ICE_PENDING
SIT_PARAMETER
```

INHIBIT_DELETE

Indicates whether the called domain wants to inhibit the deletion of the named transaction definition.

Values for the parameter are:

```
NO
YES
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMDN gate, TRANDEF_NOTIFY function

The TRANDEF_NOTIFY function of the XMDN format is used to notify other domains that a transaction definition has been installed, changed, or deleted. The called domain can then modify any transaction definition related data they are keeping for that definition.

Input Parameters

EVENT

Indicates the event that has caused the notify to be sent.

Values for the parameter are:

```
CHANGE
DELETE
INSTALL
```

TRANDEF_TOKEN

Token identifying the transaction definition instance subject to the event.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
LOGIC_ERROR
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMNT gate, MXT_CHANGE_NOTIFY function

The MXT_CHANGE_NOTIFY function of XMNT format is used to notify other domains of a change to the MXT limit. The called domains indicate whether they can cope with the new limit.

Input Parameters

REQUESTED_MXT

The new limit requested for MXT.

Output Parameters

REASON

The following values are returned when RESPONSE is EXCEPTION:

```
LIMIT_TOO_HIGH
```

ALLOCATED_MXT

Indicates the limit that the called domain can cope with when the LIMIT_TOO_HIGH exception is returned.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMNT gate, MXT_NOTIFY function

The MXT_NOTIFY function of XMNT format is used to notify other domains when CICS is at, or no longer at, the maximum task limit for user tasks.

Input Parameters

MXTQUEUING

Indicates whether queuing for MXT has just started or just stopped.

Values for the parameter are:

```
STARTED
STOPPED
```

Output Parameters

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XMPP gate, FORCE_PURGE_INHIBIT_QUERY function

The FORCE_PURGE_INHIBIT_QUERY function of the XMPP format allows other domains to object to the force purge request for the specified transaction.

Input Parameters

RESOURCE_NAME
The name of the resource for which the task is waiting in the dispatcher.

RESOURCE_TYPE
The type of resource for which the task is waiting in the dispatcher.

TRANSACTION_TOKEN
Token identifying the transaction that is subject to the force purge request.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

INHIBIT_PURGE
Indicates whether the called domain wants to inhibit the force purge of the transaction.

Values for the parameter are:

```
NO
YES
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Modules

Module	Function
DFHXMAB	XM domain abend program
DFHXMAT	Handles the following requests: ATTACH
DFHXMBD	Handles the following requests: START_BROWSE_TRANDEF GET_NEXT_TRANDEF END_BROWSE_TRANDEF

Module	Function
DFHXMCL	Handles the following requests: ADD_REPLACE_TCLASS ADD_TCLASS INQUIRE_TCLASS SET_TCLASS DELETE_TCLASS START_BROWSE_TCLASS GET_NEXT_TCLASS END_BROWSE_TCLASS REGISTER_TCLASS_USAGE DEREGISTER_TCLASS_USAGE LOCATE_AND_LOCK_TCLASS UNLOCK_TCLASS
DFHXMDD	Handles the following requests: DELETE_TRANDEF
DFHXMDM	Handles the following requests: PRE_INITIALIZE INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHXMDUF	XM domain offline dump formatting routine
DFHXMER	Handles the following requests: SET_DEFERRED_MESSAGE INQUIRE_DEFERRED_MESSAGE SET_DEFERRED_ABEND INQUIRE_DEFERRED_ABEND REPORT_MESSAGE ABEND_TRANSACTION
DFHXMFD	Handles the following requests: FIND_PROFILE
DFHXMIQ	Handles the following requests: INQUIRE_TRANSACTION SET_TRANSACTION START_BROWSE_TRANSACTION GET_NEXT_TRANSACTION END_BROWSE_TRANSACTION START_BROWSE_TXN_TOKEN GET_NEXT_TXN_TOKEN END_BROWSE_TXN_TOKEN INQUIRE_TRANSACTION_TOKEN SET_TRANSACTION_TOKEN PURGE_TRANSACTION
DFHXMLD	Handles the following requests: LOCATE_AND_LOCK_TRANDEF UNLOCK_TRANDEF
DFHXMQC	Is an internal module which handles the following requests: TCLASS_ACQUIRE TCLASS_RELEASE TCLASS_LIMIT_CHANGE TCLASS_QUEUE_CHANGE
DFHXMQD	Is an internal module which handles the following requests: QUIESCE_TRANDEF DELETE_INSTANCE

Module	Function
DFHXMRP	Is an internal module which handles the following requests: DEFINITION_RECOVERY
DFHXMSR	Handles the following requests: INQUIRE_MXT SET_MXT INQUIRE_DTRTRAN SET_DTRTRAN
DFHXMST	Handles the following requests: COLLECT_STATISTICS COLLECT_RESOURCE_STATS
DFHXMTRI	Interprets XM domain trace entries
DFHXMXD	Handles the following requests: ADD_REPLACE_TRANDEF SET_TRANDEF INQUIRE_TRANDEF INQUIRE_REMOTE_TRANDEF
DFHXMXE	Handles the following requests: GET_TXN_ENVIRONMENT FREE_TXN_ENVIRONMENT

Exits

There is one specific global user exit point in the transaction manager, XXMAT which is called during Attach processing. Note also that the general resource install/discard exit, XRSINDI is also called by transaction manager to log installs and discards of TRANSACTION and TCLASS definitions.

Security domain (XS)

The security domain manages the security of CICS resources and the interaction with the security manager.

Security domain's specific gates

The specific gates provide access for other domains to functions that are provided by the XS domain.

XSAD gate, ADD_USER_WITH_PASSWORD function

The ADD_USER_WITH_PASSWORD function of the XSAD gate is used to add a user to the security domain and verify the associated password or oidcard.

Input Parameters

APPLID
is the application identifier for the CICS region.

PASSWORD
is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

SIGNON_TYPE
is the type of signon for the userid (specified by the USERID value).

Values for the parameter are:

```
ATTACH_SIGN_ON
DEFAULT_SIGN_ON
IRC_SIGN_ON
```

LU61_SIGN_ON
LU62_SIGN_ON
NON_TERMINAL_SIGN_ON
PRESET_SIGN_ON
USER_SIGN_ON
XRF_SIGN_ON

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

ENTRY_PORT_NAME
Optional Parameter

is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

ENTRY_PORT_TYPE
Optional Parameter

is the type of the optional entry port to be assigned to the userid. This parameter is only valid if ENTRY_PORT_NAME is also specified.

Values for the parameter are:
TERMINAL
CONSOLE

GROUPID
Optional Parameter

is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid is to be assigned.

GROUPID_LENGTH
Optional Parameter

is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

NEW_PASSWORD
Optional Parameter

is a new password, 1 through 10 alphanumeric characters, to be assigned to the userid (specified by the USERID value). This parameter is only valid if PASSWORD is also specified.

OIDCARD
Optional Parameter

is an optional oidcard (operator identification card); a 65-byte field containing further security data from a magnetic strip reader (MSR) on 32xx devices.

PASSWORD_TYPE
Optional Parameter

specifies if the password is masked.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is EXCEPTION:
APPLICATION_NOTAUTH
ENTRY_PORT_NOTAUTH

ESM_INACTIVE
ESM_TRANQUIL
GETMAIN_FAILURE
GROUP_ACCESS_REVOKED
INVALID_GROUPID
INVALID_NEW_PASSWORD
INVALID_USERID
OIDCARD_NOTAUTH
OIDCARD_REQUIRED
PASSWORD_EXPIRED
PASSWORD_NOTAUTH
PASSWORD_REQUIRED
SECLABEL_FAILURE
SECURITY_INACTIVE
UNKNOWN_ESM_ERROR
USERID_NOT_DEFINED
USERID_NOT_IN_GROUP
USERID_REVOKED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SECURITY_TOKEN
is the token identifying the userid.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSAD gate, ADD_USER_WITHOUT_PASSWORD function

The ADD_USER_WITHOUT_PASSWORD function of the XSAD gate is used to add a user to the security domain without verification of a associated password or oidcard.

Input Parameters

APPLID
is the application identifier for the CICS region.

SIGNON_TYPE
is the type of signon for the userid (specified by the USERID value).

Values for the parameter are:

ATTACH_SIGN_ON
DEFAULT_SIGN_ON
IRC_SIGN_ON
LU61_SIGN_ON
LU62_SIGN_ON
NON_TERMINAL_SIGN_ON
PRESET_SIGN_ON
USER_SIGN_ON
XRF_SIGN_ON

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

ENTRY_PORT_NAME
Optional Parameter

is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

ENTRY_PORT_TYPE
Optional Parameter

is the type of the optional entry port to be assigned to the userid (specified by the USERID value). This parameter is only valid if ENTRY_PORT_NAME is also specified.

Values for the parameter are:
```
CONSOLE
TERMINAL
```

GROUPID
Optional Parameter

is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

GROUPID_LENGTH
Optional Parameter

is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:
```
APPLICATION_NOTAUTH
ENTRY_PORT_NOTAUTH
ESM_INACTIVE
ESM_TRANQUIL
GETMAIN_FAILURE
GROUP_ACCESS_REVOKED
INVALID_GROUPID
INVALID_USERID
SECLABEL_FAILURE
SECURITY_INACTIVE
UNKNOWN_ESM_ERROR
USERID_NOT_DEFINED
USERID_NOT_IN_GROUP
USERID_REVOKED
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SECURITY_TOKEN
is the token identifying the userid.
ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.
SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSAD gate, DELETE_USER_SECURITY function

The DELETE_USER_SECURITY function of the XSAD gate is used to delete the storage held to store the ACEE and ACEE pointer for the user represented by the security token.

Input Parameters

SECURITY_TOKEN
is the token identifying the userid.
SIGNOFF_TYPE
is the type of signoff for the userid identified by the SECURITY_TOKEN value.

Values for the parameter are:
```
ATTACH_SIGN_OFF
DEFERRED_SIGN_OFF
LINK_SIGN_OFF
NON_TERMINAL_SIGN_OFF
PRESET_SIGN_OFF
TIMEOUT_SIGN_OFF
UNFLATTEN_USER_SIGN_OFF
USER_SIGN_OFF
USRDELAY_SIGN_OFF
XRF_SIGN_OFF
```

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:
```
ESM_INACTIVE
ESM_TRANQUIL
INVALID_SECURITY_TOKEN
SECURITY_INACTIVE
SECURITY_TOKEN_IN_USE
UNKNOWN_ESM_ERROR
```

The following values are returned when RESPONSE is INVALID:
```
INVALID_FORMAT
INVALID_FUNCTION
```
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSAD gate, INQUIRE_USER_ATTRIBUTES function

The INQUIRE_USER_ATTRIBUTES function of the XSAD gate is used to inquire about the attributes of the user represented by the security token.

Input Parameters

SECURITY_TOKEN
is the token identifying the userid.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
ESTAE_FAILURE
EXTRACT_FAILURE
INVALID_ACEE
INVALID_ESM_PARAMETER
INVALID_SECURITY_TOKEN
NOTAUTH
PROFILE_UNKNOWN
SECURITY_INACTIVE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ACEE_PTR
Optional Parameter

is a pointer to the access control environment element, the control block that is generated by an external security manager (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER's ACEEis returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

CURRENT_GROUPID
Optional Parameter

is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

CURRENT_GROUPID_LENGTH
Optional Parameter

is the 8-bit length of the GROUPID value.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

NATIONAL_LANGUAGE
Optional Parameter

is a three-character code identifying the national language for the userid. It can have any of the values in “National language codes (three-characters).”

OPCLASS
Optional Parameter

is the operator class, in the range 1 through 24, for the userid.

OPIDENT
Optional Parameter

is the operator identification code, 1 through 3 alphanumeric characters, for the userid.

OPPRTY
Optional Parameter

is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the userid.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

TIMEOUT
Optional Parameter

is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.

1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

USERID
Optional Parameter

is the identifier of the user (a userid of 1 through 10 alphanumeric characters). the userid (specified by the SECURITY_TOKEN value) is assigned.

USERID_LENGTH
Optional Parameter

is the length of the USERID value.

USERNAME
Optional Parameter

is an optional buffer into which the attributes of the user are placed.

XRFSOFF
Optional Parameter

indicates whether or not you want CICS to sign off the userid following an XRF takeover.

Values for the parameter are:

```
FORCE
NOFORCE
```

National language codes (three-characters)

Code	Language Name	Original Name
AFR	Afrikaans	Afrikaans
ARA	Arabic	Arabi
BEL	Byelorussian	Belaruskaja (mova)
BGR	Bulgarian	Bulgarski
CAT	Catalan	Catala
CHT	Traditional Chinese	Zhongwen
CHS	Simplified Chinese	

Code	Language Name	Original Name
CSY	Czech	Cesky
DAN	Danish	Dansk
DEU	German	Deutsch
DES	Swiss German	Schweizer-Deutsch
ELL	Greek	Ellinika
ENA	Australian English	
ENG	UK English	English
ENU	US English	
ENP	English Upper Case	
ESP	Spanish	Espanol
FAR	Farsi	Persian
FIN	Finnish	Suomi
FRA	French	Francais
FRB	Belgian French	
FRC	Canadian French	
FRS	Swiss French	Suisse-francais
GAE	Irish Gaelic (Irish)	Gaeilge
HEB	Hebrew	Ivrith
HRV	Croatian	Hrvatski
HUN	Hungarian	Magyar
ISL	Icelandic	Islenska
ITA	Italian	Italiano
ITS	Swiss Italian	Italiano svizzero
JPN	Japanese	Nihongo
KOR	Korean	Choson-o; Hanguk-o
MKD	Macedonian	Makedonski
NLD	Dutch	Nederlands
NLB	Belgian Dutch	
NOR	Norwegian - Bokmal	Norsk - Bokmal
NON	Norwegian - Nynorsk	Norsk - Nynorsk
PLK	Polish	Polski
PTG	Portuguese	Portugues
PTB	Brazilian Portuguese	
RMS	Rhaeto-Romanic	Romontsch
ROM	Romanian	Romana
RUS	Russian	Russkij
SHC	Serbo-Croatian (Cyr)	Srpsko-hrvatski
SHL	Serbo-Croatian (Lat)	
SKY	Slovakian	Slovensky
SLO	Slovenian	Slovenski
SRL	Serbian (Latin)	Srpski (Latin)
SRB	Serbian	Srpski
SQI	Albanian	Shqip
SVE	Swedish	Svenska
THA	Thai	Thai
TRK	Turkish	Turkce
UKR	Ukrainian	Ukrainska (mova)
URD	Urdu	Urdu

XSAD gate, VALIDATE_USERID function

The VALIDATE_USERID function of the XSAD gate is used to check whether the specified userid is valid. It is used especially when the userid has to be validated without the user being added to the system; usually because the userid was

specified in a deferred START command, and the user does not need to be added to the system until the started task begins to execute.

Input Parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
GROUP_ACCESS_REVOKED
SECURITY_INACTIVE
USERID_NOT_DEFINED
USERID_NOT_DETERMINED
USERID_REVOKED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSAD gate, ADD_USER_VIA_ICRX function

The ADD_USER_VIA_ICRX function of the XSAD gate requests the External Security Manager to add a user, using ICRX details.

Input Parameters

ICRX
Is the extended identity context reference (ICRX) of the user.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
APPLICATION_NOTAUTH
ENTRY_PORT_NOTAUTH
ESM_TRANQUIL
ESM_INACTIVE
GETMAIN_FAILURE
GROUP_ACCESS_REVOKED
INVALID_GROUPID
SECLABEL_FAILURE
SECURITY_INACTIVE
UNKNOWN_ESM_ERROR
USERID_REVOKED
USERID_NOT_DEFINED
INVALID_USERID
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SECURITY_TOKEN
Is the token identifying the ICRX in the user domain.

ESM_RESPONSE
Optional parameter

Is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional parameter

Is the optional 32-bit SAF response code to the call.

XSAD gate, INQUIRE_ICRX function

The INQUIRE_ICRX function of the XSAD gate retrieves an ICRX from the External Security Manager.

Input Parameters

SECURITY_TOKEN
Is the token identifying the ICRX in the user domain.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
ESTAE_FAILURE
EXTRACT_FAILURE
INVALID_ACEE
INVALID_ESM_PARAMETER
INVALID_SECURITY_TOKEN
NOTAUTH
PROFILE_UNKNOWN
SECURITY_INACTIVE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ICRX
Is the extended identity context reference (ICRX) of the user.

ESM_RESPONSE
Optional parameter

Is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional parameter

Is the optional 32-bit SAF response code to the call.

XSAD gate, RELEASE_ICRX function

The RELEASE_ICRX function of the XSAD gate requests that the External Security Manager removes an ICRX that is no longer required.

Input Parameters

SECURITY_TOKEN
Is the token identifying the ICRX in the user domain.

ICRX
Is the extended identity context reference (ICRX) of the user.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
ESM_INACTIVE
ESM_TRANQUIL
INVALID_SECURITY_TOKEN
SECURITY_INACTIVE
SECURITY_TOKEN_IN_USE
UNKNOWN_ESM_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE
Optional parameter

Is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional parameter

Is the optional 32-bit SAF response code to the call.

XSAD gate, RELEASE_ICRX_STORAGE function

The RELEASE_ICRX_STORAGE function of the XSAD gate requests that the virtual storage associated with an ICRX is made available.

Input Parameters

SECURITY_TOKEN
Is the token identifying the ICRX in the user domain.

ICRX
Is the extended identity context reference (ICRX) or distributed identity of the user.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following value is returned when RESPONSE is EXCEPTION:

```
INVALID_SECURITY_TOKEN
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
INVALID_ICRX
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSCT gate, INQUIRE_CERTIFICATE function

The INQUIRE_CERTIFICATE function extracts data fields out of an X-509 certificate.

Input Parameters

CERTIFICATE

Optional Parameter

On input, contains a full DER-encoded X-509 certificate. Alternatively, CERTIFICATE_LABEL can be used to identify a certificate in the keyring. If neither is specified, the default certificate in the key ring is used. On output, contains the certificate from which the data is extracted.

CERTIFICATE_LABEL

Optional Parameter

Identifies a certificate in the keyring

COMMON_NAME

Optional Parameter

A buffer in which the common name contained within the certificate is returned.

DISTINGUISHED_NAME

Optional Parameter

A buffer in which the BER-encoded distinguished name from the certificate is returned.

EMAIL_ADDRESS

Optional Parameter

A buffer in which the e-mail address contained within the certificate is returned.

FOR

Optional Parameter

Specifies from which of the distinguished names in the certificate the data is to be extracted.

Values for the parameter are:

```
ISSUER
SUBJECT
```

LOCALITY

Optional Parameter

A buffer in which the locality contained within the certificate is returned.

ORGANIZATION

Optional Parameter

A buffer in which the organization contained within the certificate is returned.

ORGANIZATIONAL_UNIT

Optional Parameter

A buffer in which the organizational unit contained within the certificate is returned.

SERIAL_NUMBER
Optional Parameter

A buffer in which the serial number of the certificate is returned.

STATE_OR_PROVINCE
Optional Parameter

A buffer in which the state or province unit contained within the certificate is returned.

TITLE
Optional Parameter

A buffer in which the title contained within the certificate is returned.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
SEVERE_ERROR
```

The following values are returned when RESPONSE is EXCEPTION:

```
CERTIFICATE_INVALID
CERTIFICATE_NOT_FOUND
ESM_INACTIVE
GETMAIN_FAILED
KEYRING_NOT_FOUND
NOTAUTH
REVOCATION_LIST_INVALID
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE
Optional Parameter

The external security manager's response to the call.

SAF_RESPONSE
Optional Parameter

The system authorization facility's response to the call.

STATUS
Optional Parameter

The status of the certificate.

Values for the parameter are:

```
EXPIRED
NOT_OWNER
NOT_YET_CURRENT
TRUSTED
UNREGISTERED
UNTRUSTED
```

USAGE
Optional Parameter

The intended usage of the certificate, as recorded by the External Security Manager.

Values for the parameter are:
CERTAUTH
PERSONAL
SITE

USERID
Optional Parameter

The user ID of the certificate's owner.

USERID_LENGTH
Optional Parameter

The length of the user ID field.

VALID_FROM_ABSTIME
Optional Parameter

The date and time from when the certificate is valid (in CICS ABSTIME format).

VALID_UNTIL_ABSTIME
Optional Parameter

The date and time until when the certificate is valid (in CICS ABSTIME format).

XSCT gate, INQUIRE_REVOCATION_LIST function

The INQUIRE_REVOCATION_LIST function extracts data fields out of a Certificate Revocation List.

Input Parameters

REVOCATION_LIST
The certificate revocation list from which data is to be extracted.

DISTINGUISHED_NAME
Optional Parameter

A buffer in which the distinguished name of the issuer of the revocation list is returned.

Output Parameters

REASON
The values for the parameter are:
ABEND
LOOP
REVOCATION_LIST_INVALID

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CURRENT_ISSUE_ABSTIME
Optional Parameter

The date and time that this revocation list was issued (in CICS ABSTIME format).

NEXT_ISSUE_ABSTIME
Optional Parameter

The date and time that the next revocation list is due to be issued (in CICS ABSTIME format).

XSEJ gate, ADD_REPL_ROLE_FOR_METHOD function

Add a specified role for a specified method within the CORBASERVER to the in storage look up table.

Input Parameters

BEAN_NAME
The name of the bean.

CORBASERVER
The name of the CORBASERVER.

METHOD_AND_SIGNATURE
The method and signature for which the role is to be added.

ROLE_NAME
The role name to be added.

APPLICATION_NAME
Optional Parameter

An application name that qualifies the role name.

INTERFACE_TYPE
Optional Parameter

The type of interface.

Values for the parameter are:
HOME
REMOTE

Output Parameters

REASON
The values for the parameter are:
ABEND
GETMAIN_FAILED
INVALID_ROLE_NAME
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSEJ gate, CHECK_CALLER_IN_ROLE function

Checks whether the user associated with the current transaction is defined to be in the named role.

Input Parameters

BEAN_NAME
The bean name for which the check is being made.

CODED_ROLE_NAME
The name of the coded role.

CORBASERVER
The CORBASERVER for which the check is being made.

APPLICATION_NAME
Optional Parameter

An application name that qualifies the bean name.

LOGMESSAGE
Optional Parameter

Specifes whether access failures are to be logged to the CSCS TD queue and the MVS System Management Facility (SMF). The default is YES.

Values for the parameter are:
NO
YES

Output Parameters

REASON

The values for the parameter are:

```
ABEND
ESM_INACTIVE
LOOP
NOT_IN_ROLE
NOTAUTH
```

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE

Optional Parameter

The external security manager's response to the call.

FAILING_USERID

Optional Parameter

The user ID for which the check failed.

FAILING_USERID_LENGTH

Optional Parameter

The length of the user ID for which the check failed.

SAF_RESPONSE

Optional Parameter

The system authorization facility's response to the call.

XSEJ gate, CHECK_EJB_METHOD function

Check whether the user associated with the current transaction is authorized to invoke the specified method of the named bean.

Input Parameters

BEAN_NAME

The name of the bean for which the check is being made.

CORBASERVER

The name of the CORBASERVER for which the check is being made.

METHOD_AND_SIGNATURE

The method and signature name for which the check is being made.

APPLICATION_NAME

Optional Parameter

An application name that qualifies the bean name.

INTERFACE_TYPE

Optional Parameter

The type of interface.

Values for the parameter are:

```
HOME
REMOTE
```

LOGMESSAGE

Optional Parameter

Specifes whether access failures are to be logged to the CSCS TD queue and the MVS System Management Facility (SMF). The default is `YES`.

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON
The values for the parameter are:
ABEND
ESM_INACTIVE
LOOP
NOTAUTH

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE
Optional Parameter

The external security manager's response to the call.

FAILING_USERID
Optional Parameter

The user ID for which the check failed.

FAILING_USERID_LENGTH
Optional Parameter

The length of the user ID for which the check failed.

SAF_RESPONSE
Optional Parameter

The system authorization facility's response to the call.

XSEJ gate, DELETE_BEAN_SECURITY function

Delete all entries at the bean level from the in-storage lookup table. This includes all method and coded_role entries belonging to the specified bean.

Input Parameters

BEAN_NAME
The name of the bean.

CORBASERVER
The name of the CORBASERVER.

Output Parameters

REASON
The values for the parameter are:
ABEND
LOOP

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSEJ gate, INQUIRE_DISTINGUISHED_NAME function

Obtains the sub-fields of the distinguished name from the certificate identified by its label in the key ring.

Input Parameters

CERTIFICATE_LABEL
Optional Parameter

The label that identifies the certificate.

COMMON_NAME
Optional Parameter

A buffer in which the common came contained within the certificate is returned.

EMAIL_ADDRESS
Optional Parameter

A buffer in which the e-mail address contained within the certificate is returned.

LOCALITY
Optional Parameter

A buffer in which the locality contained within the certificate is returned.

ORGANIZATION
Optional Parameter

A buffer in which the organization contained within the certificate is returned.

ORGANIZATIONAL_UNIT
Optional Parameter

A buffer in which the organizational unit contained within the certificate is returned.

STATE_OR_PROVINCE
Optional Parameter

A buffer in which the organizational unit contained within the certificate is returned.

TITLE
Optional Parameter

A buffer in which the title contained within the certificate is returned.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
CERTIFICATE_INVALID
CERTIFICATE_NOT_FOUND
ESM_INACTIVE
KEYRING_NOT_FOUND
LOOP
SEVERE_ERROR
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

COUNTRY
Optional Parameter

The country name contained in the certificate.

ESM_RESPONSE
Optional Parameter

The external security manager's response to the call.

SAF_RESPONSE
Optional Parameter

The system authorization facility's response to the call.

XSEJ gate, INQUIRE_HASH_CODE function

This function returns a unique hash code to represent the Principal.

Output Parameters

REASON
The values for the parameter are:
ABEND
LOOP

HASH_CODE
The intended hash code value.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSEJ gate, INQUIRE_PRINCIPAL function

This function obtains information for creating a Java Principal object and building its distinguished name.

Input Parameters

CLIENT_CERTIFICATE
Optional Parameter

On input, contains a full DER-encoded X-509 certificate. Alternatively, CERTIFICATE_LABEL can be used to identify a certificate in the keyring. If neither is specified, the default certificate in the key ring is used. On output, contains the certificate from which the data is extracted.

CERTIFICATE_LABEL
Optional Parameter

Identifies a certificate in the keyring

COMMON_NAME
Optional Parameter

A buffer in which the common name contained within the certificate is returned.

DISTINGUISHED_NAME
Optional Parameter

A buffer in which the distinguished name in RFC2253 format is returned if the DISTINGUISHED_NAME_URM para,mater is specified.

EMAIL_ADDRESS
Optional Parameter

A buffer in which the e-mail address contained within the certificate is returned.

LOCALITY
Optional Parameter

A buffer in which the locality contained within the certificate is returned.

ORGANIZATION
Optional Parameter

A buffer in which the organization contained within the certificate is returned.

ORGANIZATIONAL_UNIT
Optional Parameter

A buffer in which the organizational unit contained within the certificate is returned.

STATE_OR_PROVINCE
Optional Parameter

A buffer in which the organizational unit contained within the certificate is returned.

TITLE
Optional Parameter

A buffer in which the title contained within the certificate is returned.

DISTINGUISHED_NAME_URM
Optional Parameter

The name of a user-replaceable module that is called to create a distinguished name string.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
CERTIFICATE_INVALID
CERTIFICATE_NOT_FOUND
ESM_INACTIVE
KEYRING_NOT_FOUND
LOOP
SEVERE_ERROR
URM_FAILED
```

COUNTRY
The country name contained in the certificate.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USERID
Optional Parameter

The user ID of the certificate's owner.

USERID_LENGTH
Optional Parameter

The length of the user ID field.

USERNAME
The name of the user as defined in the external security manager.

ESM_RESPONSE
Optional Parameter

The external security manager's response to the call.

SAF_RESPONSE
Optional Parameter

The system authorization facility's response to the call.

XSEJ gate, SET_ROLE_FOR_CODED_ROLE function

Populates a lookup table indexed by CORBASERVER, adding a role for the coded_role names for a bean installed in a CORBASERVER.

Input Parameters

BEAN_NAME
The name of the bean.

CODED_ROLE_NAME
The coded role name.

CORBASERVER
The name of the CORBASERVER.

ROLE_NAME
The role name.

APPLICATION_NAME
Optional Parameter

An application name that qualifies the bean name.

Output Parameters

REASON
The values for the parameter are:

```
ABEND
INVALID_ROLE_NAME
LOOP
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSFL gate, FLATTEN_USER_SECURITY function

The FLATTEN_USER_SECURITY function of the XSFL gate is used to flatten the user's security state and place into the FLATTENED_SECURITY buffer provided.

Input Parameters

FLATTENED_SECURITY
is the buffer into which the flattened security state is placed.
SECURITY_TOKEN
is the token identifying the userid.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
ESM_ABENDED
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
INVALID_SECURITY_TOKEN
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FLATTENED_BUFFER
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.
SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSFL gate, UNFLATTEN_ESM_UTOKEN function

The UNFLATTEN_ESM_UTOKEN function of the XSFL gate returns userid and groupid information associated with the external security manager's user token.

Input Parameters

ESM_UTOKEN_PTR
is a pointer to a security manager user pointer.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
ESM_ABENDED
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
APPLID_NOTAUTH
ENTRY_PORT_NOTAUTH
ESM_INACTIVE
ESM_TRANQUIL
GETMAIN_FAILED
GROUP_ACCESS_REVOKED
SECLABEL_CHECK_FAILED
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE
USERID_NOT_DEFINED
USERID_NOT_IN_GROUP
USERID_REVOKED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FLATTENED_BUFFER
INVALID_FORMAT
INVALID_FUNCTION
```

CURRENT_GROUPID
is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

CURRENT_GROUPID_LENGTH
is the 8-bit length of the GROUPID value.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters). the userid (specified by the SECURITY_TOKEN value) is assigned.

USERID_LENGTH
is the length of the USERID value.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSFL gate, UNFLATTEN_USER_SECURITY function

The UNFLATTEN_USER_SECURITY function of the XSFL gate is used to unflatten the user security state data in the FLATTENED_SECURITY buffer, and add the userid to the security domain.

Input Parameters

FLATTENED_SECURITY
is the buffer into which the flattened security state is placed.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
ESM_ABENDED
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
APPLID_NOTAUTH
ENTRY_PORT_NOTAUTH
ESM_INACTIVE
ESM_TRANQUIL
GETMAIN_FAILED
GROUP_ACCESS_REVOKED
SECLABEL_CHECK_FAILED
SECURITY_INACTIVE
UNKNOWN_ESM_RESPONSE
USERID_NOT_DEFINED
USERID_NOT_IN_GROUP
USERID_REVOKED
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FLATTENED_BUFFER
INVALID_FORMAT
INVALID_FUNCTION
```

ACEE_PTR
is a pointer to the access control environment element, the control block that is generated by an external security manager (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER's ACEEis returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

CURRENT_GROUPID
is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

CURRENT_GROUPID_LENGTH
is the 8-bit length of the GROUPID value.

ENTRY_PORT_NAME
is the name of an entry port, 1 through 8 alphanumeric characters, for the userid.

ENTRY_PORT_TYPE
is the type of the entry port for the userid.

Values for the parameter are:

```
CONSOLE
NULL
TERMINAL
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SECURITY_TOKEN
is the token identifying the userid.

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters). the userid (specified by the SECURITY_TOKEN value) is assigned.

USERID_LENGTH
is the length of the USERID value.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSIS gate, INQ_SECURITY_DOMAIN_PARMS function

The INQ_SECURITY_DOMAIN_PARMS function of the XSIS gate is used to return the current values of parameters from the security state data.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

APPLID
Optional Parameter

is the generic applid of the CICS region

CMDSEC
Optional Parameter

indicates whether or the CICS region should obey the CMDSEC option specified on a transaction's resource definition.

Values for the parameter are:

```
ALWAYS
ASIS
```

EJBROLE_PREFIX
Optional Parameter

is the prefix that is used to qualify the security role defined in an enterprise bean's deployment descriptor.

ESMEXITS
Optional Parameter

indicates whether or not installation data is to be passed via the RACROUTE interface to the ESM for use in user exits written for the ESM.

Values for the parameter are:

```
NO
YES
```

KEYRING
Optional Parameter

is the fully qualified name of the key ring that contains the keys and X.509 certificates used to support the secure sockets layer (SSL).

PREFIX
Optional Parameter

returns the value of the prefix that is being applied to all resource names in authorization requests sent to the external security manager. It can contain 0 through 8 alphanumeric characters.

PSBCHK
Optional Parameter

indicates whether or not DL/I security checking is to be performed for a remote terminal initiating a transaction with transaction routing.

Values for the parameter are:
NO
YES

RESSEC
Optional Parameter

indicates whether the CICS region should obey the RESSEC option specified on a transaction's resource definition.

Values for the parameter are:
ALWAYS
ASIS

SECURITY
Optional Parameter

indicates whether or not security is active for this CICS region.

Values for the parameter are:
NO
YES

XAPPC
Optional Parameter

indicates whether or not session security checking is used when establishing APPC sessions.

Values for the parameter are:
NO
YES

XCMD
Optional Parameter

indicates whether or not the ESM performs security checking for EXEC CICS commands.

Values for the parameter are:
NO
YES
name where *name* is the resource class name for EXEC CICS commands.

XDB2
Optional Parameter

indicates whether or not CICS performs DB2ENTRY security checking.

Values for the parameter are:
NO
YES
name where *name* is the resource class name for DB2 entries.

XDCT
Optional Parameter

indicates whether or not the ESM performs security checking for transient data queues.

Values for the parameter are:
NO
YES
name where *name* is the resource class name for transient data queues.

XFCT
Optional Parameter

indicates whether or not the ESM performs security checking for files.

Values for the parameter are:
NO
YES
name where *name* is the resource class name for files.

XJCT
Optional Parameter

indicates whether or not the ESM performs security checking for journals.

Values for the parameter are:
NO
YES
name where *name* is the resource class name for journals.

XPCT
Optional Parameter

indicates whether or not the ESM performs security checking for CICS transactions started by an EXEC CICS START command (started transactions).

Values for the parameter are:
NO
YES
name where *name* is the resource class name for started transactions.

XPPT
Optional Parameter

indicates whether or not the ESM performs security checking for programs.

Values for the parameter are:
NO
YES
name where *name* is the resource class name for programs.

XPSB
Optional Parameter

indicates whether or not the ESM performs security checking for DL/I program specification blocks (PSBs).

Values for the parameter are:
NO
YES
name where *name* is the resource class name for PSBs.

XTRAN
Optional Parameter

indicates whether or not the ESM performs security checking for transactions initiated at a CICS terminal (transaction-attach).

Values for the parameter are:
NO
YES
name where *name* is the resource class name for transactions initiated at a CICS terminal.

XTST
Optional Parameter

indicates whether or not the ESM performs security checking for temporary storage queues.

Values for the parameter are:
NO
YES
name where *name* is the resource class name for temporary storage queues.

XUSER
Optional Parameter

indicates whether or not the ESM performs security checking for surrogate users.

Values for the parameter are:
NO
YES
name where *name* is the resource class name for surrogate users.

XSIS gate, INQUIRE_REALM_NAME function

Obtains the realm names under which the CICS system is executing; a realm is an environment in which a userid and password pairing is valid.

Input Parameters

REALM_TYPE
Indicates that the request is for the Basic realm name.

Values for the parameter are:
BASIC
KERBEROS

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
LOOP

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

REALM_NAME
Returns the name of the realsm under which CICS is executing.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSIS gate, INQUIRE_REGION_USERID function

The INQUIRE_REGION_USERID function of the XSIS gate is used to return the userid and groupid associated with the jobstep that is currently executing this CICS region.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

REGION_USERID

is the user identifier of the CICS jobstep (a userid of 1 through 8 alphanumeric characters).

REGION_USERID_LENGTH

is the length of the REGION_USERID value.

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

REGION_GROUPID

Optional Parameter

is the identifier, 1 through 8 alphanumeric characters, of the current RACF user group to which the region userid is assigned.

REGION_GROUPID_LENGTH

Optional Parameter

is the 8-bit length of the REGION_GROUPID value.

XSIS gate, SET_NETWORK_IDENTIFIER function

When CICS issues an OPEN ACB for z/OS Communications Server, the CICS SVC is invoked to store the name (netid) of the local network combined with the local luname, and to RACLIST the profiles in the External Security Manager (ESM) APPCLU Class. If you have specified either of the SEC=NO or XAPPC=NO system initialization parameters, no action is performed, and the return code is set to OK.

Input Parameters

CONDITIONAL

indicates whether or not CICS can tolerate errors in XSIS calls due to the APPCLU profiles not being in storage (LU6.2 connections cannot be validated).

Values for the parameter are:

```
NO
YES
```

LOCAL_LUNAME

is the z/OS Communications Server LU name of the local CICS region.

LOCAL_LUNAME_LENGTH

is the length of the z/OS Communications Server LU name specified by LOCAL_LUNAME.

Output Parameters

REASON

The following values are returned when RESPONSE is DISASTER:

```
ABEND
LOOP
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSIS gate, SET_SECURITY_DOMAIN_PARMS function

At CICS startup, loads information for the security domain from the system initialization table (SIT) into the security state data.

Input Parameters

APPLID
The application identifier for the CICS region.

ESMEXITS
Determines whether installation data is to be passed via the RACROUTE interface to the ESM for use in user exits written for the ESM.

Values:
NO
YES

PSBCHK
Determines whether DL/I security checking is to be performed for a remote terminal initiating a transaction with transaction routing.

Values:
NO
YES

SECURITY
Determines whether security is active for this CICS region.

Values:
NO
YES

XAPPC
Determines whether session security checking is used when establishing APPC sessions.

Values:
NO
YES

CMDSEC
Optional

Determines whether or the CICS region uses the CMDSEC option specified on a transaction resource definition.

Values:
ALWAYS
ASIS

EJBROLE_PREFIX
Optional

Prefix that qualifies the security role defined in an enterprise bean deployment descriptor.

KEYRING
Optional

Fully qualified name of the key ring that contains the keys and X.509 certificates used to support the secure sockets layer (SSL).

PREFIX
Optional

Prefix applied to the resource name in any authorization requests sent to the external security manager.The value is 1 through 8 alphanumeric characters, or the single character '*', which indicates that the CICS region user ID is to be used as the prefix.

RESSEC

Optional

Determines whether the CICS region uses the RESSEC option specified on a transaction resource definition.

Values:

ALWAYS
ASIS

XCMD

Optional

Determines whether EXEC CICS commands are checked by the ESM.

Values:

NO
YES
name where *name* is the resource class name for EXEC CICS commands.

XDB2

Optional

Determines whether CICS performs DB2ENTRY security checking.

Values:

NO
YES
name where *name* is the resource class name for DB2 entries.

XDCT

Optional

Determines whether destination control entries are checked by the ESM.

Values:

NO
YES
name where *name* is the resource class name for destination control entries.

XFCT

Optional

Determines whether file control entries are checked by the ESM.

Values:

NO
YES
name where *name* is the resource class name for file control entries.

XJCT

Optional

Determines whether journal entries are checked by the ESM.

Values:

NO
YES
name where *name* is the resource class name for journal entries.

XPCT

Optional

Determines whether EXEC-started transactions entries are checked by the ESM.

Values:
NO
YES
name where *name* is the resource class name for EXEC-started transaction entries.

XPPT
Optional

Determines whether program entries are checked by the ESM.

Values:
NO
YES
name where *name* is the resource class name for program entries.

XPSB
Optional

Determines whether PSB entries are checked by the ESM.

Values:
NO
YES
name where *name* is the resource class name for PSB entries.

XTRAN
Optional

Determines whether attached transaction entries are checked by the ESM.

Values:
NO
YES
name where *name* is the resource class name for attached transaction entries.

XTST
Optional

Determines whether temporary storage entries are checked by the ESM.

Values:
NO
YES
name where *name* is the resource class name for temporary storage entries.

XUSER
Optional

Determines whether user entries are checked by the ESM.

Values:
NO
YES
name where *name* is the resource class name for user entries.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
ABEND
CWA_WAIT_PHASE_FAILURE
INQUIRE_CWA_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:
GETMAIN_FAILED

KEYRING_NOT_FOUND
KEYRING_NOTAUTH

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE
Optional

32-bit ESM response code that is returned to the call.

SAF_RESPONSE
Optional

32-bit SAF response code that is returned to the call.

XSIS gate, SET_SPECIAL_TOKENS function

The SET_SPECIAL_TOKENS function of the XSIS gate sets the security tokens for the default user ID and the region user ID.

Input Parameters

DEFAULT_SECURITY_TOKEN
The security token for the default user ID.

REGION_SECURITY_TOKEN
The security token for the region user ID.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSLU gate, GENERATE_APPC_BIND function

The GENERATE_APPC_BIND function of the XSLU gate generates a random number which is sent to the partner LU for partner verification.

Output Parameters

REASON
The following values are returned when RESPONSE is EXCEPTION:
BINDSECURITY_INACTIVE
SECURITY_INACTIVE

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RANDOM_STRING
A random eight-character string.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSLU gate, GENERATE_APPC_RESPONSE function

The GENERATE_APPC_RESPONSE function of the XSLU gate encrypts the string received from the LU partner, and generates a new random string for the partner to validate.

Input Parameters

LOCAL_LUNAME
is the z/OS Communications Server LU name of the local CICS region.

REMOTE_LUNAME
is the z/OS Communications Server LU name of the remote CICS region (that sent the bind).

TEST_STRING
is a random eight-character string receive with a bind request (RANDOM_STRING of the GENERATE_APPC_BIND function).

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
ESM_ABENDED
ESTAE_FAILURE
EXTRACT_FAILURE
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
BINDSECURITY_INACTIVE
NOTAUTH
PROFILE_EXPIRED
PROFILE_LOCKED
PROFILE_UNKNOWN
SECURITY_INACTIVE
SECURITY_INACTIVE
SESSION_KEY_NULL
UNKNOWN_ESM_RESPONSE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

ENCRYPTED_TEST_STRING
is an eight-character string formed by encrypting the test string using shared DES (Data Encryption Standard/System) encryption keys.

RANDOM_STRING
A random eight-character string.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSLU gate, VALIDATE_APPC_RESPONSE function

The VALIDATE_APPC_RESPONSE function of the XSLU gate encrypts the string that was previously sent to the partner, and compares it with the encrypted string received from the partner.

Input Parameters

ENCRYPTED_TEST_STRING
is an eight-character string formed by encrypting the test string using shared DES (Data Encryption Standard/System) encryption keys.

LOCAL_LUNAME
is the z/OS Communications Server LU name of the local CICS region.

REMOTE_LUNAME
is the z/OS Communications Server LU name of the remote CICS region (that sent the bind).

TEST_STRING
is a random eight-character string receive with a bind request (RANDOM_STRING of the GENERATE_APPC_BIND function).

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
ESM_ABENDED
ESTAE_FAILURE
EXTRACT_FAILURE
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
BINDSECURITY_INACTIVE
NOTAUTH
PROFILE_EXPIRED
PROFILE_LOCKED
PROFILE_UNKNOWN
SECURITY_INACTIVE
SESSION_KEY_NULL
UNKNOWN_ESM_RESPONSE
VALIDATION_ERROR
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSPW gate, CREATE_PASSTICKET function

The CREATE_PASSTICKET function of the XSPW gate is used to create a RACF PassTicket (an alternative to a password). When created, the RACF PassTicket can be presented for userid verification once only.

Input Parameters

APPLID
is the application identifier for the CICS region.

TRANSACTION_NUMBER
Optional Parameter

is an optional number that identifies a transaction from which the caller's security token is located. If not specified, the caller's security token is located from the principal security token associated with the current CICS task.

Output Parameters

ESM_REASON
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

ESM_RESPONSE
is the optional 32-bit ESM response code to the call.

PASSTICKET
is the 10-character passticket to be used for the CICS region specified by the APPLID value.

PASSTICKET_LENGTH
is the 8-bit length of the PASSTICKET value.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSPW gate, INQUIRE_CERTIFICATE_USERID function

The INQUIRE_CERTIFICATE_USERID function of the XSPW gate obtains the userid associated with an X.509 certificate that has been installed into the External Security Manager.

Input Parameters

CERTIFICATE
an X.509 certificate

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

```
ABEND
ESM_ABENDED
ESTAE_FAILURE
EXTRACT_FAILURE
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:

```
ESM_INACTIVE
FREEMAIN_FAILED
GETMAIN_FAILED
INVALID_CERTIFICATE
LENGTH_ERROR
NOTAUTH
SECURITY_INACTIVE
UNKNOWN_CERTIFICATE
UNKNOWN_ESM_ERROR
UNTRUSTED_CERTIFICATE
```

The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

ESM_REASON
is the optional 32-bit ESM reason returned with ESM_RESPONSE.
ESM_RESPONSE
is the optional 32-bit ESM response code to the call.
RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.
USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters). the userid (specified by the SECURITY_TOKEN value) is assigned.
USERID_LENGTH
is the length of the USERID value.

XSPW gate, INQUIRE_PASSWORD_DATA function

The INQUIRE_PASSWORD_DATA function of the XSPW gate provides information from the ESM.

Input Parameters

PASSWORD
is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.
PASSWORD_LENGTH
is the 8-bit length of the PASSWORD value. This parameter is only valid if PASSWORD is also specified.
USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.
USERID_LENGTH
is the length of the USERID value.
PASSWORD_TYPE
Optional Parameter

specifies if the password is masked.

Values for the parameter are:
```
CLEAR
MASKED
```

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
```
ABEND
ESM_ABENDED
ESTAE_FAILURE
EXTRACT_FAILURE
LOOP
```

The following values are returned when RESPONSE is EXCEPTION:
```
APPLID_NOTAUTH
ESM_INACTIVE
GROUP_CONNECTION_REVOKED
NOTAUTH
PASSWORD_EXPIRED
PASSWORD_NOTAUTH
SECURITY_INACTIVE
UNKNOWN_ESM_ERROR
USERID_FORMAT_ERROR
USERID_REVOKED
```

USERID_UNDEFINED

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

CHANGE_ABSTIME
Optional Parameter

is the date and time of when the password was last changed.

DAYS_LEFT
Optional Parameter

is the number of days remaining before the password must be changed.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

EXPIRY_ABSTIME
Optional Parameter

is the date and time of when the password will expire.

LASTUSE_ABSTIME
Optional Parameter

is the date and time of when the password was last used.

PASSWORD_FAILURES
Optional Parameter

is the number of times that the user has unsuccessfully entered tried to enter the password.

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSPW gate, REGISTER_CERTIFICATE_USER function

The REGISTER_CERTIFICATE_USER function of the XSPW gate associates a user with an X.509 certificate that has been installed into the External Security Manager.

Input Parameters

CERTIFICATE
an X.509 certificate

PASSWORD
is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

PASSWORD_LENGTH
is the 8-bit length of the PASSWORD value. This parameter is only valid if PASSWORD is also specified.

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
ESM_ABENDED
ESTAE_FAILURE
EXTRACT_FAILURE
LOOP

The following values are returned when RESPONSE is EXCEPTION:

ESM_INACTIVE
FREEMAIN_FAILED
GETMAIN_FAILED
INVALID_CERTIFICATE
NOTAUTH
SECURITY_INACTIVE
UNKNOWN_CERTIFICATE
UNKNOWN_ESM_ERROR
UNTRUSTED_CERTIFICATE

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

ESM_REASON
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

ESM_RESPONSE
is the optional 32-bit ESM response code to the call.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSPW gate, UPDATE_PASSWORD function

The UPDATE_PASSWORD function of the XSPW gate assigns a new password to the userid, if the current password is input correctly and the new password meets ESM and installation defined password quality rules.

Input Parameters

NEW_PASSWORD
is the new password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

NEW_PASSWORD_LENGTH
is the 8-bit length of the NEW_PASSWORD value.

PASSWORD
is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

PASSWORD_LENGTH
is the 8-bit length of the PASSWORD value.

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) requesting the ESM information.

USERID_LENGTH
is the length of the USERID value.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:

ABEND
ESM_ABENDED
ESTAE_FAILURE
EXTRACT_FAILURE

LOOP

The following values are returned when RESPONSE is EXCEPTION:

ESM_INACTIVE
GROUP_CONNECTION_REVOKED
INVALID_NEW_PASSWORD
PASSWORD_NOTAUTH
SECLABEL_FAILURE
SECURITY_INACTIVE
UNKNOWN_ESM_ERROR
USERID_REVOKED
USERID_UNDEFINED

The following values are returned when RESPONSE is INVALID:

INVALID_FORMAT
INVALID_FUNCTION

ESM_REASON
is the external security manager's reason code.

ESM_RESPONSE
is the external security manager's response code.

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

SAF_REASON
The system authorization facility's reason code.

SAF_RESPONSE
The system authorization facility's response to the call.

XSRC gate, CHECK_CICS_COMMAND function

The CHECK_CICS_COMMAND function of the XSRC gate performs CICS command access checks.

Input Parameters

ACCESS
is the type of access to be made on the resource.

Values for the parameter are:

COLLECT
CREATE
DEFINE
DELETE
DISCARD
INQUIRE
INSTALL
PERFORM
SET

RESOURCE_TYPE
is the type of the resource.

Values for the parameter are:

AUTINSTMODEL
AUTOINSTALL
BEAN
BRFACILITY
CFDTPOOL
CONNECTION
CORBASERVER
DB2CONN

```
DB2ENTRY
DB2TRAN
DELETSHIPPED
DISPATCHER
DJAR
DOCTEMPLATE
DSNAME
DUMP
DUMPDS
ENQMODEL
EXCI
EXITPROGRAM
FEPIRESOURCE
FILE
HOST
IRBATCH
IRC
JOURNALMODEL
JOURNALNAME
JVM
JVMPROFILE
LINE
LSRPOOL
MAPSET
MODENAME
MONITOR
MVSTCB
NONVTAM
PARTITIONSET
PARTNER
PIPELINE
PROCESSTYPE
PROFILE
PROGRAM
PSB
REQID
REQUESTMODEL
RESETTIME
RRMS
SECURITY
SESSIONS
SHUTDOWN
STATISTICS
STORAGE
STREAMNAME
SUBPOOL
SYSDUMPCODE
SYSTEM
TASK
TCLASS
TCPIP
TCPIPSERVICE
TDQUEUE
TEMPSTORAGE
TERMINAL
TIME
```

TRACE
TRACEDEST
TRACEFLAG
TRACETYPE
TRANCLASS
TRANDUMPCODE
TRANSACTION
TRANSATTACH
TSMODEL
TSPOOL
TSQUEUE
TYPETERM
UOW
UOWDSNFAIL
UOWENQ
UOWLINK
URIMAP
VOLUME
VTAM
WEB
WEBSERVICE
WORKREQUEST

FORCE
Optional Parameter

indicates (optionally) whether or not security checking is forced regardless of the setting of RESSEC in the Security Domain's transaction token.

Values for the parameter are:
NO
YES

LOGMESSAGE
Optional Parameter

indicates whether access failures are logged to the CSCS transient data queue and the MVS System Management Facility (SMF).

Values for the parameter are:
NO
YES

Output Parameters

REASON
The values for the parameter are:
NOTAUTH

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

FAILING_USERID
Optional Parameter

is the userid that failed to access the resource.

FAILING_USERID_LENGTH
Optional Parameter

is the length of the userid (specified by the FAILING_USERID value).

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSRC gate, CHECK_CICS_RESOURCE function

The CHECK_CICS_RESOURCE function of the XSRC gate performs CICS resource access checks.

Input Parameters

ACCESS
is the type of access to be made on the resource.

Values for the parameter are:

```
COLLECT
CREATE
DEFINE
DELETE
DISCARD
EXECUTE
INQUIRE
INSTALL
PERFORM
READ
SET
UPDATE
```

RESOURCE
is the name of the resource, padded with blanks to eight-characters.

RESOURCE_TYPE
is the type of the resource.

Values for the parameter are:

```
DB2ENTRY
FILE
JOURNALNAME
PROGRAM
PSB
TDQUEUE
TRANSACTION
TRANSATTACH
TSQUEUE
```

FORCE
Optional Parameter

indicates (optionally) whether or not security checking is forced regardless of the setting of RESSEC in the Security Domain's transaction token.

Values for the parameter are:

```
NO
YES
```

LOGMESSAGE
Optional Parameter

indicates whether access failures are logged to the CSCS transient data queue and the MVS System Management Facility (SMF).

Values for the parameter are:

```
NO
YES
```

Output Parameters

REASON

The values for the parameter are:

NOTAUTH

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE

Optional Parameter

is the optional 32-bit ESM response code to the call.

FAILING_USERID

Optional Parameter

is the userid that failed to access the resource.

FAILING_USERID_LENGTH

Optional Parameter

is the length of the userid (specified by the FAILING_USERID value).

SAF_RESPONSE

Optional Parameter

is the optional 32-bit SAF response code to the call.

XSRC gate, CHECK_NON_CICS_RESOURCE function

The CHECK_NON_CICS_RESOURCE function of the XSRC gate performs non-CICS resource access checks.

Input Parameters

ACCESS

is the type of access to be made on the resource.

Values for the parameter are:

ALTER
CONTROL
READ
UPDATE

CLASSNAME

is the ESM class name in which the resource is defined.

RESOURCE_NAME

is the address and length of the resource name, in the form RESOURCE_NAME(addr,length).

LOGMESSAGE

Optional Parameter

indicates whether access failures are logged to the CSCS transient data queue and the MVS System Management Facility (SMF).

Values for the parameter are:

NO
YES

Output Parameters

REASON

The values for the parameter are:

CLASS_NOT_FOUND
ESM_INACTIVE
ESM_NOT_PRESENT
INVALID_RESOURCE_NAME
NOTAUTH

RESOURCE_NOT_FOUND

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

FAILING_USERID
Optional Parameter

is the userid that failed to access the resource.

FAILING_USERID_LENGTH
Optional Parameter

is the length of the userid (specified by the FAILING_USERID value).

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSRC gate, CHECK_SURROGATE_USER function

The CHECK_SURROGATE_USER function of the XSRC gate performs surrogate user checking.

Input Parameters

ACCESS
is the type of access to be made on the resource.

Values for the parameter are:
CHANGE
INSTALL
START

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the length of the USERID value.

Output Parameters

REASON
The values for the parameter are:
NOTAUTH

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE
Optional Parameter

is the optional 32-bit ESM response code to the call.

FAILING_USERID
Optional Parameter

is the userid that failed to access the resource.

FAILING_USERID_LENGTH
Optional Parameter

is the length of the userid (specified by the FAILING_USERID value).

SAF_RESPONSE
Optional Parameter

is the optional 32-bit SAF response code to the call.

XSRC gate, REBUILD_RESOURCE_CLASSES function

The REBUILD_RESOURCE_CLASSES function of the XSRC gate rebuilds the resource-class profiles.

Output Parameters

REASON

The values for the parameter are:

ESM_INACTIVE
REBUILD_ALREADY_ACTIVE
REBUILD_ERROR
REBUILD_NOT_NEEDED
SECURITY_INACTIVE

RESPONSE

Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

ESM_RESPONSE

Optional Parameter

is the optional 32-bit ESM response code to the call.

SAF_RESPONSE

Optional Parameter

is the optional 32-bit SAF response code to the call.

XSRN gate, RECORD_NONCOMPLIANCE function

The RECORD_NONCOMPLIANCE function of the XSRN gate is used to report that CICS is about to use a function that might not be compliant with a specified security protocol.

Input parameters

VIOLATOR

Specifies the function that is causing the violation.

Values for this parameter are as follows:

WSSE
LIBERTY

SEVERITY

Specifies the severity of the violation.

Values for this parameter are as follows:

WARN
ERROR

COMPLIANCE

Specifies which security standard is being violated.

The only value for this parameter is as follows:

SP800131A

Output parameters

Reason

The following values are returned when **RESPONSE** is DISASTER:

ABEND
LOOP

The following value is returned when **RESPONSE** is EXCEPTION:

INVALID_SECURITY

The following values are returned when **RESPONSE** is INVALID:
INVALID_FUNCTION
INVALID_FORMAT

Response
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSXM gate, ADD_TRANSACTION_SECURITY function

The ADD_TRANSACTION_SECURITY function of the XSXM gate sets the transaction options input to be stored as extended security tokens maintained by the transaction manager.

Input Parameters

EDF_SECURITY_TOKEN
Optional Parameter

is the optional EDF security token.

PRINCIPAL_SECURITY_TOKEN
Optional Parameter

is the optional principal security token.

SESSION_SECURITY_TOKEN
Optional Parameter

is the optional session security token.

Output Parameters

REASON
The following values are returned when RESPONSE is DISASTER:
GETMAIN_FAILED

The following values are returned when RESPONSE is EXCEPTION:
NO_SECURITY_TOKEN

The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSXM gate, DEL_TRANSACTION_SECURITY function

The DEL_TRANSACTION_SECURITY function of the XSXM gate deletes the security token of the specified token type for the transaction.

Input Parameters

TOKEN_TYPE
is the type of security token for the transaction.

Values for the parameter are:
EDF
PRINCIPAL
SESSION

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:
INVALID_FORMAT
INVALID_FUNCTION

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

XSXM gate, END_TRANSACTION function

The END_TRANSACTION function of the XSXM gate deletes transaction-related data.

Output Parameters

REASON
The following values are returned when RESPONSE is INVALID:

```
INVALID_FORMAT
INVALID_FUNCTION
```

RESPONSE
Indicates whether the domain call was successful. For more information, see "**RESPONSE** parameter on domain interfaces" on page 7.

Security manager domain's generic gates

Table 65 summarizes the domain's generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 65. Security manager domain's generic gates

Gate	Trace	Functions	Format
XSDM	XS 0101 XS 0102	INITIALISE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN	DMDM

In initialization processing, the security manager domain performs internal routines, and sets the initial security options, as for "XSIS gate, SET_SECURITY_DOMAIN_PARMS function" on page 1432.

For all starts the information comes from the system initialization parameters.

Security manager domain also issues console messages during initialization to report whether or not security is active.

In quiesce and termination processing, the security manager domain performs internal routines only.

Modules

Module	Function
DFHXSAD	Handles the following requests: ADD_USER_WITH_PASSWORD ADD_USER_WITHOUT_PASSWORD DELETE_USER_SECURITY INQUIRE_USER_ATTRIBUTES VALIDATE_USERID ADD_USER_VIA_ICRX INQUIRE_ICRX RELEASE_ICRX RELEASE_ICRX_STORAGE

Module	Function
DFHXSDM	Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN
DFHXSDUF	XS domain offline dump formatting routine
DFHXSFL	Handles the following requests: FLATTEN_USER_SECURITY UNFLATTEN_USER_SECURITY UNFLATTEN_ESM_UTOKEN
DFHXSIS	Handles the following requests: INQUIRE_SECURITY_DOMAIN_PARMS INQUIRE_REGION_USERID SET_SECURITY_DOMAIN_PARMS SET_NETWORK_IDENTIFIER SET_SPECIAL_TOKENS INQUIRE_REALM_NAME
DFHXSLU	Handles the following requests: GENERATE_APPC_BIND GENERATE_APPC_RESPONSE VALIDATE_APPC_RESPONSE
DFHXSPW	Handles the following requests: INQUIRE_PASSWORD_DATA UPDATE_PASSWORD CREATE_PASSTICKET INQUIRE_CERTIFICATE_USERID REGISTER_CERTIFICATE_USER
DFHXSRC	Handles the following requests: CHECK_CICS_RESOURCE CHECK_CICS_COMMAND CHECK_NON_CICS_RESOURCE CHECK_SURROGATE_USER REBUILD_RESOURCE_CLASSES
DFHXSSA	Manages the routing of all security domain supervisor requests, and handles those requests that are concerned with adding and deleting users.
DFHXSSB	Handles all the supervisor state interfaces with the ESM that are concerned with extracting data from the ESM database.
DFHXSSC	Handles all the supervisor state interfaces with the ESM that are concerned with resource checking, including the building and deleting of in-storage profiles for the use of the resource check functions.
DFHXSSD	Handles supervisor state interfaces with RACF that are concerned with PassTicket generation.
DFHXSSI	Handles the following requests: DEACTIVATE_SECURITY INITIALIZE_SECURITY_SVC TERMINATE_SECURITY_SVC
DFHXSTRI	Interprets XS domain trace entries.

Chapter 2. CICS Components

Autoinstall for terminals, consoles and APPC connections

Autoinstall for terminals provides the ability to log on to CICS from a logical unit (LU), known to z/OS Communications Server but not previously defined to CICS, and to make a connection to a running CICS system.

A new connection is created and installed automatically if autoinstall for connections is enabled, and either of the following occurs:

- An APPC BIND request or CINIT request is received for an APPC service manager (SNASVCMG) session that does not have a matching CICS CONNECTION definition
- A BIND is received for a single session that does not have a matching CICS CONNECTION definition.

A new console is created and installed automatically if autoinstall for consoles is enabled and a CIB (Command Input Buffer sent from MVS) is received by CICS (DFHZCNA) and the console TCTTE does not already exist.

For an introduction to autoinstall, and information about how to implement it, see Autoinstall.Writing a program to control autoinstall of LUs gives information about implementing the autoinstall user program. The CICS-supplied programs are:

- DFHZATDX, which provides autoinstall for terminals only
- DFHZATDY, which provides autoinstall for terminals and APPC connections.

These programs are user-replaceable, because you may need to tailor the basic function to suit your CICS environment.

Design overview

Before a z/OS Communications Server device can communicate with CICS, a z/OS Communications Server session must be established between the device and CICS. The sequence of operations is LOGON, Open Destination (OPNDST), and Start Data Traffic (SDT). CICS can also initiate the LOGON by using a SIMLOGON.

The session can be requested by:

- Specifying AUTOCONNECT when the terminal is defined to CICS
- A z/OS Communications Server master terminal command requesting a LOGON to CICS for a given terminal; for example, V NET,LOGON=CICSA,ID=L3277C1
- An individual terminal operator issuing a LOGON request (LOGON APPLID(CICSA))
- A CICS master terminal command requesting LOGON for a given terminal (CEMT SET TERMINAL(xxxx) INSERVICE ACQUIRED)
- CICS internally requesting a LOGON; for example, to process an ATI request
- LOGAPPL=CICS in the LU statement.

Consoles are not z/OS Communications Server resource but they usse a similar mechanism to autoinstall the TCTTE.

© Copyright IBM Corp. 1997, 2017

Autoinstall of APPC device logon flow

This section describes the flow of control for an APPC parallel session device (or single session via a BIND) that is to be logged on by autoinstall.

1. When an APPC device attempts to logon, z/OS Communications Server drives the logon exit DFHZLGX if a CINIT is received, or the SCIP exit DFHZBLX if a BIND is received.

   Note that DFHZBLX is a new z/OS Communications Server exit module that is called by DFHZSCX if an LU62 BIND has been received.

   In the following circumstances, an APPC LU is a candidate for autoinstall.

   - If the connection is not already defined to CICS.
   - If the connection is not already installed.
   - If the autoinstall user program (specified by the AIEXIT system initialization parameter) exists and caters for functions 2-4 as well as functions 0-1.
   - If the z/OS Communications Server ACB is open.
   - If it is an APPC parallel session connection.
   - If it is an APPC single session connection with an incoming BIND (as opposed to CINIT - which uses terminal autoinstall).
   - If ISC=YES is specified in the SIT.
   - If the maximum number of concurrent logon requests (specified by AIQMAX) has not been exceeded.
   - If the customer has installed the correct 'template' connection that is to be 'cloned' (or copied) to create the new connection.

   DFHZLGX or DFHZBLX searches for the connection in the terminal control table (TCT) by comparing the NETNAME passed by z/OS Communications Server with the NETNAME found in the NIB descriptor for each installed session.

   If a match is found and AUTOINSTALL is enabled (TCTVADEN is set), CICS verifies that the terminal is eligible for autoinstall. Processing then consists of:

   - Building an autoinstall work element (AWE) by issuing an MVS GETMAIN for subpool 1.
   - Copying the CINIT RU (DFHZLGX) or BIND (DFHZBLX) into the AWE.
   - Adding the AWE to the end of the AWE chain, which is chained from the TCT prefix.

   If a match is found showing that this connection already exists then the logon proceeds as for a defined connection.
2. Later, the AWE is actioned by DFHZACT attaching transaction CATA. For every AWE on the AWE chain, the DFHZATA autoinstall program is dispatched, passing to DFHZATA the AWE's address.
3. The DFHZATA program:
   a. Validates the BIND image passed in the AWE. If the image is not valid, issue message DFHZC6901.
   b. Calls DFHZGAI Function(CREATE_CLONE_BPS) to create a Builder Parameter Set from which to create the new connection ('clone'). This is done by calling the customer supplied autoinstall user exit program (which can be based on DFHZATDY) in which the customer chooses which 'template' connection the new connection should be copied from.

      If at any point DFHZGAI finds a problem it issues message DFHZC6920 or DFHZC6921 or DFHZC6922 with an exception trace entry which will explain the reason for failure.

c. Issue DFHZCP function(INSTALL) to create the CONNECTION, MODEGROUP and SESSIONs, based on the attributes of the template connection.
d. For parallel sessions with an incoming BIND, chose the SNASVCMG secondary session and call DFHZGAI (SET_TCTTE_FOR_OPNDST). This mimics code in DFHZBLX to check the session against the incoming BIND.

   If at any point DFHZGAI finds a problem it issues message DFHZC6923 with an exception trace entry which explains the reason for failure.
e. For parallel session with an incoming CINIT, chose the SNASVCMG primary session.
f. If the installation was successful, commit the CONNECTION and queue it for logon processing. The new CONNECTION is queued for OPNDST processing.
g. Free the AWE.

Autoinstall of an APPC Generic Resource connection: If this system is registered as a generic resource and a bind is received from another generic resource then z/OS Communications Server exit DFHZBLX will initiate an autoinstall if there is no generic or member name connection available for use.

An AWE is created with extra parameters such as the generic resource name and member name of the partner and possibly a suggested template.

Autoinstall then continues as for normal APPC and the extra parameters are reflected into the TCSE and TCTTE via the BPS.

Autoinstall of consoles install flow

1. The modify command comes into DFHZCNA via a CIB (Command Input Buffer) from MVS when a user types a console command for CICS.
2. DFHZCNA scans the Console Control Elements for a matching console name. If no CCE is found and autoinstall for consoles is enabled, an Autoinstall Work Element is created and added to the AWE queue.
3. DFHZACT scans the AWE queue and attaches the CATA transaction.
4. The CATA transaction calls DFHZATA, which sees that the AWE is for a console (sometimes called a Console Work Element) and calls DFHZATA2.
5. DFHZATA2 proceeds as follows:
   a. It finds the console models (AICONS is supplied in group DFHTERMC).
   b. If SIT AICONS(YES) is specified, the models are passed to the autoinstall user-replaceable program, which returns the termid. The default autoinstall user-replaceable program returns the last four characters of the console name.
   c. If SIT AICONS(AUTO) is specified, DFHZGBM is called to get a name in the console bitmap, in the form ¬AAA. The autoinstall user-replaceable program is not called.
   d. It calls DFHZCP FUNCTION(INSTALL).
   e. It issues EXEC CICS SYNCPOINT.
   f. It signs on if using preset security of USERID=*EVERY|*FIRST specified in the AI model TYPETERM.
   g. It gets a TIOA to hold the data specified in the command. For example, if `/f jobname,CEMT I TE` is typed at the console, CEMT I TE is put into the TIOA.

h. It calls DFHZATT to attach the transaction specified in the MODIFY command (for example, CEMT).

Sign-on to consoles flow

If a CIB is received with the same console name but with a different USERID then the autoinstall program DFHZATA2 is called to sign off the original USERID and sign on to the new USERID as follows:

1. DFHZCNA receives the modify and
   a. Finds the CCE
   b. Finds that the USERID is different and is already signed on
   c. Creates an AWE for signoff/on
   d. Chains the AWE for DFHZACT.
2. DFHZACT attaches CATA
3. CATA calls DFHZATA which calls DFHZATA2 for signoff/on
4. DFHZATA2 issues preset security sign off for the original USERID followed by sign on for the new USERID
5. DFHZATA2 then gets a TIOA for the modify command data and calls DFHZATT to attach the transaction as for normal autoinstall for consoles.

Disconnection flow for terminals (LU-initiated)

This section describes the flow of control when a request is made to disconnect an autoinstalled terminal; for example, by entering a CESF LOGOFF command that causes an **EXEC CICS ISSUE LOGOFF** command to be issued.

1. First the following functions are performed:
   - Set on the CLSDST flag in the TCTTE.
   - Put the TCTTE on the activate chain for DFHZACT to dispatch.
2. Control is then passed to the Close destination program, DFHZCLS, which performs the following functions:
   - Set on the SHUTDOWN_IN_PROGRESS flag in the TCTTE.
   - Set on the REQUEST_SHUTDOWN flag in the TCTTE.
3. The Send asynchronous commands program, DFHZDSA is then called to send a VTAM SHUTD command to the LU (autoinstalled terminal) to be disconnected. The DFHZDSA program removes the TCTTE from the activate chain, pending completion of the SHUTD command.
4. When the VTAM SHUTD command has completed, z/OS Communications Server calls the asynchronous send exit, DFHZSAX, which performs the following functions:
   - Set off the REQUEST_SHUTDOWN flag in the TCTTE.
   - Set on the SHUTDOWN_SEND flag in the TCTTE.
   - Put the TCTTE back on the activate chain for DFHZACT to dispatch.
5. z/OS Communications Server then drives the asynchronous receive exit, DFHZASX, with the SHUTC (“shutdown complete”) command sent by the LU to be disconnected. DFHZASX performs the following functions:
   - Ensures that the NODE_QUIESCED_BY_CICS, SHUTDOWN_IN_PROGRESS, and CLSDST flags are still on.
   - Puts the TCTTE back on the activate chain for DFHZACT to dispatch.
6. Control is then passed to the Close_Destination program, DFHZCLS. The DFHZCLS program performs the following functions:
   - Set on the PENDING_DELETE flag in the TCTTE to prevent z/OS Communications Server exits scheduling requests for the device.

- Issue UNBIND (CLSDST POST=RESP) for the device.

7. The Close destination exit, DFHZCLX, is driven. If the CLSDST request is successful (that is, there is a positive response from UNBIND), the following functions are performed:
   - Set on the SESSION_CLOSED flag in the TCTTE.
   - Flag the TCTTE for deletion.
   - Enqueue the TCTTE to DFHZNAC.
8. Control is passed to the DFHZNAC program, which performs the following functions:
   - Set on the DELETE_REQUIRED flag in the TCTTE.
   - Put the TCTTE on the activate chain for DFHZACT to dispatch.
   - Issue message DFHZC3462 (session terminated).
9. On the delete request, the DFHZNCA copybook of DFHZNAC checks the value of the **AILDELAY** system initialization parameter.
   - If AILDELAY is zero, the TCTTE is queued via DFHZACT with the address of the TCTTE as input. Its function is to perform clean up operations, the principal operation being to ask DFHZCQ to delete the TCTTE.
   - If AILDELAY is not zero, DFHZNCA initiates CATD using the delay specified and passes the address of the TCTTE.

   Up to three attempts are made to delete the TCTTE. This is because the reason for the failure may be the existence of a transient condition, such as the TCTTE being on the DFHZNAC queue to output a message to CSMT. If the initial delete attempt fails, it is attempted again after one second; if this fails, another attempt is made after a further 5 seconds. If the third attempt fails, it is assumed that the failure is a hard failure, which will not disappear until the device is reconnected; in this case, message DFHZC6943 is issued, a syncpoint is taken, and the TCTTE delete status is reset to make the TCTTE reusable.

   If the deletion is successful, the delete is committed, the autoinstall control program is invoked to permit any specific clean up operations to take place, and message DFHZC6966 is issued.

   If a PWE exists for this TCTTE, the PWE is requeued onto the AWE chain.

Disconnection of an autoinstalled terminal can also be requested by CICS shutdown, terminal timeout, and terminal errors. In these cases the flow is slightly different.

Deletion of autoinstalled APPC devices.

This section describes the flow of control when an APPC sync level 1 device has its last session released. This can occur as a result of unbind flows from the partner or a RELEASE command being issued against the connection in this system.

Only synclevel 1 autoinstalled connections are deleted in this way. They will have had TCSE_IMPLICIT_DELETE set by the builders from zx_delete_x in the BPS (set by DFHZGAI).

TCSE_CATLG_NO indicates that the connection is not to be written to the catalog (SIT Parameter AIRDELAY=0).

1. After DFHZCLS, the CLSDST program, issues DFHTCPLR TIDYUP TCSEDDP and TCSE_DELETE_SCHEDULE are set and CATD is initiated with a delay of AILDELAY.

2. CATD runs DFHZATD which sets TCSE_DELETE_STARTED and calls DFHZCP FUNCTION=DELETE to delete the sessions, modegroup and connection.

If a SIMLOGON or BIND occur before the delete starts (TCSE_DELETE_SCHEDULED) then the connection delete is aborted and the connection reused.

If a SIMLOGON occurs during the actual delete (TCSE_DELETE_STARTED) then the delete is vetoed and the connection is reacquired.

If a BIND occurs during the actual delete (TCSE_DELETE_STARTED) then the delete goes ahead and the PWE that was created is turned into an AWE and the logon will create a new connection.

If TCSE_DELETE_AT_RESTART is set then DFHZATR will delete the connection if it has not been used after restart with a delay specified in the **AIRDELAY** system initialization parameter.

Disconnection flow (APPC devices): These connections are not deleted at LOGOFF time, so the disconnection flow is the same as for a defined connection.

Deletion of autoinstalled consoles

Consoles are deleted after a certain period of inactivity. The default is 60 minutes but this can be overridden in the autoinstall user-replaceable program.

1. The delete time is saved in the CCE during install in TCTCE_TIMEOUT_TIME.
2. DFHCESC runs at certain intervals
3. DFHCESC checks the CCEs for any console whose delete time has expired
4. For each expired CCE DFHCESC does the following
   a. Attaches CATD to do the delete
   b. CATD calls DFHZATD as for a terminal

Shipping a TCTTE for transaction routing

For transaction routing, a terminal can be defined by an entry in the terminal-owning region (TOR) with the SHIPPABLE=YES attribute.

In this case, the terminal definition is shipped to any application-owning region (AOR) when the terminal user invokes a transaction owned by and defined to that region. Definitions for advanced program-to-program communication (APPC) devices always have the SHIPPABLE=YES attribute set.

The entry in the TOR could have been installed using CEDA INSTALL, the GRPLIST at system initialization, or autoinstall. When an autoinstalled TCTTE in a TOR is deleted, the relevant shipped terminals are deleted using a separate timing mechanism.

The first time a transaction is invoked

For non-APPC devices (see Figure 3 on page 1459), the following processing is performed:

1. In the AOR, look for an existing skeleton TCTTE (TCTSK) whose REMOTENAME is the same as the local name in the TOR. If found, skip the following steps; otherwise:
2. Issue ZC_INQUIRE to the TOR.

3. In the TOR:
   - Send a builder parameter set (BPS) representing the TCTTE to the AOR.
   - Set on the SHIPPED flag (TCTEMROP) in the TCTTE.
   - Set on the SHIPPED flag (TCSEMROP) in the TCTSE for the AOR system.
   - Rewrite each entry to the catalog.
4. In the AOR:
   - Use the existing name from the TOR.
   - INSTALL the terminal (DFHZATS does the remote install).
   - Set on the SHIPPED flag (TCTSKSHI) in the TCTSK.
   - Set on the SHIPPED flag (TCSEMROG) in the TCTSE for the TOR system.
   - Rewrite each entry to the catalog.

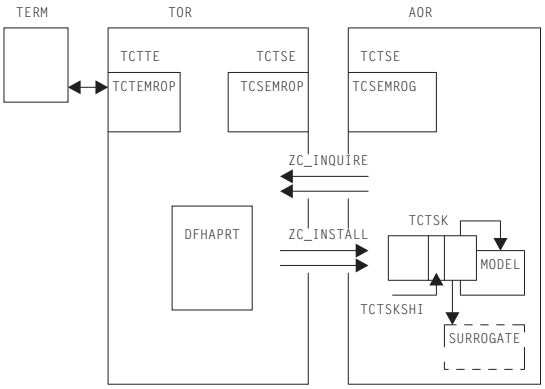


Figure 3. Transaction-routing flow for non-APPC devices

For APPC devices:

1. In the AOR, look for an existing skeleton TCTTE (TCTSK) whose REMOTENAME is the same as the local name in the TOR. If found, skip the following steps; otherwise:
2. INSTALL the terminal (DFHZATS does the remote install).
3. Set on the SHIPPED flag (TCTSKSHI) in the TCTSK.
4. Set on the SHIPPED flag (TCSEMROG) in the TCTSE for the TOR system.
5. Rewrite each entry to the catalog.

Modules

ZC (terminal control) together with the following:

Module	Function
DFHZATA	Autoinstall program
DFHZATA2	Console autoinstall program linkedits with DFHZATA
DFHZATD	Autoinstall delete program
DFHZATDX	Default autoinstall user exit
DFHZATDY	Sample autoinstall user exit
DFHZATR	Autoinstall restart program
DFHZATS	Remote autoinstall|delete program
DFHZCTRI	Trace interpretation for DFHZGAI
DFHZGAI	APPC-specific autoinstall functions

DFHZATDX

The DFHZATDX module provides user input to autoinstall processing. This module is a component of ZCP, and is the default autoinstall user program (that is, it is used if you choose not to provide your own).

For further information about the DFHZATDX sample program, see The autoinstall control program for terminals .

DFHZATDX is also called when creating and deleting shipped terminals (skeletons).

DFHZATDY

DFHZATDY is a sample autoinstall user-replaceable program, which you must modify before you can use it. Its main function is to choose a template connection which is to be used in creating the new autoinstall connection clone.

It also has to chose a name for the new connection. For further information about the DFHZATDY sample program, see the Writing a program to control autoinstall of APPC connections.

DFHZATDY is also called when creating and deleting shipped terminals (skeletons).

Diagnosing autoinstall problems

When diagnosing problems with autoinstall, consult the following list. If you have a problem with autoinstall of APPC devices, and the following list does not resolve the problem, see "Diagnosing APPC autoinstall problems" on page 1461.

- The autoinstall model table (AMT) in an SDUMP
- CEMT INQUIRE AUTINSTMODEL to show which models are installed
- TC level-1 trace, point ID AP FC8A, which shows the CINIT RU contained in the AWE on entry to DFHZATA
- CADL, CSMT, and CSNE logs:
  - Autoinstall messages (DFHZC69xx)
  - Builder messages (DFHZC59xx, DFHZC62xx, and DFHZC63xx)
  - Terminal error messages
  - Information produced by DFHZNAC
- Dump taken in the user install program (the CICS-supplied default is DFHZATDX).

Most autoinstall problems can be grouped into three categories:

1. CICS rejects the LOGON request (message DFHZC2411 on the CSNE log).
2. The device rejects the actual BIND parameters (message DFHZC2403 on the CSNE log).
3. DFHZATA diagnoses a problem (message DFHZC69xx on the CADL log).

The first category of problem is caused by CICS being in the wrong state to accept an autoinstall, for example, CICS is shutting down or AUTOINSTALL is disabled (message DFHZC2433).

The second category of problem arises when the two BIND images match, but the BIND is rejected by the actual device (message DFHZC2403). For information

about valid BIND parameters, see Descriptions of common problems in subarea networks in z/OS Communications Server: SNA Diagnosis or IBM 3274 controller diagnosis in GDDM Diagnosis.

The BIND image is contained in the CINIT RU passed to the LOGON exit. This is shown in trace point ID AP FC8A.

The reason for the third category of problem should be shown in the contents of the associated DFHZC69xx message on the CADL log. For example, message DFHZC6987 shows a BIND image mismatch between the incoming CINIT and the best available model (unlikely).

The length of each BIND image is found in the halfword preceding the image. A comparison is made for the ***smaller*** of the two length values, but not exceeding X'19' (decimal 25) bytes. The comparison is accomplished by an XC (exclusive OR) of the two BIND images into a work area. The result is ANDed with a mask that defines the required settings.

Additional bits are reset if the LU type, found in byte 14 of the BIND image, is 1, 2, 3, or 4. The final result in the work area must be 256 bytes of X'00'; any other value causes DFHZATA to reject the LOGON and write message DFHZC6987 to the CADL log.

For autoinstall to function correctly, three items must match:

1. The CINIT BIND image taken from the LOGMODE entry specified for the LU in the VTAMLST
2. The CICS MODEL BIND image built according to the specifications in the TYPETERM and TERMINAL entries
3. Device characteristics.

Diagnosing APPC autoinstall problems

When diagnosing APPC autoinstall problems, first refer to "Diagnosing autoinstall problems" on page 1460. Most of points in that section apply to APPC autoinstall problems except for points that refer to autoinstall models.

Any APPC autoinstall problem should be accompanied by message DFHZC6920 to 23. These messages each have exception trace entries which should trace enough information to allow you to diagnose the problem.

There are three autoinstall instances of DFHZC2411:

- 4 System termination - CSASTIM tested.
- 5 z/OS Communications Server termination - TCTVVTQS tested.
- 6 ISC=NO specified in the SIT.

There are two additional instances of DFHZC2433:

- 3 Autoinstall disabled - TCTVADEN tested in DFHZBLX.
- 4 Autoinstall temporarily disabled - TCTVADIN tested in DFHZBLX.

There are two additional instances of DFHZC3482:

- 3 No MVS storage for DFHZBLX to obtain MVS AWE storage.
- 4 No MVS storage for reporting a failure in a dummy work element.

Diagnosing console autoinstall problems

Much of the autoinstall for terminal advice is relevant. However, the following points should also be helpful.

1. Information about autoinstalled consoles is contained in:
   - The AWE (CWE)
   - The TCT prefix in the console BITMAP
   - The CCE
   - The SNEX
   - The interface to the autoinstall user-replaceable program.
2. When DFHZCNA is called with a modify command trace point AP FCF0 is issued and traces the CIB and CIB extension.
3. Trace point AP FCA7 shows the AWE/CWE created by DFHZCNA and passed to DFHZATA2.
4. DISCARD (used via CEMT or EXEC CICS) is useful whilst testing autoinstall for consoles.
5. CEMT INQUIRE TERMINAL is useful for seeing what consoles are installed and what their console names are.
6. The console names can vary depending on how the modify command was issued:
   - /f jobname,CEMT I TE from a TSO SDSF panel gives a console name of the USERID or the console name if changed using option 8 of SDSF.
   - f jobname,CEMT I TE from a TSO console gives a console name of the TSO USERID.
   - M/F jobname, CEMT I TE from the TSO SDSF panel gives a console name of MASTnn where nn is the names of the system. If SEC=YES is specified in the SIT then the user must first sign on with m/f jobname,CESN.
   - // MODIFY jobname,CEMT I TE from a job stream gives a console names of INTERNAL. If SEC=YES is specified in the SIT then the user must first sign on with m/f jobname,CESN.
7. The console name BITMAP is dumped in the TCP section of system dumps.
8. The extended control blocks are dumped if present when a system dump is taken.

z/OS Communications Server exits

A z/OS Communications Server exit is a special-purpose user-written routine that is scheduled by the Communications Server when the requested operation is complete. The Communications Server creates a trace record when the exit is given control.

RE entries represent RPL exits except SEND, RECEIVE, OPNDST, and CLSDST. UE entries represent non-RPL and asynchronous exits SCIP, LOGON, and LOSTERM.

For more information, see *z/OS Communications Server: SNA Programming*.

Trace

The following point IDs are provided for the autoinstall programs (DFHZATA, DFHZATD, DFHZATR, and DFHZATS), as part of terminal control:

- AP FC80 through AP FC8C, for which the trace levels are TC 1 and TC 2.

The following point IDs are provided for APPC autoinstall:

- AP FA00 to FA21, for which the trace levels are TC1 and TC2.

The following point IDs are provided for console autoinstall:

- AP FCF0
- AP FCA3 to FCA7

RE and UE trace points are recorded when the z/OS Communications Server trace API option is requested by:

```
F NET,TRACE,TYPE=VTAM,OPTION=API,MODE=EXT
```

GTF must have been started with the USR option.

Each z/OS Communications Server exit routine in CICS sets an ID byte in the TCTTE exit trace field (TCTEEIDA).

Note: VTAM is now z/OS Communications Server.

Autoinstall terminal model manager

The autoinstall terminal model manager (an OCO component of the AP domain) is responsible for managing all operations involving the autoinstall terminal model table. Autoinstall terminal models are used during the autoinstall logon process. They are installed either at system initialization or using CEDA INSTALL (see "Resource definition online (RDO)" on page 1761), and can be discarded using either the CEMT transaction or EXEC CICS commands.

The acronym AITM is often used for "autoinstall terminal model" in the contexts of both the manager and the associated table; it is also the name of one of the subroutine call formats.

The AITM manager is implemented as a set of subroutine interfaces.

Functions provided by the autoinstall terminal model manager

Table 66 summarizes the external subroutine interfaces provided by the autoinstall terminal model manager. It shows the subroutine call formats, the level-1 trace point IDs of the modules providing the functions for these formats, and the functions provided.

Table 66. Autoinstall terminal model manager's subroutine interfaces

Format	Trace	Function
AIIN		
	AP 0F10	START_INIT
	AP 0F11	COMPLETE_INIT
AIIQ		
	AP 0F18	LOCATE_TERM_MODEL
	AP 0F19	UNLOCK_TERM_MODEL
		INQUIRE_TERM_MODEL
		START_BROWSE
		GET_NEXT
		END_BROWSE
AITM		
	AP 0F08	ADD_REPL_TERM_MODEL
	AP 0F09	DELETE_TERM_MODEL

AIIN format, START_INIT function

The START_INIT function of the AIIN format is used to attach a CICS task to perform initialization of the AITM manager.

Input parameters: None.

Output parameters:
RESPONSE
is the subroutine's response to the call. It can have any of these values:
`OK|DISASTER|KERNERROR`

AIIN format, COMPLETE_INIT function

The COMPLETE_INIT function of the AIIN format is used to wait for the initialization task attached by the START_INIT function to complete processing.

Input parameters: None.

Output parameters:
RESPONSE
is the subroutine's response to the call. It can have any of these values:
`OK|DISASTER|KERNERROR`

AIIQ format, LOCATE_TERM_MODEL function

The LOCATE_TERM_MODEL function of the AIIQ format is used to obtain the attributes of a named autoinstall terminal model, and obtain a read lock on that entry in the AITM table in virtual storage.

Input parameters:
TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be located.
BPS identifies a buffer into which the attributes of the autoinstall terminal model are to be placed.

Output parameters:
RESPONSE
is the subroutine's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|KERNERROR`
[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	TM_LOCATE_FAILED
EXCEPTION	TERM_MODEL_NOT_FOUND

AIIQ format, UNLOCK_TERM_MODEL function

The UNLOCK_TERM_MODEL function of the AIIQ format is used to release a read lock on a previously located entry from the AITM table in virtual storage.

Input parameters:
TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be unlocked.

Output parameters:
RESPONSE
is the subroutine's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|KERNERROR`
[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	TM_UNLOCK_FAILED
EXCEPTION	TERM_MODEL_NOT_FOUND

AIIQ format, INQUIRE_TERM_MODEL function

The INQUIRE_TERM_MODEL function of the AIIQ format is used to obtain the attributes of a named autoinstall terminal model. (No read lock is retained.)

Input parameters:
TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be located.
BPS identifies a buffer into which the attributes of the autoinstall terminal model are to be placed.

Output parameters:
RESPONSE
is the subroutine's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|KERNERROR`
[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	TM_LOCATE_FAILED TM_UNLOCK_FAILED
EXCEPTION	TERM_MODEL_NOT_FOUND

AIIQ format, START_BROWSE function

The START_BROWSE function of the AIIQ format is used to initiate a browse of the AITM table. The browse starts at the beginning of the table.

Input parameters: None.

Output parameters:
BROWSE_TOKEN
is a token used to refer to this browse session on subsequent browse requests.
RESPONSE
is the subroutine's response to the call. It can have any of these values:
`OK|DISASTER|KERNERROR|PURGED`
[REASON]
is returned when RESPONSE is DISASTER. It has this value:
`START_BROWSE_FAILED`

AIIQ format, GET_NEXT function

The GET_NEXT function of the AIIQ format is used to obtain the name and attributes of the next autoinstall terminal model in the AITM table for the specified browse session.

Input parameters:

BROWSE_TOKEN
is the token identifying this browse session.

BPS identifies a buffer to receive the attributes of the next entry in the AITM table.

Output parameters:

TERM_MODEL_NAME
is the name of the next entry in the AITM table.

RESPONSE
is the subroutine's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|KERNERROR
```

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	TM_GET_NEXT_FAILED TM_UNLOCK_FAILED
EXCEPTION	END_OF_MODELS

AIIQ format, END_BROWSE function

The END_BROWSE function of the AIIQ format is used to terminate a browse of the AITM table.

Input parameters:

BROWSE_TOKEN
is the token identifying this browse session.

Output parameters:

RESPONSE
is the subroutine's response to the call. It can have either of these values:

```
OK|KERNERROR
```

AITM format, ADD_REPL_TERM_MODEL function

The ADD_REPL_TERM_MODEL function of the AITM format is used to add or update an entry in the AITM table in virtual storage, and record the entry on the CICS catalog.

Input parameters:

TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be added or updated.

BPS specifies the attributes of the named autoinstall terminal model.

SYSTEM_STATUS
specifies the status of the CICS system at the time of the call. It can have any one of these values:

```
COLD_START|WARM_START|ONLINE
```

where ONLINE means during execution.

Output parameters:
RESPONSE
is the subroutine's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|KERNERROR`
[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	NOT_INITIALISED ADD_REPL_FAILED
EXCEPTION	TERM_MODEL_IN_USE

AITM format, DELETE_TERM_MODEL function

The DELETE_TERM_MODEL function of the AITM format is used to remove an entry from the AITM table in virtual storage and the CICS catalog.

Input parameters:
TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be added or updated.
SYSTEM_STATUS
specifies the status of the CICS system at the time of the call. It can have any one of these values:
`COLD_START|WARM_START|ONLINE`

where ONLINE means during execution.

Output parameters:
RESPONSE
is the subroutine's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|KERNERROR`
[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
DISASTER	NOT_INITIALISED DELETE_FAILED
EXCEPTION	TERM_MODEL_IN_USE TERM_MODEL_NOT_FOUND

Modules

Module	Function
DFHAIDUF	Formats the AITM manager control blocks in a CICS system dump
DFHAIIN1	Handles the following requests: • START_INIT • COMPLETE_INIT
DFHAIIN2	Runs as a CICS task to perform initialization of the AITM manager

Module	Function
DFHAIIQ	Handles the following requests: • LOCATE_TERM_MODEL • UNLOCK_TERM_MODEL • INQUIRE_TERM_MODEL • START_BROWSE • GET_NEXT • END_BROWSE
DFHAIRP	Initializes the AITM table at CICS startup
DFHAITM	Handles the following requests: • ADD_REPL_TERM_MODEL • DELETE_TERM_MODEL
DFHAPTRN	Interprets AITM manager trace entries

Exits

No global user exit points are provided for this component.

Trace

The following point IDs are provided for the AITM manager:

- AP 0F00 through AP 0F1F, for which the trace levels are AP 1 and Exc.

Basic mapping support

Basic mapping support (BMS) allows the CICS application programmer to have access to input and output data streams without including device-dependent code in the CICS application program.

BMS provides the following services:

Message routing
: This allows application programs to send output messages to one or more terminals not in direct control of the transaction.

Terminal paging
: This allows the user to prepare a multipage output message without regard to the physical size of the output terminal; the output can then be retrieved by page number in any order.

Device independence
: This allows the user to prepare output without regard to the control characters required for a terminal; CICS automatically inserts the control characters and eliminates trailing blanks from each line.

Most of the BMS programs are resident in the CICS nucleus.

Design overview

BMS is an interface between CICS and its application programs. BMS formats input and output display data in response to BMS commands in programs. To do this, it uses device information from CICS system tables, and formatting information from *maps* that you have prepared for the program.

BMS enables an application program to read in device-dependent data and convert it to a device-independent standard form, or to generate device-dependent output

data from this device-independent standard form. In both cases, the structure of the device-independent standard form, and the layout of the data on the display terminal, are determined by a user-defined map. Related maps—for example, maps used in the same application program—are grouped together into a *map set*. See Basic mapping support for further information about the definition and use of maps and map sets.

On some terminals (such as the IBM 8775 display terminal and the IBM 3290 information panel), the available display area may be divided into a set of related "logical" screens called *partitions*. The layout and properties of the set of partitions that can be simultaneously displayed on a terminal are defined by the BMS user in a *partition set*. See Basic mapping support for further details about the definition and use of partition sets.

Maps, map sets, and partition sets are assembled offline using CICS macros. The user defines and names fields and groups of fields that can be written to and read from the devices supported by BMS. The assembled maps contain all the device-dependent control characters necessary for the proper manipulation of the data stream.

Associated with each map is a table of field names which is copied into each application program that uses the map. Data is passed to and from the application program under these field names. The application program is written to manipulate the data under the various field names so that alteration of a map format does not necessarily lead to changes in program logic. New fields can be added to a map format without making it necessary to reprogram existing applications.

Output data can be supplied from the application program by placing the data in the table under the appropriate field name. As an alternative, output maps can contain field default data that is sent when data is not supplied by an application program. This facility permits the specification of titles, headers, and so on, for output maps.

Optionally, the display of all the default data can be suppressed by the application program for any output map. Each time a map is used, the application program can temporarily modify the attributes of any named field in the output map. The extended attributes can also be modified if maps are defined with the DSATTS operand.

Output map fields with no field names can contain default data, but the application program cannot replace the default data or modify the attributes of unnamed fields.

For input, the user assembles a map defining the fields that can be written to and received from a particular device. Any data received for a particular field is moved across using the field name in the symbolic storage definition for the map. Light-pen-detectable fields defined in an input map are flagged as detected if present in an IBM 3270 Information Display System input stream. An input map for a particular case can specify a subset of the fields potentially receivable; any fields received and not represented in that map are discarded. This permits the number of fields from a map that can be typed or selected to be changed, without making it necessary to reprogram applications that currently receive data from the map.

Maps are stored in the CICS program load library. When a map is required by BMS, a copy is automatically retrieved by CICS from the program load library without application program action. Multiple users of a map contained in the program load library share a single copy in main storage.

BMS permits any valid combination of field attributes to be specified by the user when generating maps. Inclusion of BMS in CICS is a system generation option and does not prevent the application program from accessing a particular device in native mode (without using BMS). Intermixing BMS and native mode support for a terminal from the same application program may yield unpredictable results. When using mixed mode support, it is the user's responsibility to ensure the correct construction and interpretation of native mode data streams.

BMS permits the application program to pass a native mode data stream that has already been read in, and (if, for a terminal of the IBM 3270 Information Display System, the screen has been formatted) to interpret this data stream according to a given input map. This facility allows data entered with the initial reading of a transaction to be successfully mapped using BMS.

BMS provides the following services:

- Message routing
- Terminal paging
- Device independence.

Message routing

Message routing permits the application program to send an output message to one or more terminals not in direct control of the transaction. The message is automatically sent to a terminal if the terminal status allows reception of the message. If a terminal is not immediately eligible to receive the message, the message is preserved for that terminal until a change in terminal status allows it to be sent. The message routing function is used by the CICS message-switching transaction.

A BMS map that specifies extended attributes can be used for terminals that do not support extended attributes. When sending data to a variety of terminals, some of the terminals may support extended attributes and others may not. When a BMS ROUTE request is processed, BMS looks at the TCTTEs for all specified terminals and constructs a set of all the supported attributes.

A data stream is produced by BMS using this set of attributes, and the data stream and set of attributes for each page are written to a temporary-storage record. When the page is later read from temporary storage, the data stream for each terminal is modified, if necessary, to delete attributes not supported by that terminal.

Terminal paging

Terminal paging allows the user to prepare more output than can be conveniently or physically displayed at the receiving terminal. The output can then be retrieved by pages in any order; that is, in the order in which they were prepared or by skipping forward or backward in the output pages.

Terminal paging also provides the ability to combine several small areas into one area, which is then sent to the terminal. This enables the user to prepare output without regard for the record size imposed by the output terminal.

CICS provides the terminal operator with a generalized page retrieval facility that can be used to retrieve and dispose of pages.

Device independence

Device independence allows the user to prepare output without regard for the control characters required for message heading, line separation, and so on. Input to device independence consists of a data string with optional new-line characters.

Device independence divides the data string into lines no longer than those defined for the particular terminal. If new-line characters appear occasionally in the data string to further define line lengths, they are not ignored. CICS inserts the appropriate leading characters, carriage returns, and idle characters, and eliminates trailing blanks from each line. If the device does not support extended attributes, the extended attributes are ignored.

CICS allows the user to set horizontal and vertical tabs on those devices that support the facility (for example, the IBM 3767 Communication Terminal, and the IBM 3770 Data Communication System). For such devices, CICS supports data compression inbound and data compression outbound, based on the tab characteristics in the data stream under the control of the appropriate maps.

Control blocks

BMS makes use of the following control blocks (see Figure 4 on page 1473):

DSECT	Function
DFHMAPDS	Defines a physical map. It contains overlays for map set data, map data, and field data. The physical map set is stored in the CICS program library and requires a resource definition when loaded into main storage by BMS.
DFHMCAD	Defines a mapping control area (MCA). MCAs are used in DFHM32 and DFHML1 to merge (both) and sort (DFHML1 only) fields in different maps in the chain of map copies. The MCA contains field position, flags, and pointers to map and application data structure relating to this field.
DFHMCBDS	Defines the message control block (MCB). MCBs are built and referenced by DFHTPR. There is one MCB per level of page chaining. The MCBs are chained together, with the head of the chain anchored off the TCTTE BMS extension. The MCB contains a copy of the MCR, with additional working data.
DFHMCRDS	Defines the message control record (MCR). MCRs are held in CICS temporary storage. There is one MCR per BMS message in temporary storage. The MCR contains data such as the number of pages in this message, the list of target terminals for this message, data on which pages are for which LDCs or partitions, and so on. The MCR is written to temporary storage by DFHMCP. It is read and purged by DFHTPR, DFHTPS, and DFHTPQ.

DSECT	Function
DFHOSPWA	Defines the output services processor work area (OSPWA). This is the main BMS control block. For standard and full-function BMS, there is an OSPWA that is chained off the TCA and is built by DFHMCP on the first BMS command in a transaction. It contains a copy of the BMS TCA request bytes, together with the BMS status and working area. DFHTPR has its own private OSPWA. This overlays the TWA for DFHTPR unless SEND PAGE RETAIN is used. If SEND PAGE RETAIN is used, DFHTPR obtains an additional OSPWA, and chains the base OSPWA off the new OSPWA. This avoids DFHTPR damaging the base OSPWA. The OSPWA is deleted during task termination. A shorter version of the OSPWA is used by DFHMCPE (part of both the minimum-function BMS mapping control program DFHMCPE$ and also the BMS fast-path module DFHMCX). It is built in DFHMCPE⌂€™s LIFO storage, and includes space for the request information from the TCA. The DFHMCPE OSPWA is defined within DFHMCPE.
DFHPGADS	Defines a page control area (PGA). DFHTPP builds a PGA at the end of the device data stream in the terminal input/output area (TIOA) (addressed as ADDR(TIOADBA) + TIOATDL) for the SET and PAGING disposition. The PGA contains the 3270 write control character (WCC), flags about the type of TC write required, and the extended features used in this page of data stream.
DFHPSDDS	Defines a physical partition set. The partition set is stored in the CICS program library and requires a resource definition when loaded into main storage by BMS.
DFHTTPDS	Defines the terminal type parameter (TTP). This contains information for a terminal type. Note that BMS builds pages on a TTP basis. For standard and full-function BMS, DFHRLR builds TTPs as follows: 1. A "direct TTP" is built for the transaction terminal. If this supports partitions or LDCs, a further direct TTP is built for each referenced LDC or partition. This contains data for that LDC or partition. These direct TTPs are chained together, and the head of the chain is contained in the OSPWA. Direct TTPs are deleted by DFHMCP on a SEND PAGE, PURGE MESSAGE, or SEND PARTNSET command. 2. If routing is in effect, there is a chain of routed TTPs, with one TTP per terminal type in the route list. Routed TTPs are deleted by DFHMCP on a SEND PAGE or PURGE MESSAGE command. Most of BMS uses the TTP rather than the TCTTE to determine terminal-related information.
TCTTETTE	The TCTTETTE DSECT in the DFHTCTZE macro defines the TCTTE BMS extension. It is chained off the TCTTE (TCTTETEA field).
DFHTPE	Defines the BMS partition extension. This is chained off the TCTTE BMS extension if the terminal supports partitions.

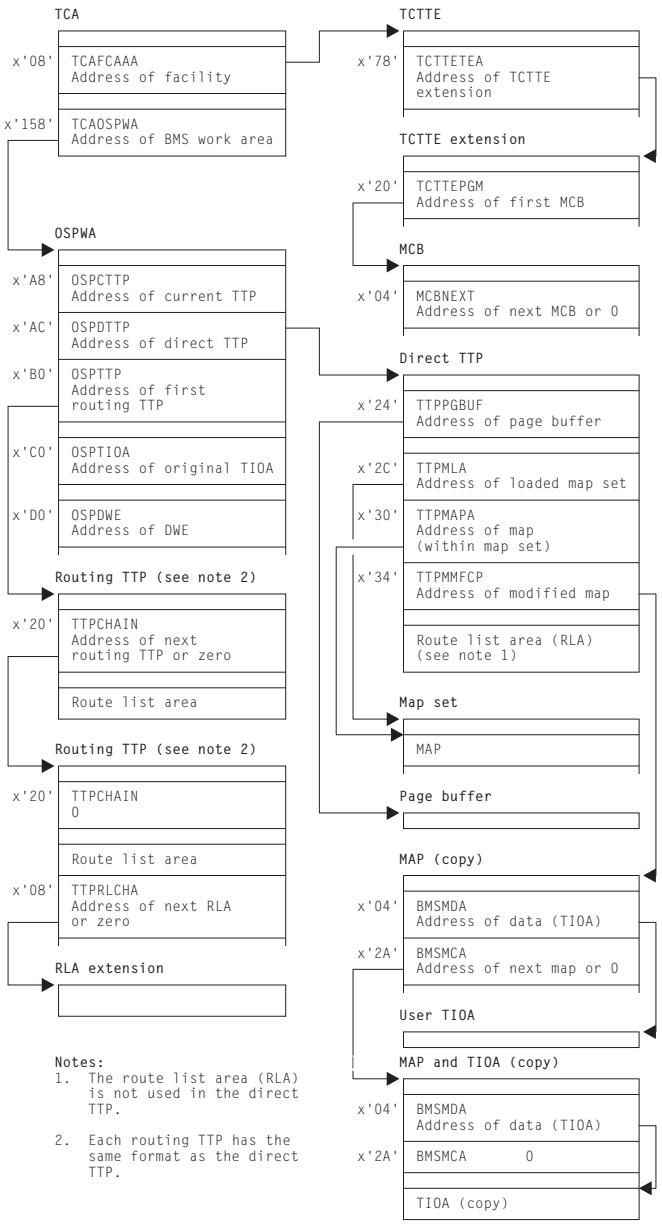


Figure 4. Control blocks associated with basic mapping support (BMS)

Modules

BMS makes use of the following modules (see Figure 5 on page 1476):

Module	Function
DFHBMSX	URM called when a 3270 data stream validation error has been detected when issuing BMS RECEIVE MAP commands.
DFHDSB	Addresses the page buffer, which was composed by the page and text build program (DFHPBP).
DFHEMS	The EXEC interface processor for BMS commands.

Module	Function
DFHIIP	Called in response to requests for BMS services involving terminals other than IBM 3270 Information Display Systems.
DFHMCP	The interface between application programs and the modules that perform mapping, message switching, page and text building, device-dependent output preparation, and message disposition to terminals, temporary-storage areas, or the application program.
DFHMCX	The BMS fast path module for standard and full-function BMS, and the program for minimum BMS support. It is called by DFHMCP if the request satisfies one of the following conditions: • It is a non-cumulative direct terminal send map or receive map issued by a command-level program. • It is for a 3270 display or an LU3 printer which does not support outboard formatting. If the terminal supports partitions, it is in the base state. • The CSPQ transaction has been started. • The message disposition has not changed.
DFHM32	Called in response to requests for BMS services involving terminals of the 3270 Information Display System.
DFHPBP	Processes all BMS output requests (SEND MAP, SEND PAGE, and SEND TEXT). It performs the following functions: • Positions the data in the page, either by placing it in a buffer, or by copying it and adjusting the map for an IBM 3270 Information Display System (SEND MAP ACCUM) • Places the data into the page buffer (SEND TEXT ACCUM) • Inserts device-dependent control characters for other than 3270 Information Display System devices, removing extended attributes.
DFHPHP	Processes terminal operations that involve partitions.
DFHRLR	Builds terminal type parameters (TTPs), which are the main blocks for building and writing out data in BMS.
DFHTPP	Directs completed pages to a destination specified in the BMS output request: SEND TEXT sends to the originating terminal; SEND MAP PAGING or SEND TEXT PAGING directs to temporary storage; and SEND MAP SET or SEND TEXT SET directs to a list of completed pages that are returned to the application program).
DFHTPQ	Checks the chain of automatic initiate descriptors (AIDs) to detect and delete AIDs that have been on the chain for an interval exceeding the purge delay time interval specified by the PRGDLAY system initialization parameter, if this has a nonzero value.
DFHTPR	Processes messages built by BMS and placed in temporary storage.
DFHTPS	Invoked for each terminal type to which a BMS logical message built with SEND MAP PAGING or SEND TEXT PAGING is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status.

Basic mapping support (BMS) is provided by means of a number of modules, each of which interfaces with other BMS modules, CICS control components, and application programs. The maps that are handled by BMS may be new maps, created to utilize BMS mapping capabilities. The interrelationships of CICS

programs requesting mapping services are summarized in Figure 5 on page 1476. Further details for specific programs within BMS are given in the topics that follow.

One of three versions (MINIMUM, STANDARD, or FULL) of basic mapping support can be selected by the system initialization parameter BMS (see BMS system initialization parameter). Where the generated versions of a BMS module differ according to the level of function provided, a suffix identifies the version as follows:

- E$ for minimum function
- A$ for standard function
- 1$ for full function.

In the module lists that follow, an asterisk (*) after a module name shows that the module is suffixed in this way. Elsewhere in this book, however, the BMS modules are usually referenced by their unsuffixed names with no distinction made between the minimum, standard, and full-function versions.

The module used by all three versions of BMS (minimum, standard, and full-function) is:

- DFHMCP* (mapping control program).

Additional modules used by both standard and full-function versions of BMS are:

- DFHDSB* (data stream build)
- DFHIIP* (non-3270 input mapping)
- DFHMCX (fast path module)
- DFHML1 (LU1 printer mapping)
- DFHM32* (3270 mapping)
- DFHPBP* (page build program)
- DFHPHP (partition handling program)
- DFHRLR* (route list resolution)
- DFHTPP* (terminal page processor).

Additional modules used only by full-function BMS are:

- DFHTPQ (terminal page clean up)
- DFHTPR (terminal page retrieval)
- DFHTPS (terminal page scheduling).

A detailed description of each of these modules follows in alphabetic order of module name.

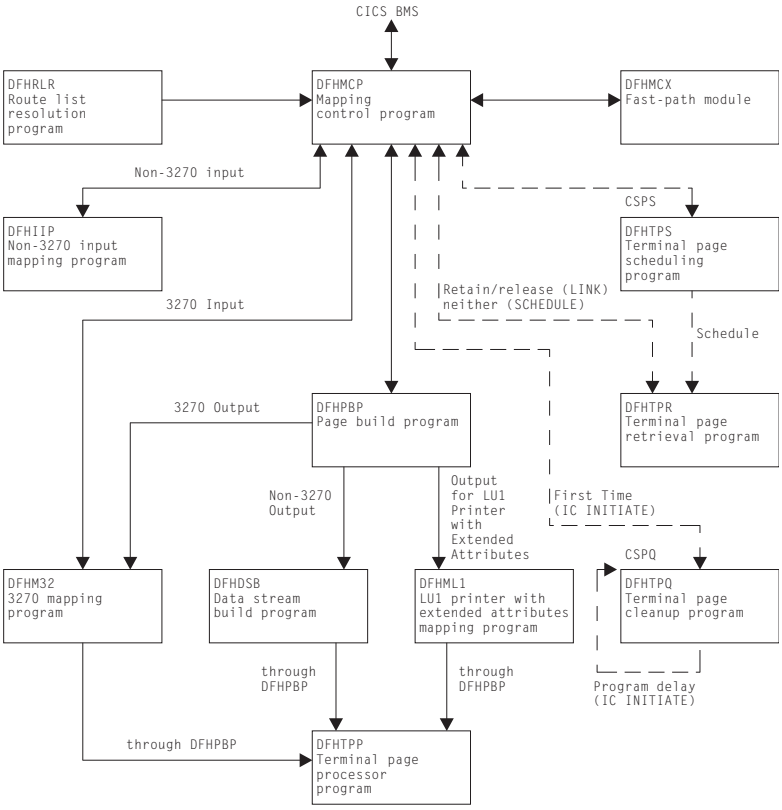


Figure 5. Modules associated with basic mapping support (BMS)

DFHBMSX (3270 data stream validation program)

The 3270 data stream validation program (DFHBMSX) is part of the BMS 3270 Intrusion Detection Service (IDS). It is called when a 3270 data stream validation error has been detected when issuing **BMS RECEIVE MAP** commands. It is a user replaceable module (URM).

Use this URM you need to configure BMS 3270 IDS to be specific about which applications or maps the service applies to. In general, this would only be necessary if an application made unusual use of the 3270 data stream and reported false hits. For most users, the BMS 3270 IDS feature will be sufficient; see BMS 3270 Intrusion Detection Service for details.

DFHBMSX has a commarea interface, which is defined in *DFHBMSXn*, where *n* is the usual language suffix, for example, D for Assembler.

This URM is called with the BMSX_FUNCTION set. BMSX_FUNCTION has the following two values:

BMSX_FUNCTION_INIT
This is set once at CICS initialization to identify whether CICS should validate 3270 data streams.

The URM must set BMSX_ACTION to one of the following functions:

BMSX_ACTION_CHECK
Validation is enabled.

BMSX_ACTION_NOCHECK
No validation will be done. This is the default.

BMSX_FUNCTION_OVERWRITE
This is set when CICS detects that a protected field is updated incorrectly by a 3270 terminal emulator during a BMS RECEIVE MAP command. This function is called only when validation is enabled during CICS initialization.

The DFHBMSX URM must set BMSX_ACTION to one of the following functions:

BMSX_ACTION_IGNORE
Do nothing.

BMSX_ACTION_LOG
Issue a DFHTF0200 message. This is the default.

BMSX_ACTION_ABEND
Issue a DFHTF0200 message and abend the ABSX transaction.

DFHBMSX can also set BMSX_LOGACTION to tell CICS to issue either a single message for the first field detected on a map or a message for all fields detected:

BMSX_LOGACTION_FIRST
Only output a message for the first field that is detected.

BMSX_LOGACTION_ALL
Issue a message for all fields that are detected.

The URM is also passed details of the first 16 fields that have been detected.

If the URM is disabled, it is equivalent to BMSX_ACTION_NOCHECK on initialization and BMSX_ACTION_IGNORE.

DFHDSB (data stream build)

The data stream build program addresses the page buffer, composed by the page and text build program (DFHPBP). The page buffer contains lines of output data that are to be written to a terminal other than an IBM 3270 Information Display System. The number of lines is contained in the TTPLINES field. The data stream build program performs the following functions on the data in the page buffer:

- Truncates trailing blanks within data lines
- Substitutes strings of physical device control characters for logical new-line characters that terminate each line of data
- Provides a format management header (FMH) for some z/OS Communications Server-supported devices
- Allows horizontal and vertical tab processing.

Figure 6 on page 1478 shows the relationships between the components of data stream build.

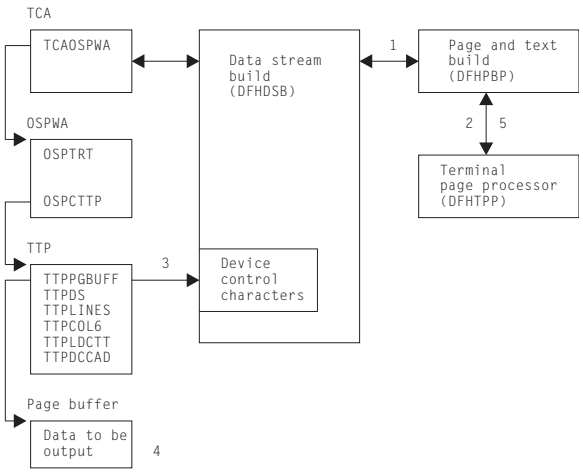


Figure 6. Data stream build interfaces

Note:

1. DFHDSB is entered from the page build program to process the page buffer.
2. For SEND TEXT commands with the NOEDIT option specified, page buffer compression is skipped and control returns to DFHPBP, which calls the terminal page processor (DFHTPP).
3. For SEND TEXT commands without the NOEDIT option, the appropriate device control characters for the target device are selected for substitution.
4. The page buffer containing the data to be compressed is located through the address stored at TTPPGBUF.
5. After compression of the page buffer data, control returns to DFHPBP, which calls DFHTPP to provide disposition of the page.

DFHIIP (non-3270 input mapping)

The non-3270 input mapping program (DFHIIP) is called in response to requests for BMS services involving terminals other than IBM 3270 Information Display Systems.

Figure 7 on page 1479 shows the relationships between the components of non-3270 input mapping.

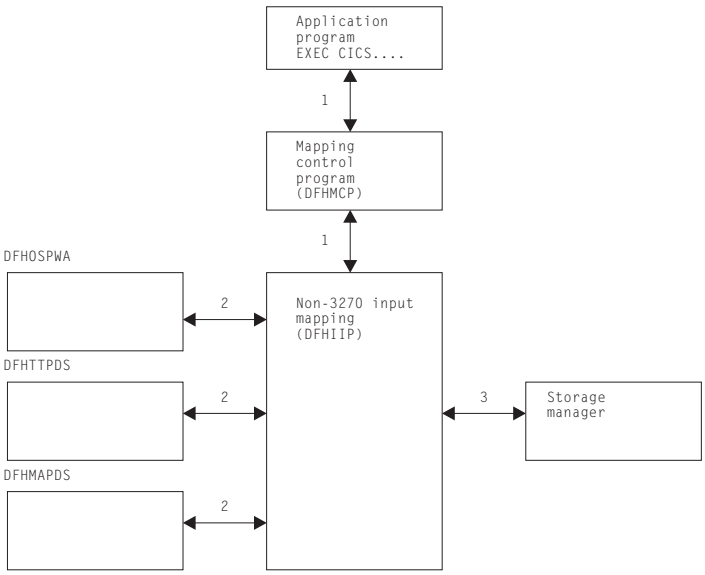


Figure 7. Non-3270 input mapping interfaces

Note:

1. A RECEIVE MAP request by an application program, communicating with other than an IBM 3270 Information Display System, passes information through the TCA through the mapping control program (DFHMCP) to DFHIIP.
2. The map required for an operation is either passed by the application program or loaded by DFHMCP.
3. DFHIIP communicates with storage control to obtain and release buffers for mapping operations.

DFHMCP (mapping control program)

The mapping control program (DFHMCP) is the interface between application programs and the modules that perform mapping, message switching, page and text building, device-dependent output preparation, and message disposition to terminals, temporary-storage areas, or the application program.

Figure 8 on page 1480 shows the relationships between the components of mapping control.

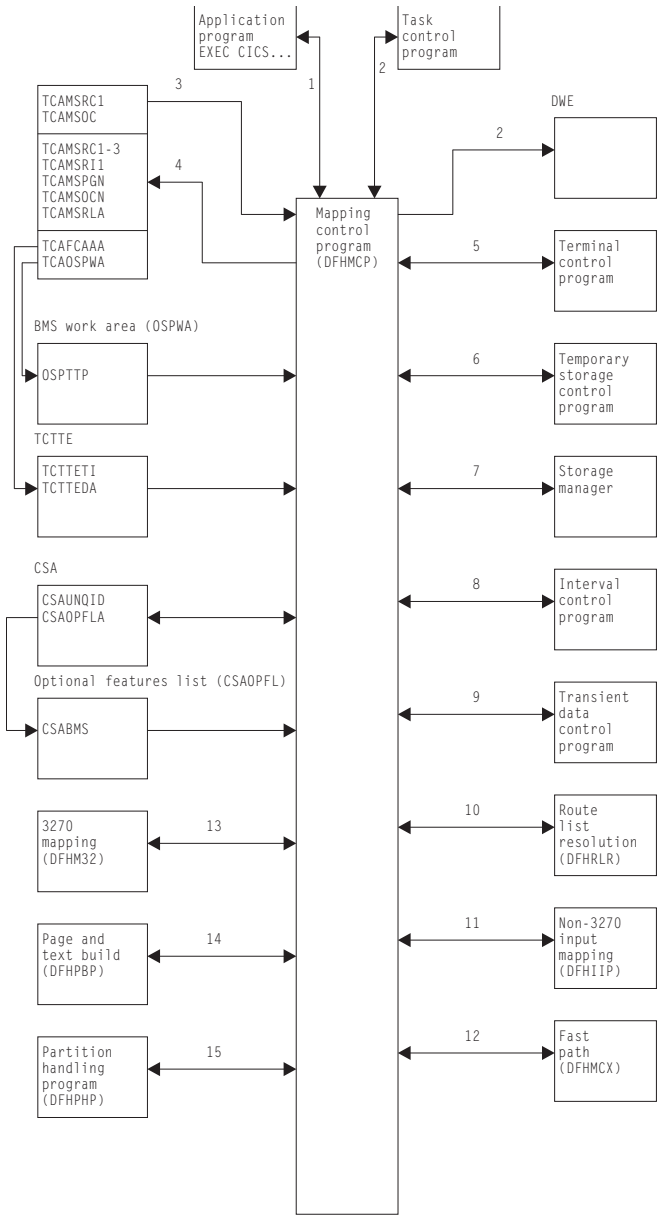


Figure 8. Mapping control program interfaces

Note:

1. This program is entered when an application program issues a request for basic mapping support services.
2. It may also be called by task control to process a deferred work element (DWE) if an application program terminates and there are partial pages in storage, or the message control record (MCR) created during execution of the task has not been placed in temporary storage.
3. The following information is returned to the requester: error codes, page overflow information, and (for a SEND MAP SET or SEND TEXT SET command) a list of completed pages.
4. DFHMCP communicates with temporary storage control to put the MCR for routed or stored messages, if a ROUTE command, or SEND MAP PAGING or

SEND TEXT PAGING command is issued. A DELETEQ TS command is issued to request that a message be purged from temporary storage if a PURGE MESSAGE command is issued.

5. DFHMCP communicates with storage control to:
   - Acquire and free storage in which the MCR is built (a SEND MAP command after a SEND MAP PAGING, SEND TEXT PAGING, or ROUTE command)
   - Acquire and free storage in which to copy the message title (a ROUTE command with the TITLE option specified)
   - Acquire storage to build automatic initiate descriptors (AIDs) for non-routed messages, or routed messages to be delivered immediately (a SEND PAGE command)
   - Acquire a BMS work area (OSPWA) at the time of the initial BMS request
   - Acquire and free an area used for user request data if a SEND PAGE command must be simulated before processing the user's request
   - Free the returned page list (a DELETEQ TS command)
   - Free map copies if SEND PAGE command was issued and pages were being built in response to SEND PAGE commands
   - Free terminal type parameters (TTPs) (SEND PAGE command).
6. DFHMCP communicates with program manager to:
   - Load and delete map sets
   - Link to the terminal page retrieval program (DFHTPR) to process one or more pages of a message if a SEND PAGE command is issued with the RETAIN or RELEASE option specified
   - Abnormally terminate tasks that incur errors that cannot be corrected.
7. DFHMCP communicates with interval control to:
   - Initiate transaction CSPQ
   - Obtain the current time of day, which is then used to time stamp AIDs for routed messages
   - Initiate transaction CSPS for messages to be delivered later.
8. DFHMCP communicates with task control to schedule transaction CSPQ for every terminal that is to receive a routed message to be delivered immediately.
9. Transient data control is used to send error and information messages to the master terminal.
10. Route list resolution (DFHRLR) is used to collect terminals from a user-supplied route list or from the entire TCT by terminal type, and build a terminal type parameter (TTP), which controls message building, for each terminal type. It is also used to build a single-element TTP for the originating terminal.
11. Non-3270 input mapping (DFHIIP) is used to process RECEIVE MAP requests for a terminal other than an IBM 3270 Information Display System.
12. The mapping control program calls DFHMCX if the request is eligible for the BMS fast-path module.
13. 3270 mapping (DFHM32) is used to process RECEIVE MAP requests for an IBM 3270 Information Display System.
14. Page and text build (DFHPBP) processes the following output requests:
15. Page and text build program (DFHPBP) processes all BMS output requests
    - SEND MAP
    - SEND MAP PAGING

- SEND MAP SET
- SEND PAGE
- SEND TEXT
- SEND TEXT PAGING
- SEND TEXT SET.

For 3270 output, DFHM32 is called; for other output, DFHML1 is called.

16. The partition handling program (DFHPHP) is called when the data is in an inbound structured field. DFHPHP extracts the partition ID, device AID, and cursor address.

DFHML1 (LU1 printer with extended attributes mapping)

The LU1 printer with extended attributes mapping program, DFHML1, is called in response to requests for BMS services involving terminals of the 3270 Information Display System. Figure 9 shows how the DFHML1 program responds to these requests.

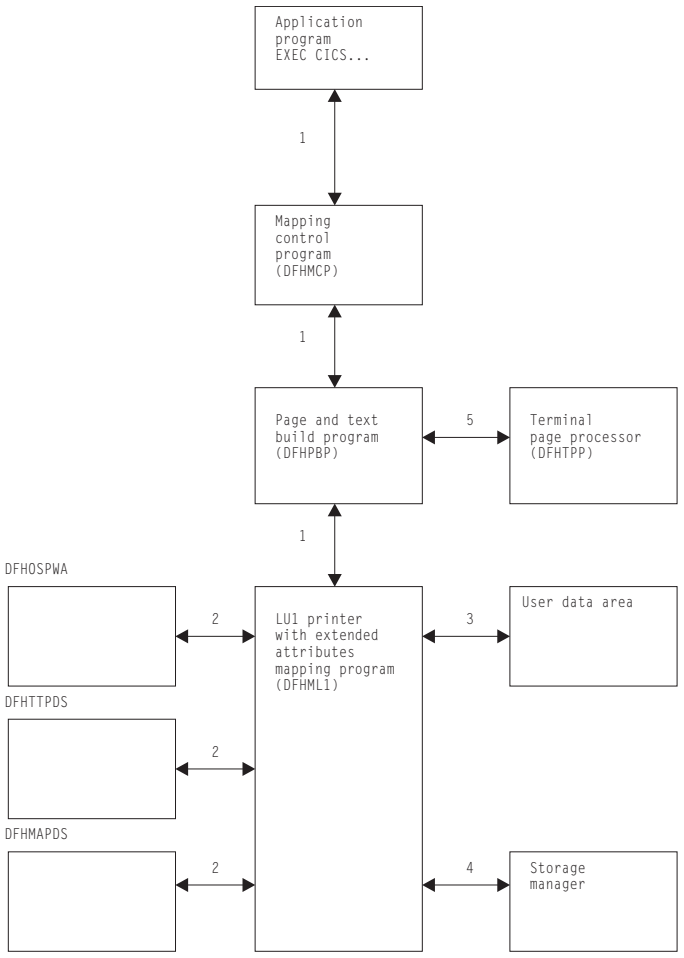


Figure 9. LU1 printer with extended attributes mapping program interfaces

Note:

1. The following types of requests, by application programs communicating with LU1 printer mapping, pass information through the mapping control program (DFHMCP), and the page and text build program (DFHPBP), to DFHML1:

- SEND MAP ACCUM
- SEND MAP SET
- SEND TEXT
- SEND TEXT ACCUM
- SEND TEXT SET

For one page of output, DFHML1 acquires an area and formats it into a chain of control blocks known as map control areas (MCAs). Each MCA corresponds to one map on the page and contains information about chaining down the maps and processing the fields in each map. DFHML1 then builds the data stream directly from the maps and the TIOAs.

2. Maps are either passed by the application program or loaded by DFHMCP.
3. The address of a terminal input/output area (TIOA) is supplied by the application program for all requests.
4. DFHML1 communicates with storage control to obtain and release storage for MCAs and for the mapped data.
5. All requests (see note 1 on page 1482) are sent to a designated destination by the terminal page processor (DFHTPP), after the return of control to DFHPBP.

DFHM32 (3270 mapping)

The 3270 mapping program (DFHM32) is called in response to requests for BMS services involving terminals of the 3270 Information Display System. Figure 10 shows how the 3270 mapping program responds to these requests.

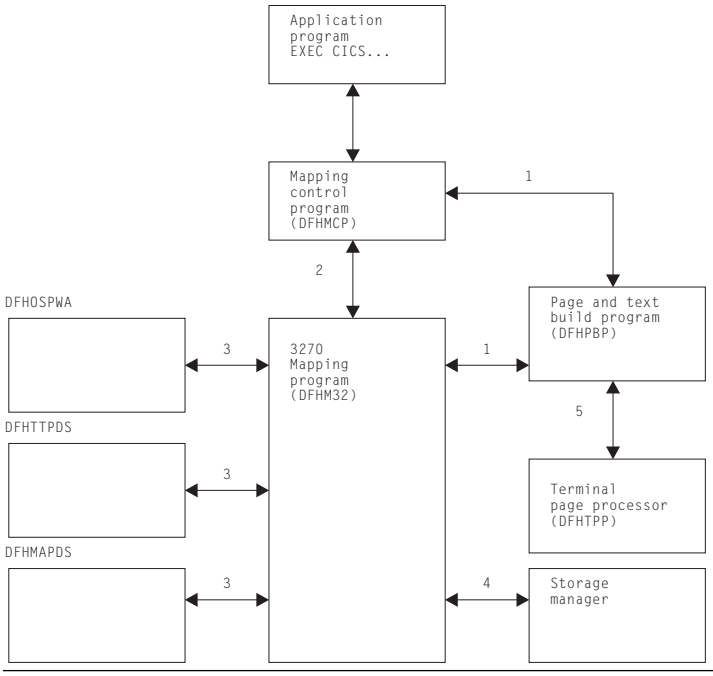


Figure 10. 3270 mapping program interfaces

Note:

1. The following types of requests by an application program communicating with an IBM 3270 Information Display System passes information through the TCA by way of the mapping control program (DFHMCP) and the page and text build program (DFHPBP) to DFHM32:
   - SEND MAP ACCUM

- SEND MAP PAGING
- SEND MAP SET
- SEND TEXT
- SEND TEXT ACCUM
- SEND TEXT PAGING
- SEND TEXT SET

For one page of output, DFHM32 acquires an area and formats it into a chain of control blocks known as map control areas (MCAs). Each MCA corresponds to one map on the page and contains information for chaining down the maps and processing the fields in each map. DFHM32 then builds the data stream directly from the maps and the TIOAs.

2. A RECEIVE MAP or RECEIVE MAP FROM request by an application program communicating with an IBM 3270 Information Display System passes information through the TCA through the message control program (DFHMCP) to DFHM32.
3. Maps are either passed by the application program or loaded by DFHMCP.
4. DFHM32 communicates with storage control to obtain and release storage for MCAs and for the mapped data.
5. All output requests (see note 1 on page 1483) are sent to a designated destination by the terminal page processor (DFHTPP) after control is returned to DFHPBP.

DFHPBP (page and text build)

The page and text build program (DFHPBP) processes all BMS output requests

- SEND MAP
- SEND MAP PAGING
- SEND MAP SET
- SEND PAGE
- SEND TEXT
- SEND TEXT PAGING
- SEND TEXT SET.

It performs the following functions:

- Positions the data in the page, either by placing it in a buffer, or by copying it and adjusting the map for an IBM 3270 Information Display System (SEND MAP ACCUM)
- Places the data into the page buffer (SEND TEXT ACCUM)
- Inserts device-dependent control characters for other than 3270 Information Display System devices, removing extended attributes.

Figure 11 on page 1485 shows the relationships between the components of page and text build.

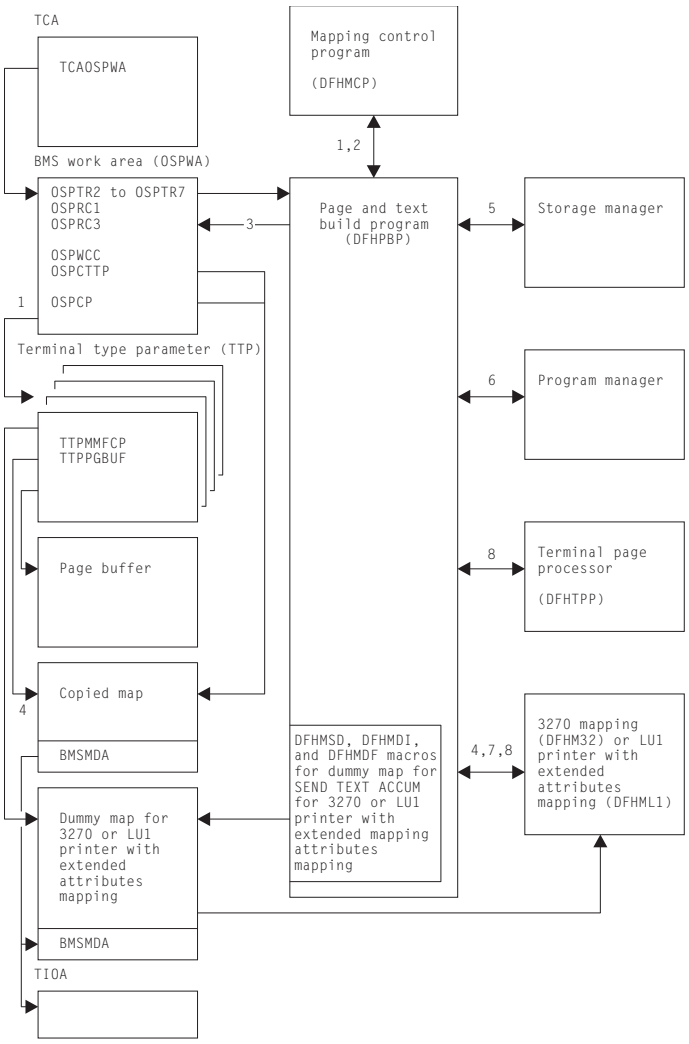


Figure 11. Page and text build program interfaces

Note:

1. DFHPBP is entered from the mapping control program, DFHMCP, to process all BMS output requests. It is called once for each terminal type parameter (TTP) on the TTP chain pointed to by OSPTTP. The current TTP in the chain is pointed to by OSPCTTP.
2. DFHPBP returns control to DFHMCP when request processing is complete, or when the page must be written out before a SEND MAP ACCUM request can be processed and an OFLOW=symbolic address operand was specified.
3. OSPTR2, OSPTR3, ..., OSPTR7 contain request data from the DFHBMS macro expansion. OSPRC1 and OSPRC3 contain return codes to be examined by DFHMCP.
4. For a SEND MAP ACCUM request for an IBM 3270 Information Display System, the map is copied and chained to the TTP. For a SEND TEXT ACCUM request for an IBM 3270 Information Display System, a dummy map is created and chained to the TTP. When a page is complete, control is given to 3270 mapping (DFHM32), which combines the map copies chained to the TTP and maps the data.

For a SEND MAP ACCUM request for an LU1 printer with extended attributes, the map is copied and chained to the TTP. For a SEND TEXT ACCUM request, a dummy map is created and chained to the TTP. When a page is complete, control is given to the LU1 printer mapping program (DFHML1), which combines the map copies chained to the TTP and maps the data.

5. DFHPBP communicates with storage control to:
   - Acquire and free buffers in which pages are built
   - Acquire storage for copies of maps for SEND MAP ACCUM or SEND TEXT ACCUM
   - Acquire storage for a copy of the user's data for SEND MAP ACCUM or SEND TEXT ACCUM.
6. DFHPBP requests program manager to terminate a transaction abnormally (ABEND) if certain errors occur that cannot be corrected.
7. A SEND TEXT ACCUM request for an IBM 3270 Information Display System causes a map set consisting of one dummy map to be passed to 3270 mapping (DFHM32). The map has one field with attributes FREEKB and FRSET.

   SEND TEXT ACCUM requests for an LU1 printer cause a map set consisting of one dummy map to be passed to the LU1 printer mapping program (DFHML1). The map has one field with attributes FREEKB and FRSET.
8. If the page is being constructed for an IBM 3270 Information Display System, control is given to DFHM32 to map the data and then to DFHTPP to output the page.

   If the page is being constructed for an LU1 printer, control is given to DFHML1 to map the data, and then to DFHTPP to output the page. Otherwise, control is given to DFHDSB to add device dependencies to the page, and then to the terminal page processor (DFHTPP) to output the page.

DFHPHP (partition handling program)

The partition handling program (DFHPHP) processes terminal operations that involve partitions. DFHPHP has one entry point, and starts with a branch table that passes control to the required routine according to the request. It consists of routines that perform the following functions:

- PHPPSI tests whether there is a partition set in storage. If there is and it is not the required partition set, that partition set is deleted. When no partition set is in storage, an attempt is made to load the appropriate partition set.
- PHPPSC builds a data stream to destroy any partitions that may already be loaded on the terminal, creates the partition set designated by the application partition set, and sets the name of the partition set in the TCTTE to be the name of the application partition set.
- PHPPIN extracts the AID, cursor address, and partition ID. The AID and cursor address are put in the TCTTE, and the partition ID is converted to a partition name and returned to the caller. A check is made that the partition ID is a member of the application partition set.
- PHPPXE sends a data stream to a terminal to activate the appropriate partition and sends an error message to any error message partition if input arrived from an unexpected partition.

Figure 12 on page 1487 shows the relationships between the components of partition handling.

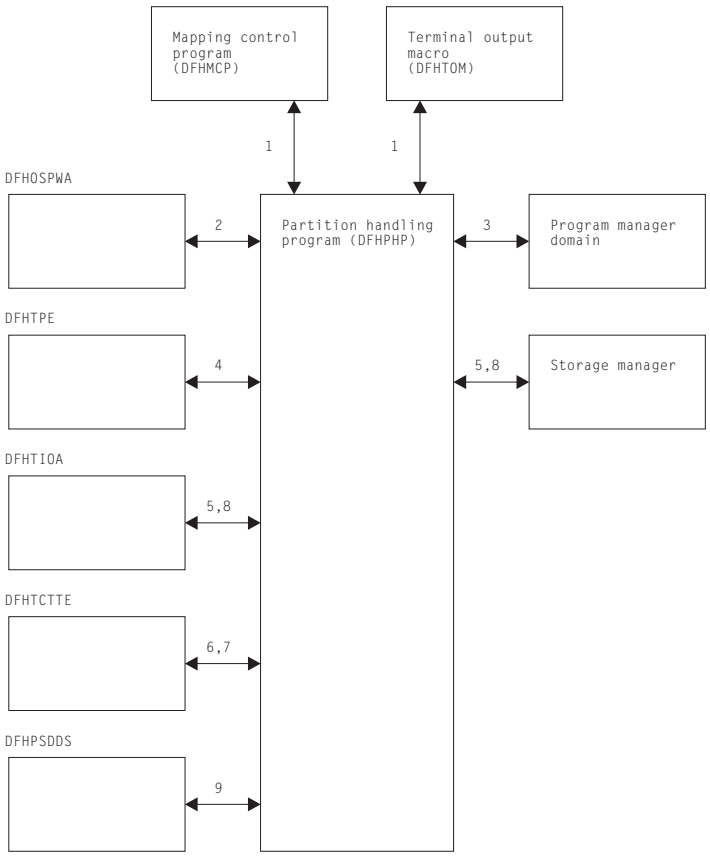


Figure 12. Partition handling program interfaces

Notes:

1. DFHPHP is called by the mapping control program (DFHMCP) and by the terminal output macro (DFHTOM).
2. PHPPSI refers to OSPWA to check whether a partition set is loaded.
3. PHPPSI communicates with program manager to load the partition set.
4. PHPPSI puts the name of the partition set in TPE (terminal partition extension) as the application partition set.
5. PHPPSC calls storage control to acquire a TIOA in which to build and free the original TIOA.
6. PHPPSC sets a slot in the TCTTE to be the partition set data stream concatenated with the terminal partition set name if the terminal is not in the base state.
7. PHPPIN places the AID and the cursor address in the TCTTE.
8. PHPPXE calls storage control to get a TIOA, retrieves the error message text by calling the message domain, fills the TIOA with data, transmits the data, and frees the TIOA.
9. PHPPSC references the partition set object to build the partition creation data stream.

DFHRLR (route list resolution program)

The route list resolution program (DFHRLR) builds terminal type parameters (TTPs), which are the main blocks for building and writing out data in BMS.

Figure 13 shows the route list resolution program interfaces.

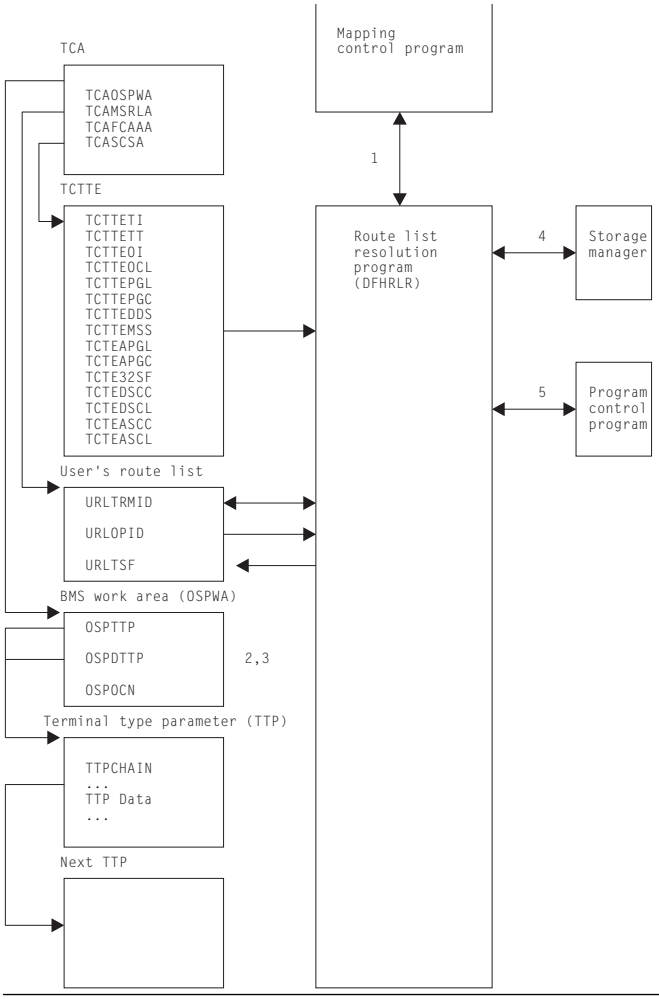


Figure 13. Route list resolution program interfaces

Note:

1. DFHRLR is called by the mapping control program (DFHMCP) to determine the grouping of terminal destinations.
2. If data is to be routed, DFHRLR groups the terminals in the user's route list by terminal type and builds a routing TTP for each type. For each TTP, the supported attributes of the corresponding terminals are accumulated. The address of the first routing TTP in the chain of TTPs is placed in OSPTTP.
3. If data is not to be routed, a direct TTP is built for the originating terminal and its address is placed in OSPDTTP.
4. DFHRLR communicates with storage control to acquire storage for the TTP.
5. Program manager services are requested by means of an ABEND command if errors occur that cannot be corrected.

DFHTPP (terminal page processor)

The terminal page processor (DFHTPP) directs completed pages to a destination specified in the BMS output request:

- SEND MAP or SEND TEXT sends to the originating terminal

- SEND MAP PAGING or SEND TEXT PAGING directs to temporary storage
- SEND MAP SET or SEND TEXT SET directs to a list of completed pages that are returned to the application program.

Figure 14 shows the relationships between the terminal page processor and other components in response to BMS output requests.

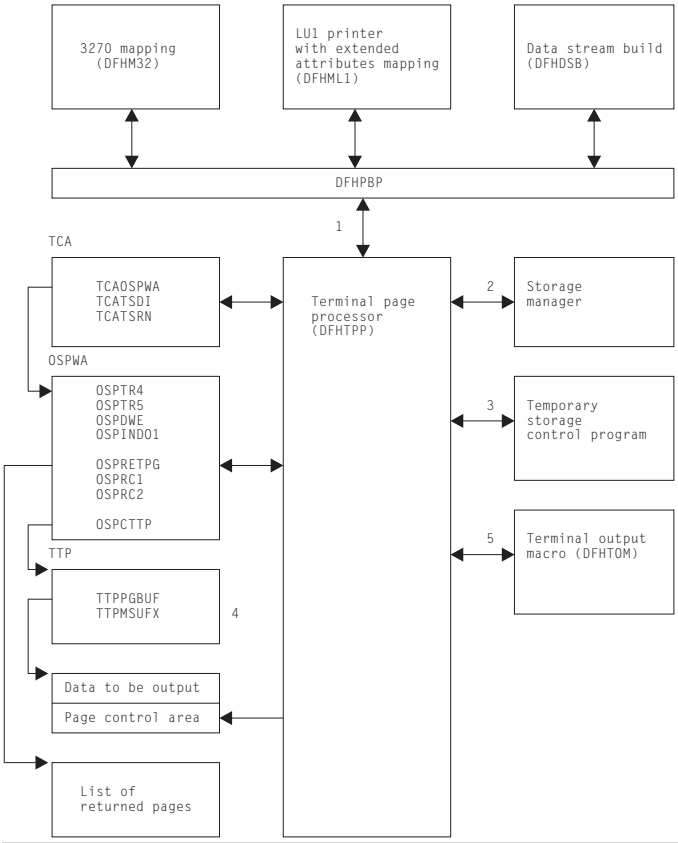


Figure 14. Terminal page processor interfaces

Note:

1. DFHTPP is entered from DFHPBP after processing by 3270 mapping (DFHM32) for 3270s, by LU1 printer with extended attributes mapping (DFHML1) for those LU1 printers, and by data stream build (DFHDSB) for other devices.
2. DFHTPP communicates with storage control to obtain:
   - The return list (to store the address of completed pages to be returned to the program)
   - Deferred work elements (DWEs), which ensure that message control information is written to disk, even if the program neglects to issue a SEND PAGE request
   - Storage for a list that correlates pages on temporary storage with the logical device codes for which they are destined.
3. Temporary-storage control is used to store pages and the message control record (MCR) for messages stored on temporary storage.
4. The terminal type parameter (TTP) controls the formatting of a message for a particular terminal type (for example, an IBM 2741 Communication Terminal). TTPPGBUF contains the address of a completed page.

5. The terminal output macro (DFHTOM) is issued to provide an open subroutine assembled within DFHTPP that puts a completed page out to the terminal. If the data stream contains extended attributes, and the terminal does not support extended attributes, the extended attributes are deleted.

DFHTPQ (undelivered messages clean up program)

The undelivered messages clean up program (DFHTPQ) checks the chain of automatic initiate descriptors (AIDs) to detect and delete AIDs that have been on the chain for an interval exceeding the purge delay time interval specified by the PRGDLAY system initialization parameter, if this has a nonzero value.

Figure 15 shows the undelivered messages clean up program interfaces.

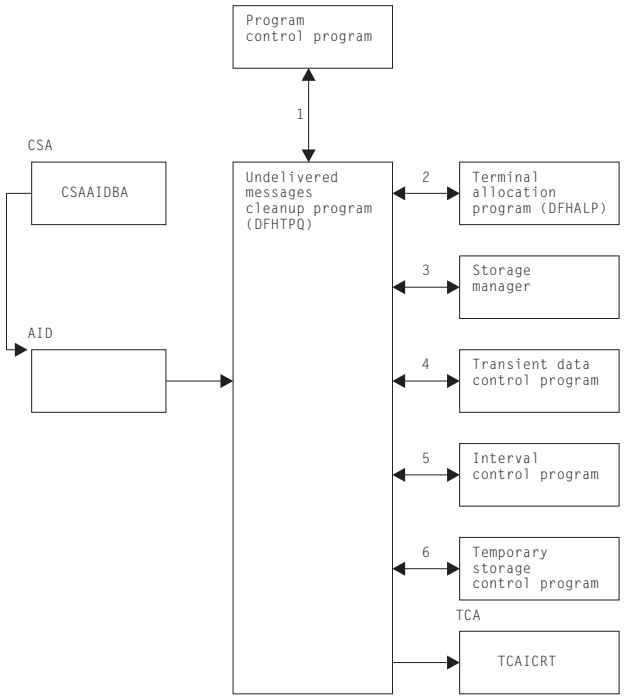


Figure 15. Undelivered messages clean up program interfaces

Note:

1. DFHTPQ is initiated the first time by the mapping control program (DFHMCP), by interval control, or by the transaction CSPQ. Thereafter, it reinitiates itself (see note 5).
2. DFHTPQ communicates with the allocation program (DFHALP) to locate and unchain AIDs.
3. DFHTPQ communicates with storage control to free AIDs that have been purged and to acquire storage for notification messages.
4. Transient data control is used to send notification messages.
5. Interval control is used to obtain the current time and to reinitiate this task (DFHTPQ).
6. DFHTPQ communicates with temporary-storage control to retrieve and replace message control records (MCRs) and to purge messages.

DFHTPR (terminal page retrieval program)

The terminal page retrieval program (DFHTPR) processes messages built by BMS and placed in temporary storage.

Figure 16 shows the relationships between the components of page retrieval.

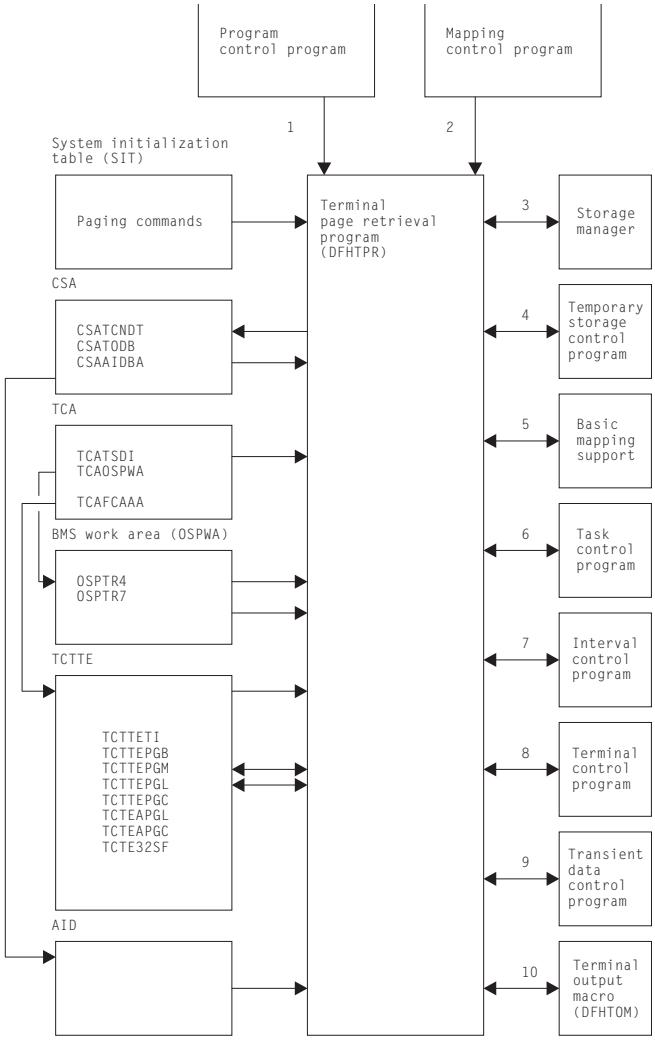


Figure 16. Page retrieval program interfaces

Note:

1. DFHTPR can be initiated as a stand-alone transaction (CSPG), or by a user-defined paging command (for example, P/, or 3270 PA/PF keys), or linked to from a BMS conversational operation (SEND PAGE request with CTRL=RETAIN or RELEASE).

   DFHTPR performs the following functions:

   - Displays the first page of a routed message
   - Displays subsequent pages of a message at a terminal for which a SEND PAGE request with CTRL=AUTOPAGE was specified
   - Processes paging commands from a terminal
   - Processes the CSPG transaction when it is entered at the terminal

- Purges a message displayed at the terminal if the terminal is in display status and other than a paging command is entered at the terminal.

2. DFHTPR is entered from the BMS mapping control program (DFHMCP) to display the first page of a message originated at the terminal if CTRL=RETAIN was specified in the BMS request. DFHTPR reads from the terminal and processes paging commands until other than a paging command is entered.
3. DFHTPR uses storage control to:
   - Acquire and free message control blocks (MCBs)
   - Free message control record (MCR) storage
   - Acquire storage for information and error messages to be sent to the destination terminal and the master terminal
   - Free an automatic initiate descriptor (AID) taken off the AID chain
   - Acquire and free storage for a route list constructed in response to a COPY command entered at a terminal
   - Acquire a TIOA into which to place a device-independent page when performing the COPY function.
4. Temporary-storage control is used to retrieve and replace MCRs and to retrieve and purge pages.
5. Basic mapping support is used to display error and information messages at a requesting terminal, and to send a page to the destination terminal in the COPY function.
6. Task control is used to retain exclusive control of an MCR while it is being updated.
7. DFHTPR communicates with interval control during error processing when a temporary-storage identification error is returned while attempting to retrieve an MCR. Up to four retries (each consisting of a one-second wait followed by another attempt to read the MCR) are performed. (The error may be due to the fact that an MCR has been temporarily released because another task is updating it. If so, the situation may correct itself, and a retry is successful.)
8. Terminal control is used to read in the next portion of terminal input after a page or information message is sent to the terminal when a SEND PAGE request with CTRL=RETAIN was specified.
9. Transient data control is used to send error or information messages to the master terminal.
10. The terminal output macro (DFHTOM) is issued to provide an open subroutine that puts a completed page out to the terminal.

DFHTPS (terminal page scheduling program)

The terminal page scheduling program (DFHTPS) is invoked for each terminal type to which a BMS logical message built with SEND MAP PAGING or SEND TEXT PAGING is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status.

Copy books

Copy book	Function
DFHBMSCA	Defines constants for field attribute values, flags returned by BMS, and character attribute types and values for SEND TEXT. It is usually copied into BMS application programs.

Copy book	Function
DFHMCPE	Included in the minimum-function BMS mapping control program DFHMCPE$, and also forms the BMS fast-path module DFHMCX used by both standard and full-function BMS. It is a small, fast, self-contained, limited-function BMS for 3270 displays and printers.
DFHMCPIN	Included in the standard and full-function versions of the BMS mapping control program, DFHMCPA$ and DFHMCP1$ respectively. It contains the code for input mapping.
DFHMIN	Included in the DFHM32 and DFHMCPE programs. It contains input mapping code for 3270 terminals.
DFHMSRCA	Defines constants for MSR control. This is usually copied into BMS application programs.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for basic mapping support, all with a trace level of BM 1:

- AP 00CD, for temporary-storage errors
- AP 00CF, for exit trace
- AP 00FA, for entry trace.

Builders

The builder modules:

- Make the autoinstall process possible (that is, build a terminal control table terminal entry (TCTTE) dynamically).
- Allows new TCT entries to be added on a running CICS system.
- Allow the TCT to be dynamically updated on a running CICS system.
- Allow TCT entries to be deleted on a running CICS system.
- Reduce emergency restart times for those systems that use the autoinstall function. These systems have to take the time to restore and recover only those terminals that were autoinstalled at the time of termination.
- Reduce warm start times for those systems that use auto-install. No auto-installed terminals (except LU6.2 parallel systems are recovered at warm start).
- Reduce shutdown times for those systems using auto-install. Auto-install catalog entries are deleted but the entry in storage is not destroyed during shutdown.

In this section, the term TCTTE is used in a general way to refer to the terminal control table entries for connections (TCT system entries, TCTSEs), mode groups (TCT modegroup entries, TCTMEs), sessions (session TCT terminal entries, TCTTEs), skeletons (TCTSKs), and models.

To build or delete a control block for a particular device, a set of builders is called. The set of builders is specified by a tree structure of patterns, each pattern specifying one builder.

The builder modules (DFHBS*) are link-edited together into the DFHZCQ load module.

Design overview

What is a builder (DFHBS*)?

A builder is responsible for all the actions that can occur on a particular subcomponent of the TCTTE. The term subcomponent means a separately obtained area of storage which is referenced from the TCTTE or a collection of fields in the TCTTE that are logically associated with one another. General terms sometimes used instead of subcomponent are **object** or **node**. For example, the NIB descriptor, LUC extension, and BMS extension are all considered to be subcomponents.

Builder parameter set (BPS)

Each time a calling module invokes DFHZCQ for INSTALL, it supplies a builder parameter set (BPS). The BPS describes the device to be defined. The device-type is determined by matching attributes in the BPS with a table of definitions, DFHTRZYT, in module DFHTRZYP.

A BPS consists of a fixed-length prefix, a bit map preceded by its own length, an area for fixed-length parameters preceded by its own length, and three variable-length parameters, BIND, USERID, and PASSWORD. Each variable-length parameter has a 1-byte length field.

TCTTE creation and deletion

This section starts by describing the structure of the main components involved in the process of creating and deleting TCTTEs. Figure 17 on page 1495 is in two halves: the top half shows those components that can initiate the process of collecting all the necessary data or parameters that go toward fully defining a TCTTE, and the bottom half is concerned with how to go about creating the TCTTE after it has the full set of parameters. Thus, all the processes are aiming for the same common interface. This section deals first with the top-level processes that are activated to create or delete TCTTEs; for the time being, assume that after returning from the DFHZCQ interface a TCTTE has been created. (For a more detailed description, see “DFHZCQ and TCTTE generation” on page 1496.)

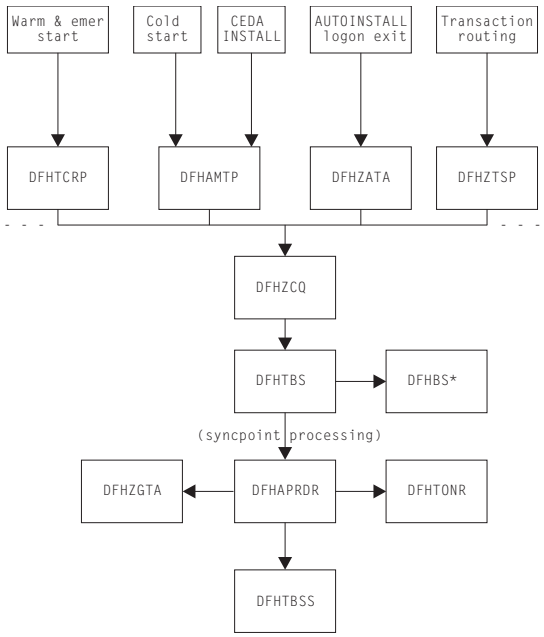


Figure 17. Top-level view of the components participating in TCTTE creation

Component overview

DFHTCRP: The DFHTCRP program is responsible for reestablishing the TCTTEs that were in existence in the previous run. There are conceptually three stages of processing in this module:

1. Initialize DFHZCQ. Initialize DFHAPRD. If START=COLD, terminate.
2. Reestablish TCTTEs that were saved on the CICS catalog. If START=WARM, terminate.
3. Call DFHAPRDR to forward-recover in-flight TCTTEs from the system log, if an emergency restart is being performed.

DFHAMTP: The DFHAMTP program is used as part of INSTALL processing. It calls DFHTOR, then DFHZCQ.

DFHZATA and the CATA transaction: CATA is a transaction that is initiated by the logon exit and causes DFHZATA to run. It is passed the CINIT which is used to deduce the parameters which must be passed to DFHZCQ in order to create a TCTTE.

DFHZTSP: The terminal sharing program, DFHZTSP, is used by transaction routing for devices of all types, exclusively so for non-APPC devices.

DFHZCQ: The DFHZCQ program supports the INSTALL and DELETE interface that results in the TCTTE being created or deleted. It relies on its callers to supply the complete set of parameters that are to be used to create the TCTTE; that is, it is not responsible for determining parameters for the TCTTE.

DFHBS* builder programs: The builders are responsible for creating the TCTTE. The parameters given to DFHZCQ are passed on to the builders. They extract the parameters and set the relevant fields in the TCTTE.

DFHTBS: The DFHTBS program is an interpreter that uses a pattern given to it by DFHZCQ to drive the whole TCTTE creation or deletion process according to certain rules.

DFHAPRDR: The DFHAPRDR program is the orchestrator of the commitment of TCTTE creation or deletion. It is responsible for driving DFHTBSS and DFHTONR for syncpoints, during cold start and also for recovering in-flight creates or deletes from the system log during emergency restart. It is called by the Recovery Manager, DFHTCRP and DFHAMTP during start-up and directly from DFHTBS (to roll-back an atom).

DFHTBSS: The DFHTBSS program is responsible for logging forward recovery records and for updating the catalog as a result of the request initiated by DFHZCQ and actioned by DFHTBS. It is driven by DFHAPRDR.

DFHTONR: The DFHTONR program is responsible for logging forward recovery records and for updating the catalog for install or delete requests for TYPETERMS. It is driven by DFHAPRDR.

DFHZGTA: DFHZGTA is the module called by DFHBS* and DFHZTSP (for remote system entry sessions) to add or delete index entries for TCTTE entries. It maintains locks on terminal namespaces, and handles calls to TMP to add, quiesce, delete, unlock and unquiesce entries. It is driven at syncpoint or rollback for an atom by DFHAPRDR.

DFHZCQ and TCTTE generation

This topic describes how a TCTTE gets built and deleted. You need to understand at least one method by which a builder parameter set (BPS) is created; for example, CEDA INSTALL or AUTOINSTALL. A BPS contains all the values necessary for the creation of a TCTTE.

Figure 18 gives a more detailed view of the main components involved in the INSTALL process.

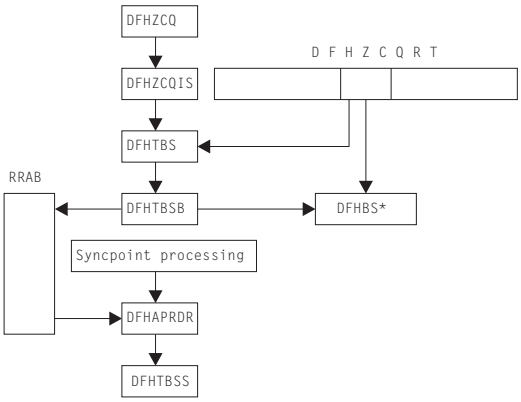


Figure 18. Major active components in the INSTALL process

The four-stage process

In summary, the process consists of four stages:

1. **Collecting the parameters** together.
2. **Creating the storage** for the TCTTE and copying the parameters. Note however, that at the end of this stage, a TCTTE has effectively been built. It is

still unknown to the rest of the CICS system, that is, the TCTTE name has not been exposed. The modules involved here are DFHTBSB and DFHBS*.

3. **Producing a recovery record**. This is done at syncpoint processing time in the DFHTBSS module. This stage is usually called Phase 1 syncpoint.
4. **Writing or updating the catalog**. Again, this is done in DFHTBSS and is called Phase 2 syncpoint. It is at about this stage that the TCTTE name becomes exposed and known to the rest of CICS.

What is DFHZCQRT?: DFHZCQRT is an array of "patterns" where each pattern defines a list of builders that need to be called in order to create this particular type of TCTTE, that is, a pattern is equivalent to a type of terminal. The array entry consists of two parts: information that is private to DFHZCQ, and the pattern that is interpreted by DFHTBS.

What does DFHTBSBP do?: The pattern entry is passed to DFHTBSBP (via DFHTBSB) after it has been found by DFHZCQIS. DFHTBSBP calls each builder identified by the pattern in sequence to create the object for which the builder is responsible. Note that DFHTBSBP knows nothing about the TCTTE; DFHTBSBP merely follows a set of simple rules. It keeps an audit trail of each builder that is called.

What is the RRAB used for?: The audit trail kept by DFHTBSBP is implemented by obtaining a Resource definition Recovery Anchor Block (RRAB) that has some user storage attached to it. As DFHTBSBP calls each builder to perform an action, it adds an "action element" to the RRAB. (See "What is syncpointing?" on page 1498) The address of the RRAB for a UOW is held in the 'APRD' recovery manager slot, which ensures that DFHAPRDR will be called at syncpoint. The RRAB stores the action blocks in two types of chains, one for actions that are not part of a named resource definition 'atom' and one for actions that are part of a named atom. This later type are chained off a Resource definition Action Name block (RABN). Also held in the RRAB is an indicator set by DFHTOR if DFHTONR should be called at syncpoint (if a typeterm has been installed), and a chain of Resource Definition Update Blocks (RDUB).

What is a resource definition 'atom'?: Certain resource definitions must be installed or deleted as a single set. These definitions are called a resource definition 'atom'. CICS installs the members of a RDO group as individual resource definitions, which can fail without causing the other resources to fail except for these atoms, which bear the name of the logical set of definitions. For example:

A connection and its associated sessions
: is named for the connection

A pool of terminals
: is named for the pool of terminals

What is a Resource definition Atom Name block (RABN)?: The RABN is only created for those atoms of resource recovery that are named. It holds the name of the atom, a chain of action elements for the atom, and the recovery outcome of the atom (whether it failed and was backed out, or succeeded and should be committed). DFHTBSB uses the RABN to decide if a session definition should not be installed because the installation of the parent connection has already failed, for example. In our auto-install example, if the definition being installed is a parallel connection, there will be a RABN for it from which the action elements are chained.

What is a Resource Definition Update Block (RDUB)?: The RDUB is a record of locks held by a UOW against names in three namespaces:

1. Termids and Sysids
2. Netnames
3. Unique ids (Composed of the Netname of a Terminal Owning Region followed by a period '.' followed by the Termid or Sysid in that TOR)

During the installation, deletion, or replacement of a TCTTE definition the builders DFHBS* obtain locks by calling DFHZGTA. These locks guarantee exclusive or shared access to names in these namespaces. Exclusive access is used to prevent another task from installing another definition with the same name, netname or unique-id while this UOW is trying to install or delete (an action which may have to be reversed). Shared access is used to block another task from deleting an entry that a definition that this task is updating (for example, a system definition name may be locked by a remote terminal definition that refers to it).

RDUBs also exist on a global chain so that other UOWs can easily find out if a particular lock is held.

What is syncpointing?: When DFHTBSBP has exhausted the list of builders, it returns to its caller. Similarly, DFHZCQIS returns to its caller, which could have been autoinstall. However, there is still an audit trail that is attached to the RRAB. It is only when the calling task terminates or issues DFHSP USER or EXEC CICS SYNCPOINT that the next two stages occur.

Syncpoint processing consists of two phases. The first phase (prepare phase) requires the resource manager to write a forward-recovery record to the log. Thus, if the second phase (commit phase) fails to write to the catalog, this recovery record can be used to forward-recover the action on an emergency restart.

DFHTBS: The DFHTBS program is an interpreter that uses a pattern given to it by DFHZCQ to drive the whole TCTTE installation or deletion process according to certain rules.

DFHAPRDR: DFHAPRDR is invoked by recovery manager if the 'APRD' RM slot is non-zero. This slot contains the address of the RRAB for this UOW if any resource definition has taken place. It is also called by DFHTBS directly if an atom needs to be rolled-back or to commit an atom during Cold Start. DFHAPRDR examines the RRAB and chooses whether to call DFHTBSS, DFHTONR and DFHZGTA for each phase of syncpoint or individual atom commitment.

If either DFHTBSS or DFHTONR have records to log/catalog, DFHAPRDR calls the recovery manager to request that a record is written to the catalog noting that a forget record will be written once syncpoint completes. The purpose of this call is that if CICS should fail between the start of syncpoint phase 2 and the end, on an emergency restart recovery manager will call DFHAPRDR with the log records for this UOW so that they can be re-applied to the catalog, and the TCTTE entry or entries can be re-built.

DFHTBSS: The DFHTBSS program is responsible for performing the correct recovery actions for each atom and UOW at syncpoint (or during the rollback of an individual atom). It writes forward recovery records to the system log and updates the catalog during phase 1 and phase 2 of syncpoint respectively. It is directly driven by DFHAPRDR.

The purpose of the builder (DFHBS*) modules is to build a TCTTE, TCTSE, and TCTME and its associated control blocks. A TCTTE is built for terminals only; a TCTSE and TCTME are built for both LU6.1 with MRO and LU6.2 single sessions; all three are built for LU6.2 parallel sessions. DFHTBSS is invoked by DFHAPRDR with a parameter list that indicates whether this call is for an individual atom or for syncpoint and which phase is in force. For phase 1, it uses the action blocks audit-trail to recall each builder. It asks each builder to supply the address and length of the subcomponent so that it can create a single record containing a copy of each component as a list; that is, the first part of the record contains a copy of the object created by the first builder in the sequence, the second part contains a copy of the object created by the second builder, and so on until the audit trail list is finished. This record is then written to the system log as a forward recovery record.

When DFHTBSS is reentered for the second phase (again a parameter on the call by DFHAPRDR), it uses the record created in the first phase as the record that is written to the catalog. During this stage, each builder is called to tidy up after the object for which it is responsible; for example, for the TCTTE itself, it puts the TCTTE in service.

Again note, DFHTBSS only implements a set of rules.

DFHTONR: DFHTONR is responsible for writing catalog records for TYPETERMs. It is called by DFHAPRDR.

DFHZGTA: DFHZGTA is the module that is called by DFHBS* modules to add index entries for TCTTE entries so that they can be located quickly either by DFHZLOC, DFHZGTI or in z/OS Communications Server exit code. It calls DFHTMP services. It obtains and releases locks using the RDUB blocks, and at syncpoint is responsible for releasing all TMP locks and unquiescing any TMP entries that were quiesced by DFHBS* modules.

Summary:

- In overview, the process consists of four stages: parameter collection, obtaining and initializing, phase 1 recovery record and logging, and phase 2 catalog record.
- A builder contains TCTTE specific code.
- DFHTBS* modules implement the abstract rules for creating generic “objects”.
- DFHZCQRT contains patterns that define what builders are to be used to build the TCTTE.
- Syncpoint processing consists of two stages (prepare and commit).
- DFHAPRDR is responsible for orchestrating the syncpoint process for all of resource definition recovery.
- DFHTBSS is driven by DFHAPRDR using the audit trail produced by DFHTBSB.
- DFHTONR is driven by DFHAPRDR if any TYPETERMs were installed.
- DFHZGTA is driven by DFHAPRDR if any locks need to be released.

Example of an autoinstall: Consider the following: a terminal operator has logged on to the system and is being autoinstalled. The CATA transaction is responsible for collecting together the parameters required for the DFHZCQ INSTALL.

The process continues from the point where the DFHZCQ INSTALL is issued from CATA:

1. A call has been made to cause an installation to occur. DFHZCQ ensures that other related modules are already loaded.
2. DFHZCQ calls the installation-specific module (given in the parameter block passed to DFHZCQ)
3. DFHZCQIS performs various checks on the parameters passed by the caller of DFHZCQ.
4. DFHZCQIS finds a pattern in DFHZCQRT that matches with information given in the parameters.
5. DFHZCQIS calls DFHTBS with the pattern and parameters.
6. DFHTBS routes the request to DFHTBSB; it is omitted from further discussions.
7. DFHTBSB checks that a valid pattern has been passed.
8. DFHTBSB creates the RRAB which gets attached to the APRD Recovery Manager slot.
9. DFHTBSB calls the next builder as defined by the pattern.
10. Each builder (DFHBS*) creates its section of the TCTTE.
11. DFHTBSB adds an action element to the RRAB giving information about this particular builder.
12. Steps 9, 10, and 11 are repeated until the pattern is finished.
13. DFHTBSB tidies up the RRAB and returns.
14. DFHTBS returns.
15. If the return code was 'OK', DFHZCQIS returns the address of the hidden TCTTE.
16. DFHZCQ returns.
17. The caller continues until DFHSP USER is issued or the task terminates.
18. DFHAPRDR invokes DFHTBSS with the RRAB indicating phase 1.
19. DFHTBSS examines the RRAB to determine phase.
20. Using the action elements created in step 11, DFHTBSS recalls each builder asking for information to be saved on the recovery log.
21. Each builder (DFHBS*) returns the address of the object built in step 10.
22. Using these addresses, DFHTBSS builds the recovery record.
23. DFHTBSS writes the recovery record to the system log.
24. DFHTBSS saves the stored version for the next phase.
25. DFHTBSS returns.
26. Recovery Manager calls all other resource managers that have a part to play in the process; it knows this because there are addresses in the RM slots for this UOW.
27. DFHTBSS is called for phase 2. It reuses the in-storage version of the recovery record to write to the catalog.
28. DFHTBSS returns.

Patterns, hierarchies, nodes, and builders

Patterns were introduced in the previous section. This section examines in detail what they look like. To achieve this, several terms have to be explained.

What is a hierarchy?: In this context, “hierarchy” is another word for tree. The structure of the TCTTE can be thought of as a tree: at the top **node** is the TCTTE itself, containing pointers to lower-level **nodes**.

Figure 19 shows the **master node** as the TCTTE, with **subnodes** connected to it (BMS extension, special features extension, and so on).

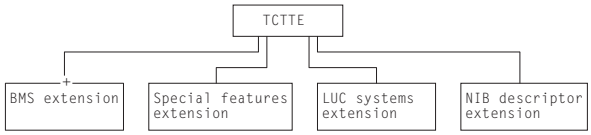


Figure 19. TCTTE structure

As a result of this structure, it can be seen that the creation process must follow several rules. For example, the storage for the **master node** has to be obtained before pointers to **subnodes** are saved in it.

What is a pattern?: The objective of a pattern is to reflect or represent the hierarchy. Figure 20 outlines the shape of a pattern. For each of the nodes in Figure 19, there is a pattern. Starting with the TCTTE (*the master node*), there is a *master pattern*. B1offset references the *subpattern* for the BIND image node; B2offset references the subpattern for the BMS extension node; B3offset and B4offset reference the subpatterns for user area and SNTTE *subnodes* respectively. In total, there are five patterns: the master pattern and four subpatterns. So what is meant by *pattern* is really a collection of patterns.

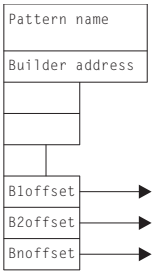


Figure 20. Pattern structure

Each pattern contains the address of a builder, so we could represent the TCTTE structure as:

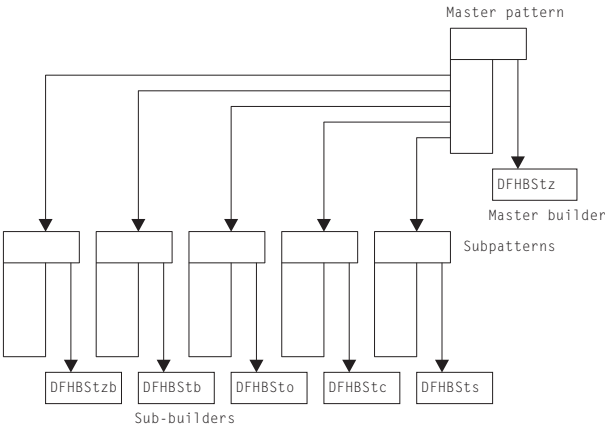


Figure 21. Patterns and subpatterns

The purpose of the builders: The purpose of the builders is to centralize the major functional code for creation and deletion of the **nodes** associated with the TCTTE. Figure 20 on page 1501 and Figure 21 on page 1501 show how the **patterns** refer to the builders; the pattern is exploited by the DFHTBS* code to activate the relevant builder function. For example, DFHTBSBP, when given a pattern, extracts the address of the builder and invokes the BUILD function belonging to the builder.

How does DFHTBSBP do its work?: First, you must examine more closely the structure of a builder in Figure 22.

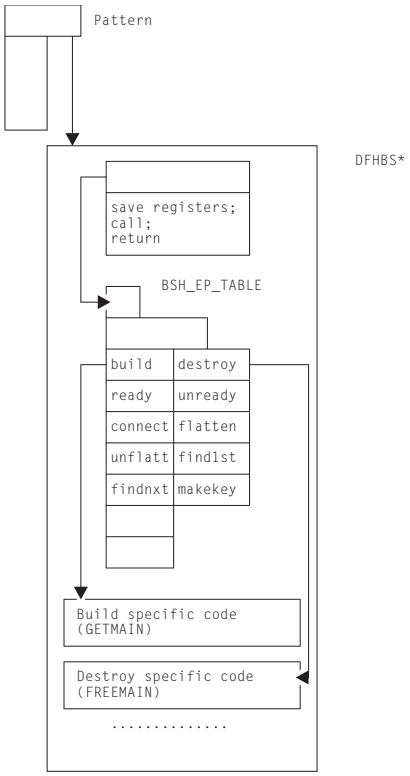


Figure 22. The builder stub

Remember that the pattern references a builder. In fact, it references a stub, the first word of which points to a table (BSH_EP_TABLE), and is followed by code that is responsible for enacting the entry as required by the caller. For example, if the caller wanted to call BUILD, a call would be made to the stub with value 1. The stub would extract the offset to the build code from the BSH_EP_TABLE, and perform the call.

Thus, making a call from DFHTBS* to DFHBS* is relatively simple: all that is needed is the function number (1 for BUILD, 2 for DESTROY, ...), a call to the stub, and the pattern.

Summary

- The TCTTE is structured as a **hierarchy** with a **master node** (the TCTTE itself) and **subnodes** (BIND image, BMS extension, and so on).
- **Patterns** mimic this hierarchy and consist of a **master pattern** which refers to **subpatterns**.

- In turn, each pattern points to a builder: the master pattern refers to the **master builder** and the subpatterns refer to the **sub-builders**.
- Builders centralize the major creation and deletion functions associated with the node for which they are responsible.
- The invocation (or activation) of the builder functions is performed under the strict control of the DFHTBS* modules.
- The **order of invocation** is totally determined by the structuring of the patterns.

The DELETE process

By examining the hierarchy (see Figure 19 on page 1501), you can see that there are certain rules that have to be established. Firstly, check that the TCTTE and its subcomponents are quiesced; that is, there is no activity in progress. Secondly, the top node must not be the first object to be freed. From this, you can derive two basic rules, or functions, that must be supplied by any DFHBS*:

UNREADY
: For all nodes associated with the master node. Ensures that no activity is occurring; for example, that a CLSDST is not in progress. It must also achieve exclusive ownership of the object; for example, ZGTA QUIESCE ensures no locates on the given TCTTE succeed and that no other UOWs can install another similarly named object until syncpoint. Further, it **initiates** the ZGTA DELETE, which does a TMP DELETE to remove the entry.

DESTROY
: Lower objects first. See "What about the "lower objects first" rule?." Frees the storage that belongs to the node.

What about the "lower objects first" rule?: Figure 23 tries to add meaning to the descriptions of the UNREADY and DESTROY functions. As each builder is called (as determined by the master pattern), DFHTBSD records an audit trail of called builders. However, the audit trail is managed slightly differently for the delete process, to guarantee order of processing by DFHTBSS at phase 2 time. For further information, see "Completing the process description" on page 1505.

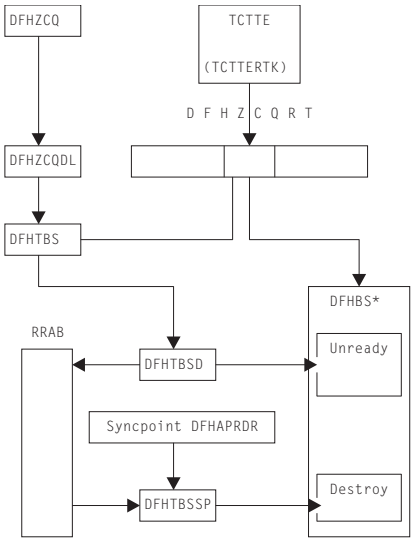


Figure 23. Major active components in the DELETE process

Example of a reinstall:

1. CEDA reads the CSD and converts the definition into a builder parameter set (BPS).
2. CEDA issues a DFHZCP INSTALL passing the BPS.
3. Using the resource type code in the BPS, DFHZCQIS searches the DFHZCQRT table for the associated pattern.
4. DFHZCQIS calls DFHTBSB passing the BPS and the pattern.
5. DFHTBSB checks the pattern and creates a resource definition recovery action block (RRAB) for the audit trail.
6. Using the pattern, DFHTBSB calls the CHECKSET entry point of the associated builder.
7. The master builder does a DFHZGTI LOCATE to check whether the TCTTE already exists.
8. A TCTTE is found to exist, so the builder issues DFHZCP DELETE passing the address of the old TCTTE.
9. When a TCTTE is created, its position within the DFHZCQRT table is saved in the TCTTE. DFHZCQDL uses this value to find the pattern associated with this TCTTE.
10. DFHZCQDL calls DFHTBSD passing the object to be deleted and the pattern.
11. DFHTBSD extends the audit trail so that information about this delete can be recorded.
12. DFHTBSD calls the UNREADY entry of each builder.
13. Each builder (DFHBS*) checks whether its part of the TCTTE is being used (and vetoes the UNREADY if it is). It calls ZGTA QUIESCE and ZGTA DELETE to lock and remove the index entries.
14. DFHTBSD updates the audit trail for each called builder.
15. DFHTBSD returns.
16. DFHZCQDL returns.
17. The master builder checks the return code (that is, that no builder vetoed the UNREADY).
18. The master builder returns.
19. DFHTBSB checks the return code and recalls each builder at the BUILD entry point passing the BPS.
20. Each builder obtains some storage and copies the parameters from the BPS. It uses ZGTA ADD calls to lock and add index entries
21. DFHTBSB tidies up the RRAB and returns.
22. DFHZCQIS records the position within DFHZCQRT that enables DFHZCQDL to find the pattern.
23. DFHZCQIS Returns.
24. CEDA checks the return code and issues DFHSP USER.

    Note: At this stage there are two TCTTEs: the old one that was UNREADY and the new one.
25. CEDA calls: DFHTBSS is entered for the first time (phase 1). The audit trail consists of two parts (A and B). Part A contains the list of builders involved with the UNREADY; part B contains the list of builders that created the new TCTTE.
26. CEDA writes a recovery record to the system log for Part A indicating that a delete is about to take place in phase 2.

27. CEDA creates a recovery record from Part B which represents the new TCTTE to be built.
28. CEDA calls each builder asking for its subcomponent (FLATTEN).
29. DFHZQIX returns an address and length.
30. CEDA concatenates each subcomponent into the recovery record.
31. CEDA writes the recovery record to the system log.
32. CEDA returns (end of phase 1).
33. CEDA reenter for phase-2 processing.
34. CEDA processes Part A, calling the DESTROY entry for each builder.
35. Each builder frees its part of the old TCTTE.
36. CEDA processes Part B of the audit trail.
37. CEDA writes the recovery record to the catalog.
38. CEDA calls the READY entry point for each builder on the audit trail.
39. Each builder does any tidying up that needs to be done.
40. CEDA returns.

Completing the process description

To complete the description of the creation and deletion process, two further functions must be described: CONNECT and READY.

CONNECT: Figure 19 on page 1501 shows the TCTTE hierarchy. All that has happened at build time is that the separate parts of the TCTTE have been obtained. Access to these subcomponents is achieved by referencing pointers that are held in the TCTTE. So the CONNECT builder entry point is used to join the subcomponent to the TCTTE.

READY: The READY builder entry point is provided to enable any final tidying up that may be required at the end of the build process. For example, if the TCTTE has the AUTOCONNECT option, a SIMLOGON is initiated from this entry point. In general, this entry point is rarely used.

The creation/deletion state machine: Figure 24 shows the symmetry between the various builder functions.

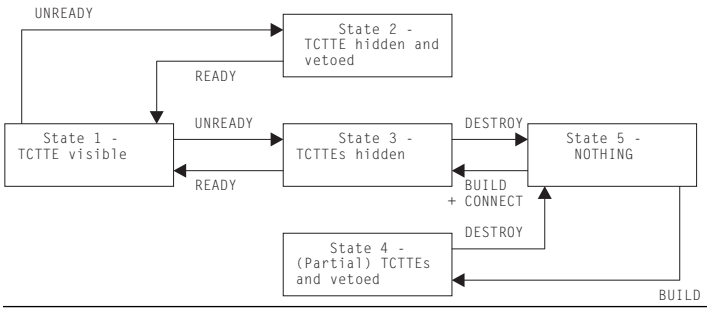


Figure 24. Create/delete state diagram

The starting point can be either state 5 (installing a TCTTE) or state 1 (deleting a TCTTE). Thus, if several TCTTEs had been successfully built, but the last one resulted in an error, we would end up in state 4. If it were not for the last one, we would have ended up in state 3. So the caller is returned an error response, and issues a DFHSP ROLLBACK. This causes DFHTBSS to call the DESTROY function of the builders for all elements on the audit trail—even for those that were

"successfully" built in this atom, or UOW. Thus, an install of a atom can be perceived as one complete unit. During the DESTROY process, if the atom is being rolled-back, the builders call ZGTA QUIESCE and ZGTA DELETE to remove index entries for the new TCTTE. Likewise during the READY process, if a delete is being rolled back, the builders call ZGTA ADD to re-instate index entries for the TCTTE.

The hierarchy and its effect upon the creation process

Summary so far

- Object creation is a four-stage process.
- It is controlled by a pattern.
- Each pattern refers to a builder.
- Each builder is responsible for a subcomponent of the TCTTE.
- Builders have a number of procedural entry points:
  - BUILD
  - CONNECT
  - DESTROY
  - READY
  - UNREADY.
- These entry points are called under the control of the DFHTBS components.

This section now looks in greater detail at how the control of the builder calling process is implemented. To do that, you need to understand in greater detail the structure of the hierarchy, and the way the DFHTBS components interpret that structure.

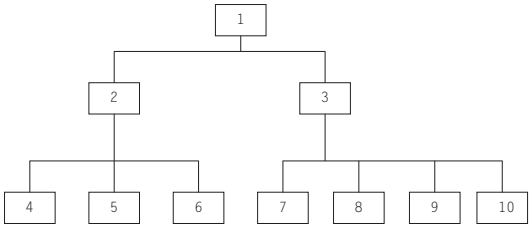


Figure 25. A general hierarchy

Figure 25 shows a more general hierarchy. Node 1 can be considered as a master node: it is at the top of the tree and has two subnodes (2 and 3). However, you could say that node 2 and its subnodes are also a tree: node 2 is the master node, and nodes 4, 5, and 6 are the subnodes. Similarly, with node 3: it has subnodes 7, 8, 9, and 10.

The DFHTBS components exploit the idea that a tree consists of a node with trees below it. In fact, DFHTBSBP uses **recursion** to access the tree of patterns.

Recursion: This section demonstrates how recursion is used to process a much simpler structure than that given in Figure 25. The example shown in Figure 26 on page 1508 is for the DFHTBSP program, which has the following parameters:
Input: PATTERN, HIGHERNODE, and BUILDER
Inout: AUDITTRAIL
Output:
NODE and RESPONSE.

The following list outlines the flow in DFHTBSBP. The step references refer to steps in this list.

1. Add and initialize an action to the AUDITTRAIL (this is used later in steps 5 and 11).
2. Using parameter PATTERN, find the address of the associated builder.
3. Call the builder stub with function number 1 (for BUILD) with the following parameters:
   Input: HIGHERNODE and BUILDER
   Output:
   NODE.

   The builder uses the BUILDER parameters to create its specific object. Storage is obtained and the parameters are copied into it.
4. Check that the response from the build is 'OK'.
5. Copy the address of the output parameter NODE into the AUDITTRAIL action.
6. Process all the subpatterns that may be attached to your pattern
7. Get the next subpattern Pn.
8. Call DFHTBSBP with the following parameters:
   Input: Pn, NODE, and BUILDER
   Inout: AUDITTRAIL
   Output:
   SUBNODE and SUBRESPONSE

   Note: In this step, you call yourself again, passing NODE. At the next level of recursion, this appears as HIGHERNODE.
9. Stop when the last pattern is processed.
10. Call the builder stub with function number 5 (for CONNECT) with the following parameters:
    Input parameters:
    NODE
    Inout parameters:
    HIGHERNODE

    The builder's CONNECT entry point now places the address as given by NODE into an offset of HIGHERNODE.
11. Finally, place the address of the pattern into the AUDITTRAIL action.

Simple recursion example:

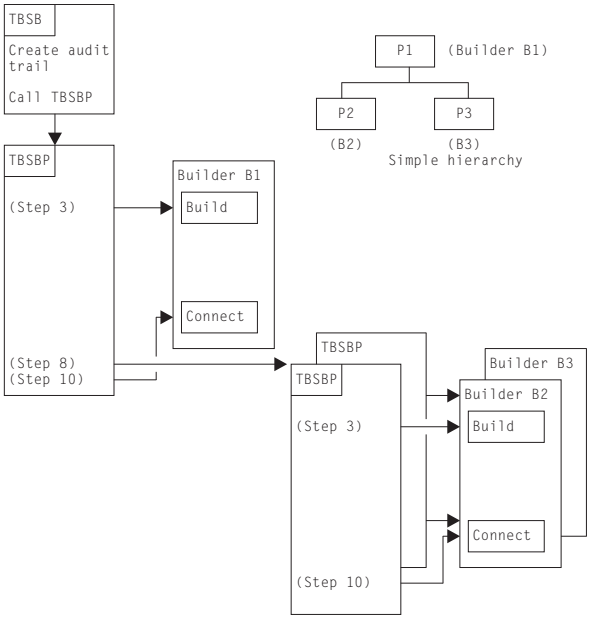


Figure 26. Simple example showing recursion

Consider the following simplified version of the hierarchy as given in Figure 26. The step references refer to steps in the list in the section "Recursion" on page 1506.

1. Start with pattern P1. Call its associated builder (step 3 on page 1507). This creates node N1.
2. All the patterns below P1 are processed, the first of which is P2.
3. Call DFHTBSBP passing P2, N1, BUILDER parameters, and others:
   a. Using the passed pattern (now P2), call the builder. This creates node N2.
   b. Process all patterns below P2; there are no subpatterns, so steps 6 on page 1507 through 9 on page 1507 6 on page 1507 are not performed.
   c. Call the CONNECT entry of the builder, passing higher node N1 and the node just created, N2. This makes N1 point to N2.
   d. Return to caller.
4. Get the next pattern, P3.
5. Call DFHTBSBP passing P3, N1, BUILDER parameters, and others:
   a. Using the passed pattern (now P3), call the builder. This creates node N3.
   b. Process all patterns below P3; there are no subpatterns, so steps 6 on page 1507 through 9 on page 1507 6 on page 1507 are not performed.
   c. Call the CONNECT entry of the builder passing in higher node N1 and the node just created N3. This makes N1 point to N3.
   d. Return to caller.
6. Last pattern processed (step 10 on page 1507).
7. Call the builder associated with P1 to connect node N1 to HIGHERNODE. (This is zero because there is no higher node. Usually, a master builder's CONNECT function either does nothing or adds the TCTTE name and address into the table management tables.)

ROLLBACK

What happens when an error occurs during the installation process? An example of this would be when one TCTTE within a group is still in service when a CEDA COPY command is being processed for the group with the REPLACE option specified. “Example of a reinstall” on page 1504 shows such a replace operation. The builders for the existing TCTTE are called (UNREADY) in order to check that the DELETE (FREEMAIN) can proceed. Thus, the audit trail refers to all called builders.

If the “total vote” from all the UNREADY builder calls indicates OK, the build proceeds for the new TCTTE that is to replace the existing one. Thus, at the end of the process, the audit trail consists of a list of references to builders associated with the old TCTTE, and a list of references to builders for the new TCTTE (lists A and B).

Consider the case when the group contains definitions for three TCTTEs, and a VETO occurs for the last one. This means that there is an audit trail for A1, B1, A2, B2 for which there was success, and list A3 for the unsuccessful UNREADY for the third TCTTE.

The failure condition is returned to the caller (CEDA), which then issues a DFHSP ROLLBACK.

Recovery Manager invokes DFHAPRDR which in turn invokes the DFHTBSS module, with a parameter that indicates a rollback is required. Thus, the “A” lists are processed, and all the READY entry points of the builders are called. Then the “B” lists are processed, and the DESTROY builder entry is called to free the storage obtained for the supposedly new TCTTEs.

To summarize, the rollback operation for UNREADY is READY, and the one for BUILD is DESTROY.

Catalog records and the CICS global catalog data set

Overview: The fourth stage of the process is to produce a catalog record that is written to the CICS global catalog data set. This record is used on a subsequent restart to re-create the TCTTE, but in a different way from the “Build” process as described. A CEDA INSTALL means that the TCTTE lives across CICS restarts, avoiding the necessity of rerunning the installation.

A RESTORE from the CICS catalog is a faster operation than a CEDA INSTALL because there is no conversion of the CSD definition to a builder parameter set, and less I/O involved.

In summary, a catalog record is produced by recalling each of the builders asking for the address of the data that they want to be recorded on the catalog. Each subcomponent of the TCTTE is then copied and concatenated into one record, which is then written to the catalog. This process is known as FLATTEN.

A CATALOG call is made when significant events change the state of a TCT entry which would be needed on a subsequent emergency restart. An example is the recording of the membername of a generic z/OS Communications Server resource connection when a bind has occurred for the first time.

On the restart, the record is read from the catalog, and presented back to each of the original builders. Each builder performs a GETMAIN, and copies its section of the recovery record into the acquired storage. This process is known as UNFLATTEN.

At shutdown, auto-installed entries are removed from the catalog with an UNCATALOG call (if they were cataloged because AIRDELAY¬=0). This drives DFHTBS and the builders to produce similar records to those for a DELETE call, but only to take action to delete the catalog record. This is significantly more efficient than calling the builders to DELETE each entry, as the copy in storage remains untouched.

The key and the recovery record: When the build process in DFHTBSBP has finally finished, this module makes a call to the master builder at the MAKEKEY entry point. The builder produces a key that is used to index the associated recovery record. (See Figure 27.)

This information is placed on the audit trail so that it can be picked up by DFHTBSS. It consists of two parts:

1. Information that allows access to the catalog
2. The recovery record header.

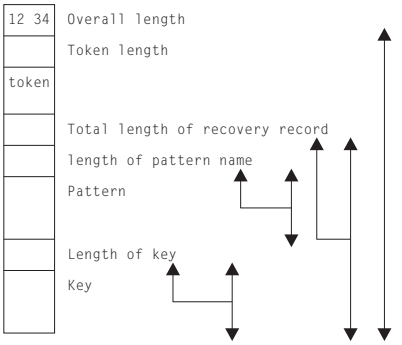


Figure 27. The recovery record

More about the audit trail: Figure 28 on page 1511 shows the layout of an audit trail. Internally it is known as an **action block**, which consists of **action elements**. As each builder is invoked by DFHTBSBP or DFHTBSDP, an action element is appended to the action block. Each element has a reference to a pattern (PATT). This is to allow DFHTBSS to enter the associated builder at the READY or DESTROY entry points.

CCRECP contains the address of the recovery record header. Only one of these is produced as a direct result of the MAKEKEY call to the **master builder**. All other action elements have their CCRECP set to zero.

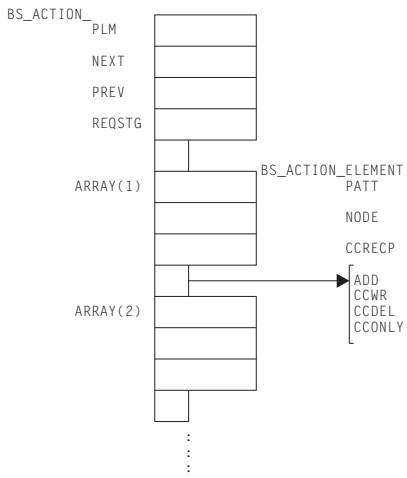


Figure 28. Action block and action elements (audit trail)

DFHTBSS and the FLATTEN process: During phase-1 syncpoint processing, DFHTBSS searches the action elements for a nonzero CCRECP. On detection, it calls DFHTBSLP, passing the reference to the pattern as given in the action element.

The storage "segments" are returned to DFHTBSSP which extracts the address and length from each segment and copies them into the recovery record.

The RESTORE process: The recovery record header contains the pattern name which is used to find the master pattern in DFHZCQRT. This is then passed to DFHTBSR to drive the recovery process by calling each builder's UNFLATTEN entry.

Each segment is extracted from the recovery record and is passed to the associated builder's UNFLATTEN entry point. These routines usually obtain some storage and copy the segment into it.

Control blocks

Builder modules all use both LIFO and a builder parameter set (BPS), which are passed between the CSECTs (DFHBS* modules). See "Builder parameter set (BPS)" on page 1494 for further information about the BPS.

Terminal storage acquired by the builders

The following terminal storage is acquired by the builders:

Control block field	Description	Storage manager subpool
TCTSE	Terminal control table system entry	ZCTCSE
TCTME	Terminal control table mode entry	ZCTCME
TCTTE	Terminal control table terminal entry	ZCTCTTEL (large TCTTEs) ZCTCTTEM (medium TCTTEs) ZCTCTTES (small TCTTEs)
TCTENIBA	NIB descriptor	ZCNIBD
TCTEBIMG	BIND image	ZCBIMG
TCTTECIA	User area	ZCTCTUA
TCTTESNT	Signon extension	ZCSNEX
TCTELUCX	LUC extension	ZCLUCEXT

```
TCTTETEA       BMS extension            ZCBMSEXT
TCTTETPA       Partition extension      ZCTPEXT
TCTTECCE       Console control element  ZCCCE
```

TCTTE layout

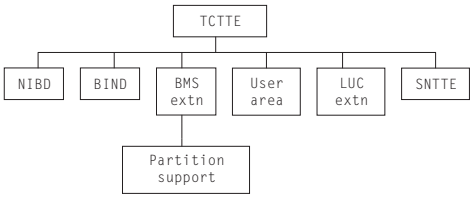


Figure 29. TCTTE layout

Formatted dumps give the TCTTE first, followed by its supporting control blocks.

Terminal definition

CEDA DEFINE puts a definition on the CSD. The definition is in the form of a CEDA command.

CEDA INSTALL reads the definition from the CSD, calls the builders and builds the definition in CICS DSA, and updates the CICS global catalog data set for future recovery.

EXEC CICS CREATE builds the same record that would be obtained from the CSD and then calls the builders just like CEDA INSTALL.

EXEC CICS DISCARD calls the builders with a pointer to the TCTTE entry that is to be deleted. The builders then freemain the TCTTE, remove index entries and the catalog record.

Modules

DFHZCQ handles all requests for the dynamic add and delete of terminal control resources. It contains the following CSECTs:

```
DFHBSIB3    DFHBSSZM    DFHBSTP3    DFHBSTZ1
DFHBSIZ1    DFHBSSZP    DFHBSTS     DFHBSTZ2
DFHBSIZ3    DFHBSSZR    DFHBSTT     DFHBSTZ3
DFHBSMIR    DFHBSSZS    DFHBSTZ     DFHBSXGS
DFHBSMPP    DFHBSSZ6    DFHBSTZA    DFHBSZZ
DFHBSM61    DFHBST      DFHBSTZB    DFHBSZZS
DFHBSM62    DFHBSTB     DFHBSTZC    DFHBSZZV
DFHBSS      DFHBSTBL    DFHBSTZE    DFHZCQCH
DFHBSSA     DFHBSTB3    DFHBSTZH    DFHZCQDL
DFHBSSF     DFHBSTC     DFHBSTZL    DFHZCQIN
DFHBSSS     DFHBSTD     DFHBSTZO    DFHZCQIQ
DFHBSSZ     DFHBSTE     DFHBSTZP    DFHZCQIS
DFHBSSZB    DFHBSTH     DFHBSTZR    DFHZCQIT
DFHBSSZG    DFHBSTI     DFHBSTZS    DFHZCQRS
DFHBSSZI    DFHBSTM     DFHBSTZV    DFHZCQRT
DFHBSSZL    DFHBSTO     DFHBSTZZ    DFHZCQ00
```

Note: The term “node” refers either to a TCTTE or to one of its subsidiary parts, such as the NIB descriptor.

Subroutines that are found in the builders:

BSEBUILD
BUILD: Create the node. For example, obtain the shared storage for the node.

BSECON
CONNECT: Connect the higher node to the lower. For example, make the TCTTE point to the NIB descriptor.

BSEDESTR
DESTROY: Abolish a deleted node. For example, free the storage removed from TMP's chains.

BSEFINDF
FINDFIRST: Find the first subsidiary node of a higher node. For example, BSFINDF(TCTTE) returns the NIBD being built.

BSEFINDN
FINDNEXT: Find the next subsidiary node of the one just found. For example, return the address of the next model TCTTE.

BSEFLAT
FLATTEN: Build the catalog or log record segment for each part of the TCTTE. This is passed back to the caller to create a complete "flattened" TCTTE.

BSEMAKEY
MAKEKEY: Create a key that is used to write out the new node to the global catalog.

BSENQIRE
ENQUIRE: The converse of BUILD, it creates a BPS from a TCTTE. The BPS can then be shipped to another system.

BSEREADY
READY: Make a node ready to use. For example, add to TMP's chains.

BSERESET
RESET: Build the TCTTE from the CICS global catalog. (RESET is a cut-down version of UNFLATTEN.)

BSEUNFLA
UNFLATTEN: Build the TCTTE from the CICS global catalog.

BSEUNRDY
UNREADY: Check that a node can be deleted. For example, ensure that no AIDs are queued on a TCTTE before deleting.

Not all subroutines are found in all builders. Certain subroutines are required, but do nothing other than return to the caller. The subroutine names are the same in each builder.

Module entry

Consider a module entry to be a router that does some housekeeping and then branches to the appropriate subroutine:

- Enter the builder at offset X'18'.
- The first X'17' bytes are taken up by the standard DFHVM macro expansion.
- Save DFHTBS's registers (DFHTBS calls each builder).
- Save the first two entries in the parameter list:
  1. The address of LIFO storage
  2. The index number of the subroutine to call.
- Increase the value of register 1 by 8 to get past the first two entries.
- Branch to the appropriate subroutine of the builder using the index number passed.
- Return from the builder subroutine.
- Restore registers.

- Return to DFHTBS.

Subroutine entry

- Register 1 points to the parameter list.
- Store Register 14 (return address) at Register 2 + X'nn' (varies by entry point).
- Store the parameter list into Register 2 + X'nn' (varies by entry point).
- The length of the parameter list varies.

Subroutine exit (return to module entry)

- Exit from the subroutine only through an "official" exit point.
- The exit point is usually the end of the subroutine.
- The end of the subroutine is indicated with "*end; /*BUILD */".
- In some cases, the end of the subroutine branches back to the exit point somewhere within the subroutine.
- Return (BR R14) from within the subroutine.
- Reload Register 14 from Register 2 + X'nn' and return to caller.

Patterns

In DFHZCQRT, a series of patterns define the flow through the builder modules. (See Figure 30.) For each kind of terminal, there is a different pattern.

If installing, DFHZCQIS selects the pattern and calls DFHTBS (table builder service). If deleting, DFHZCQDL does the selection.

DFHTBS interprets the pattern and calls each builder that the pattern calls out. DFHTBS knows nothing about the terminal or whether you are installing or deleting. It does what the pattern tells it to do.

DFHTBS passes a BPS as it calls each builder. The BPS allows one builder to be used for many different kinds of terminals. For example, DFHBSTC obtains the user area for all terminal types. The BPS contains the length to be obtained.

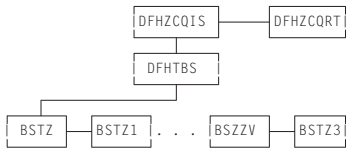


Figure 30. Calling sequence of builders (determined by patterns)

Calling sequence of builders for a 3277 remote terminal

1. DFHZCQRT contains a series of comments followed by the patterns. The comment appears as:

   ```
   /* * * * * * * * * * * * * */
   /*      3277 REMOTE      */
   /* * * * * * * * * * * * * */
   ```

2. Shortly afterward is a Declare (DCL) followed by a level-1 name:

   ```
   DCL  1 P145002  STATIC
   ```

   This is the name of the pattern that drives the build process for a 3277 remote terminal.

   - DFHBSTZ is indicated to be the first builder called.
   - One pattern is used to drive the building process.

- 18 subpatterns are to be used.
- Three of these 18 subpatterns each call one additional pattern.
- The terms “pattern” and “builder” mean the same thing. Therefore:

```
DFHBSTZ  +  DFHBSxx  +  DFHBSxx  =  22
  (1)    +    (18)   +    (3)    =  22
 pattern +    sub-   +  sub-sub- =  22
             patterns   patterns
```

  Thus we have to go through 22 builder modules to build a 3277 remote terminal.

3. Go to the cross-reference at the back of the dump and find where P145002 is defined in assembler language. Go to that address.
4. This states that the first builder to be called is DFHBSTZ. This is the main one.
5. Drop down to the 2-byte fields that follow: these state the names of the builders that are to be called, in sequence (18 should be listed).
6. The first one is IAATZ1 which does not sound familiar:
   - Go to the cross-reference at the back of the dump, look up IAATZ1, and go to where it is defined.
   - You see that this is DFHBSTZ1.
   - You can also see a close resemblance between IAATZ1 and DFHBSTZ1, but do not count on this to be always true.
7. Now you know that the second builder to be called is DFHBSTZ1.
8. The next two builders to be called are IAATCV (DFHBSTV) and IAATCB (DFHBSTB).
9. The fifth builder to be called according to the pattern needs to be looked at:
   - The pattern says that IACTZ3 should be called.
   - When you go to where IACTZ3 is defined, you find that this is DFHBSIZ3.
   - You also see that DFHBSIZ3 calls IAATM.
   - Look up IAATM and you see that it is DFHBSTM.
   - This is a sub to a subpattern, and this is how nesting of builder calls occurs.
   - Thus, DFHBSIZ3 calls DFHBSTM when building a local 3277.
   - DFHBSTM accounts for one of the “other” three mentioned in step 2.
10. If you continue through this pattern, you can identify the names of the 22 builders that would be called to build a 3270 local TCTTE.

    Here is the complete list, in order, of the builders that are called:

```
1  DFHBSTZ         12 DFHBSTH
2  DFHBSTZ1        13 DFHBSTI
3  DFHBSTZV        14 DFHBSTS
4  DFHBSTZB        15 DFHBSTT
5  DFHBSIZ3        16 DFHBSTZA
6  DFHBSTM         17 DFHBSTP3
7  DFHBSTB         18 DFHBSZZ
8  DFHBSIB3        19 DFHBSTB3
9  DFHBSTO         20 DFHBSTZE
10 DFHBSTC         21 DFHBSZZV
11 DFHBSTE         22 DFHBSTZ3
```

    A look at “Pattern Trace” supports this flow. Note that the first ZCP TBSB(P) BUILD and its matching return (the return has no builder suffix) should be ignored.

Builder parameter list

As each builder is called by DFHTBS, a parameter list is passed. Unique data is passed to enable one builder module to be called for a variety of terminal types. The length of the builder parameter list is fixed for each kind of subroutine; for example, the parameter list passed to BSEBUILD is always X'23' bytes long, regardless of the builder involved.

Subroutine	Length of parameter list (hexadecimal)
BSEBUILD	23
BSECON	13
BSEDESTR	7
BSEMAKEY	B
BSEREADY	3
BSEUNRDY	17
BSEFINDF	F
BSEFINDN	B
BSEFLAT	B
BSEUNFLA	27
BSENQIRE	7

When the builders are called

Builders are called during:

- Cold start
- Warm start
- Emergency restart
- After emergency restart
- Autoinstall logon and logoff
- APPC autoinstall
- CEDA INSTALL
- EXEC CICS CREATE
- EXEC CICS DISCARD
- Transaction routing
- Non-immediate shutdown.

Cold start:

- Read information from the CSD and call builders to build RDO-defined terminals.
- Load in DFHTCT for non-z/OS Communications Server terminals. Builders are not called.

Warm start:

Note: A warm start is identical to an emergency restart from the builders perspective. The only difference is that Recovery Manager has no forward-recovery records to pass to DFHAPRDR.

- Read information from the global catalog and call builders to restore RDO-defined terminals.
- Load in DFHTCT for non-z/OS Communications Server terminals. Builders are not called.

Emergency restart:

- Read information from the global catalog and call builders to restore RDO-defined terminals.

Note: Auto-installed terminals will not have a catalog entry if AIRDELAY=0

- Recovery Manager calls DFHAPRDR which calls the builders to restore in-flight terminals installs from the system log.
- Load in DFHTCT for non-z/OS Communications Server resources. Builders are not called.

After emergency restart: Delete autoinstalled terminals after the time period has expired as specified in the AIRDELAY parameter (if the user has not logged back on before then).

APPC autoinstall:

- Inquire on the model supplied by the autoinstall user program
- Install an APPC connection created from this inquiry.

Autoinstall logon and logoff:

- Logon: Install terminal entry using model entry in the AMT.
- Logoff: Delete terminal entry.

CEDA INSTALL: Install z/OS Communications Server terminal resources. (There is no builder process for CEDA DEFINE or ALTER.)

EXEC CICS CREATE: Install z/OS Communications Server terminal resources.

EXEC CICS DISCARD: Delete z/OS Communications Server terminal resources.

Transaction routing: If a TCTTE is defined as shippable, its definition is shipped to the remote system and installed there. The definition is obtained by an INQUIRE call to the builders in the Terminal Owning Region and built with an INSTALL call in the Application Owning Region.

Shutdown: Delete autoinstalled terminals from the catalog (if they had entries, and are not LU6.2 parallel connections). On a warm start, therefore, autoinstalled terminals are not recovered.

Diagnosing problems with the builders

When working on a problem associated with a builder (for example, abend or loop), it may be helpful to ask yourself the following questions:

- Why am I in a DFHBS* module? Am I doing CEDA GRPLIST install, CEDA GROUP install, autoinstall, logon, logoff, catalog, uncatalog, create or discard?
- What is the termid/sysid of the terminal I am working with (the one I am installing, deleting, cataloging or uncataloging)?
- Is this resource part of an resource definition atom?
- How is this terminal defined?
- Are there any messages associated with this terminal?

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for the DFHZCQxx modules:

- AP FCB0 - FCBF, for which the trace level is 1.

The following point IDs are provided for the DFHTBSx modules:

- AP FCC0 - FCC9, for which the trace level is 1.

The following point IDs are provided for the DFHTBSxP modules:

- AP 0630 - 0644, exception trace.
- AP FCD0 - FCD9, for which the trace level is 1.
- AP FCDA - FCDB, for which the trace level is 2.

The following point IDs are provided for the DFHTBSS module:

- AP 0620 - 0621, for which the trace level is 1.
- AP 0622 - 062E, and 0645 exception trace.

The following point IDs are provided for the DFHTONR module:

- AP 0648 - 0649, for which the trace level is 1.
- AP 064A - 064C, exception trace.

The following point IDs are provided for the DFHAPRDR module:

- AP 0601 - 0602, for which the trace level is 1.
- AP 0603 - 061E, exception trace.

The following point IDs are provided for the DFHZGTA module:

- AP FA80 - FA81, for which the trace level is 1.
- AP FA82 - FA9A, exception trace.

The following point ID is provided for message set production:

- AP FCDD, exception trace.

The following point ID is provided for DFHBSTZA:

- AP FCDE, exception trace.

Messages

Builder modules issue messages in the DFHZC59xx, DFHZC62xx, and DFHZC63xx series.

Message sets

If a builder finds an error, it adds a message to a message set. This set is then printed by the caller; for example:

```
DFHTCRP   Cold start (local system entry
                     and error console only)
DFHAMTP   CEDA, EXEC CICS CREATE
DFHEIQSC  EXEC CICS DISCARD CONNECTION
DFHEIQST  EXEC CICS DISCARD TERMINAL
DFHZATA   Autoinstall
DFHZATD   Autoinstall delete
DFHZATS   Install and delete transaction routed terminals
```

How messages show up in a trace

If a message is issued from a builder module (that is, those messages with a prefix of DFHZC59xx, DFHZC62xx, or DFHZC63xx), it appears in the trace as a table builder services message trace entry with the following point ID:

- AP FCDD, exception trace.

This trace entry is produced when a message is added to the message set and indicates there was a problem in building or deleting a terminal or connection.

Built-in functions

CICS provides the application programmer with two commonly used functions: field edit and phonetic conversion.

These are functions that generally used to be coded as separate subroutines by the programmer. They are referred to as built-in functions.

The field edit function is provided by the BIF DEEDIT command of the CICS application programming interface.

The phonetic conversion function is provided as a subroutine that can be called by CICS application programs, and also by any offline programs.

Design overview

The built-in functions component includes field edit and phonetic conversion, both of which are available to a CICS application program. Also, the phonetic conversion subroutine can be used offline.

Field edit (DEEDIT)

The field edit function allows the application program to pass a field containing EBCDIC digits (0 through 9) intermixed with other values, and receive a result with all non-numeric characters removed.

For further details of this function, see BIF DEEDIT.

Phonetic conversion

This facility allows the user to organize a file according to name (or similar alphabetic key), and access the file using search arguments that may be misspelled.

The phonetic conversion subroutine (DFHPHN) converts a name into a partial key, which can then be used to access a database file. The generated key is based upon the sound of the name. This means that names sounding similar, but spelled differently, generally produce identical keys. For example, the names SMITH, SMYTH, and SMYTHE all produce a phonetic key of S530. Likewise, the names ANDERSON, ANDRESEN, and ANDRESENN produce a phonetic key of A536. The encoding routine ignores embedded blanks in a name, so you can write names prefixed by 'Mc' with or without a blank between the prefix and the rest of the name, for example, 'McEWEN' or 'Mc EWEN'.

For details of how to code a CALL statement for the DFHPHN subroutine according to the language of the application program, see the Developing CICSPlex SM applications.

Modules

Module	Description
DFHEBF	EXEC interface processor for BIF DEEDIT command
DFHPHN	Phonetic conversion subroutine

Exits

No global user exit points are provided for these functions.

Trace

No tracing is performed for the phonetic conversion subroutine.

The following point ID is provided for DFHEBF:

- AP 00FB, for which the trace level is BF 1.

CICS-DB2 Attachment Facility

The CICS-DB2® Attachment facility allows applications programs to access and update data held in DB2 tables managed by the DB2 for z/OS product. It also allows applications to send operator commands to a DB2 subsystem.

Design overview

The CICS-DB2 Attachment facility allows connection to a DB2 subsystem using the CICS resource manager interface (RMI), which is also known as the task-related user exit interface.

The attachment facility interfaces to DB2 through a series of requests to three components of DB2, each of which processes specific types of requests:

- Subsystem Support Subcomponent (SSSC) for thread and system control requests
- Advanced Database Management Facility (ADMF) for SQL requests
- Instrumentation Facility Component (IFC) for IFI requests

There no are DB2 release dependencies in the attachment facility; it can connect to a DB2 subsystem running any supported level of DB2.

The architecture of the CICS-DB2 interface is described in Overview of the CICS DB2 interface. The attachment facility exploits the open transaction environment (OTE) and uses CICS-managed open TCBs.

CICS Initialization

During CICS Initialization the following modules are invoked:

CICS-DB2 initialization gate DFHD2IN1: DFHD2IN1 first receives control from DFHSII1 duiring CICS initialization by means of a DFHROINM INITIALISE call. When invoked with this function DFHD2IN1 attaches a system task CSSY to run program DFHD2IN2.

DFHD2IN1 is invoked a second time later by DFHSII1 by means of a DFHROINM WAIT_FOR_INITIALIZATION call for which DFHD2IN1 issues a CICS wait to wait for DFHD2IN2 processing to complete.

CICS-DB2 restart program DFHD2RP: DFHD2RP runs under system task CSSY during CICS initialization. DFHD2RP performs the following functions:

- Adds storage manager subpools for the DFHD2CSB, DFHD2ENT, DFHD2PKG, and DFHD2TRN control blocks.
- Issues lock manager domain ADD_LOCK requests to add the necessary locks required by the CICS-DB2 Attachment facility to manage the dynamic chains of DFHD2LOT and DFHD2CSB control blocks, plus locks to manipulate the DFHD2GLB, DFHD2ENT and DFHD2TRN control blocks.

- Loads CICS-DB2 modules DFHD2CC, DFHD2CO, DFHD2D2, DFHD2RL, DFHD2STR, DFHD2STP and DFHD2TM
- Activates the DFHD2TM and DFHD2RL gates with the kernel.
- Registers the DFHD2RL gate with RL domain for call-backs for PACKAGESET bundle parts.
- Adds D2AC directory.
- For cold and Initial CICS starts:
  - Purges the Global catalog of DFHD2GLB, DFHD2ENT and DFHD2TRN control blocks
- For warm and emergency CICS starts:
  - Installs DFHD2GLB, DFHD2ENT and DFHD2TRN blocks found on the global catalog

CICS-DB2 Attachment startup

The CICS-DB2 Attachment facility can be started using one of the following methods:

- Specifying program DFHD2CM0 in PLTPI
- Specifying SIT parameter **DB2CONN=YES**
- Issuing the DSNC STRT command
- Issuing the CEMT or EXEC CICS **SET DB2CONN CONNECTED** command

All of these methods result in an **EXEC CICS SET DB2CONN CONNECTED** command being issued and the CICS-DB2 startup program DFHD2STR getting control.

CICS-DB2 startup program DFHD2STR: The startup program starts by reading a temporary storage queue to obtain any parameters passed if a DSNC STRT command has been issued. It also retrieves any parameters specified via the INITPARM SIT parameter by linking to program DFHD2INI.

Next DFHD2STR must ensure the necessary DFHD2GLB block is installed. If a DFHD2GLB is already installed, representing an installed DB2CONN, then it is checked to make sure interface is currently shut before startup can proceed.

The remainder of DFHD2STR processing is as follows:

- Initialise the DFHD2GLB and set the state to 'connecting'
- MVS load the DB2 program request handler
- Attach a CICS system task to run the CICS DB2 service task CEX2
- Call DFHD2CO to connect to DB2 and obtain indoubts
- Enable the CICS-DB2 TRUE DFHD2EX1
- If connected to DB2 for OS/390 Version 5 or earlier, then issue an MVS Attach for the CICS-DB2 master subtask program DFHD2MSB and wait for DFHD2MSB initialization processing to complete
- Set the status of the connection to 'connected'
- Post CEX2 to process any indoubts passed from DB2
- Update state in the temporary storage queue to pass back to a DSNC STRT command

CICS-DB2 attachment shutdown

The CICS-DB2 Attachment facility can be stopped using one of the following methods:

- Issuing the DSNC STOP command

- Issuing the CEMT or EXEC CICS **SET DB2CONN NOT CONNECTED** command
- Running the CDBQ or CDBF transactions
- Shutting down CICS

All these methods result in an **EXEC CICS SET DB2CONN NOTCONNECTED** command being issued and the CICS-DB2 shutdown program DFHD2STP getting control.

CICS-DB2 shutdown program DFHD2STP: Processing in DFHD2STP is as follows:

- If CDB2SHUT is set in the dump table, take a system dump (serviceability aid)
- If a CDB2SHUT dump has not been taken, and the CICS-DB2 master subtask program DFHD2MSB has unexpectedly abended, then a system dump is taken with a dump code of MSBABEND.
- Post CICS-DB2 service task CEX2 to end all subtasks, then terminate itself. Wait for service task to complete.
- If present, post master subtask DFHD2MSB to terminate. Wait for it to terminate, then detach master subtask TCB.
- Call DFHD2CO to disconnect from DB2.
- Call DFHD2CC to write out shutdown statistics.
- If the CICS-DB2 attachment is to go into 'standbymode':
  - Re-initialize DFHD2GLB and set the state to 'connecting'.
  - Post any tasks who are waiting for shutdown to complete.
  - Issues 'Waiting for DB2 attach' message
- If the CICS-DB2 attachment is not to go into 'standbymode':
  - Disable the CICS-DB2 TRUE DFHD2EX1.
  - MVS delete the program request handler.
  - Re-initialize the DFHD2GLB, set the state to 'shut'.
  - Issue the shutdown complete message and post any tasks who are waiting for shutdown to complete.

CICS-DB2 mainline processing

CICS-DB2 task-related user exit (TRUE) DFHD2EX1:

Control is passed to the TRUE through the CICS RMI. The TRUE manages the relationship between a CICS task (represented by an LOT control block), and a CICS-DB2 thread (represented by a CSB control block). DFHD2EX1 uses parameters set in the DB2CONN and DB2ENTRY definitions to manage use of the CICS DB2 threads, each thread running under a thread TCB.

- When connected to DB2 for OS/390 Version 5 or earlier, the thread TCB is a subtask managed by the CICS DB2 attachment facility. The subtask that is running program DFHD2EX3 issues requests to DB2 on behalf of a CICS task. A wait/post protocol is executed between the CICS task that is running in the CICS-DB2 TRUE, and the subtask in program DFHD2EX3.
- When connected to DB2 for z/OS Version 6, 7, or 8, the thread TCB is a CICS open TCB (L8 mode). Program DFHD2D2 is called under the open TCB, and issues the requests to DB2. In this case, both DFHD2EX1 and DFHD2D2 run under the L8 TCB.
- When connected to DB2 for z/OS Version 9 or later with APAR PM57744 applied, the thread TCB is any eligible key 8 CICS open TCB: L8, T8, or X8, and DFHD2EX1 and DFHD2D2 both run under this TCB.

The CICS-DB2 TRUE DFHD2EX1 gets invoked by the RMI for the following events:

- EXEC SQL commands and DB2 IFI commands from application programs
- syncpoint
- end of task
- INQUIRE EXITPROGRAM commands for the DB2 TRUE with the CONNECTST or QUALIFIER keywords (RMI SPI calls)
- EDF when using EDF for EXEC SQL commands
- CICS shutdown
- Switch application environment

CICS-DB2 coordinator program DFHD2CO: The coordinator program runs under the CICS Resource owning (RO) TCB, and handles the overall connection between CICS and a DB2 subsystem. It is called :

- by DFHD2STR during startup of the attachment facility to issue the coordinator identify to DB2, that is to establish connection to DB2. Once established, it passes DB2 an ECB to be posted should DB2 terminate, and it also obtains from DB2 a list of units of work (UOWs) that DB2 is indoubt about. This list is anchored off the CICS-DB2 global block (DFHD2GLB) for processing later in startup.
- by DFHD2STP during shutdown of the attachment facility to terminate the identify to DB2 and so disconnect.
- by the CICS-DB2 TRUE DFHD2EX1 during resync processing to pass the resolution of a indoubt unit of work to DB2. Indoubt resolution has to be done under the same TCB that issued the coordinator identify to DB2.

CICS-DB2 master subtask program DFHD2MSB: When operating with DB2 for OS/390 Version 5 or earlier, the DFHD2MSB TCB is attached by DFHD2STR during startup of the Attachment facility. It runs as a 'daughter' of the main CICS TCB. It is 'mother' to all the subtask TCBs which process the DB2 work. The DFHD2MSB TCB is detached by DFHD2STP during CICS-DB2 Attachment shutdown.

The main functions of DFHD2MSB are:

- To attach thread subtasks as required
- To detach thread subtasks as required
- To provide a recovery routine to clean up if a thread subtask fails

CICS-DB2 subtask program DFHD2EX3: When operating with DB2 for OS/390 Version 5 or earlier, a CICS-DB2 subtask TCB is attached by DFHD2MSB when required by DFHD2EX1. It runs as a daughter of the DFHD2MSB TCB and a granddaughter of the main CICS TCB. A CICS-DB2 subtask TCB normally remains active for the lifetime of the CICS Attachment facility and terminates as part of CICS-DB2 Attachment facility shutdown. Exception conditions that cause a subtask TCB to be detached are:

- if the DB2CONN TCBLIMIT parameter is lowered
- if a CICS task is forcepurged whilst its associated subtask is active in DB2
- If a failure occurs during syncpoint processing during the indoubt window requiring the thread to be released.

The DFHD2EX3 program issues requests to DB2 using the DB2 SSSC, ADMF and IFC interfaces communicating via the DB2 program request handler DSNAPRH. In

order to process DB2 requests a TCB first has to IDENTIFY to DB2, secondly it has to SIGNON to DB2 to establish authorization ids to DB2. Thirdly a thread has to be created. Once a thread has been created API and syncpoint requests can flow to DB2. Subsequent SIGNON requests can occur for a thread to change authorization ids to DB2 or for the purposes of DB2 cutting accounting records (partial SIGNON) When a thread is nolonger required it is terminated. The TCB remains identified and signed on to DB2 and awaits another request requiring it to create a thread again.

Each DB2 subtask runs an instance of program DFHD2EX3 and each is represented by a DFHD2CSB control block. A CSB control block is anchored to one of three CSB chains depending on its state (an active thread within a UOW, a thread waiting for work, or an identified, signed on TCB with no thread). The CICS-DB2 TRUE DFHD2EX1 manages the CSB chains.

CICS-DB2 thread processor DFHD2D2: The thread processor program DFHD2D2 is used only when operating with DB2 for z/OS Version 6 and later, when the CICS-DB2 Attachment Facility uses CICS open TCBs rather than privately managed subtask TCBs. In the Open Transaction environment (OTE), when connected to DB2 V9 or later, the CICS-DB2 TRUE will be enabled as CONCURRENCY(REQUIRED) API(CICSAPI) and run on any eligible key 8 open TCB. When connected to DB2 V8 or earlier, it is enabled as CONCURRENCY(REQUIRED) API(OPENAPI) and will run on an L8 TCB.

Instead of posting a subtask, DFHD2EX1 calls DFHD2D2. DFHD2D2 performs the same functions as performed by subtask program DFHD2EX3 in a non OTE environment; that is, issuing the identify, signon, create thread, terminate thread calls to DB2, plus the api and syncpoint calls to DB2.

DFHD2D2 is called through a subroutine domain call on which the address of the relevant connection control block (DFHD2CSB) is passed. On the first call of a unit of work, DB2 is called to “associate” the connection with the calling TCB. After this is done, calls to DB2 can proceed as normal. When a DB2 thread is released from a CICS transaction (typically at syncpoint), the connection is “dissociated” from the open TCB. Hence a connection control block (DFHD2CSB) has an affinity to an open TCB while it is associated. With DB2 for OS/390 Version 5 and earlier, a connection has a permanent affinity to its subtask TCB.

CICS-DB2 service task program DFHD2EX2: The CICS-DB2 service task program DFHD2EX2 runs as a CICS system task under transaction CEX2. Its mains functions are:

- To wait for DB2 to startup if DB2 is down when connection is attempted if STANDBYMODE=RECONNECT or CONNECT is specified in the DB2CONN.
- To initiate shutdown of the CICS-DB2 Attachment facility if posted to do so.
- To perform the protected thread purge cycle.
- To issue EXEC CICS RESYNC to process DB2 indoubts.
- For DB2 for OS/390 Version 5 or earlier, to terminate all subtasks during CICS-DB2 Attachment facility shutdown.

CICS-DB2 PLTPI program DFHD2CM0: Used in PLTPI or as a result of DB2CONN=YES being set in the SIT. It issues an EXEC CICS SET DB2CONN CONNECTED command to start the CICS DB2 Attachment facility.

CICS-DB2 comand processor DFHD2CM1: DFHD2CM1 processes commands issues via the DSNC command. The following commands are processed:

- DSNC STRT - EXEC CICS SET DB2CONN CONNECTED command issued
- DSNC STOP - EXEC CICS SET DB2CONN NOTCONNECTED command issued
- DSNC MODIFY DEST - EXEC CICS SET DB2CONN MSGQUEUEn command issued
- DSNC MODIFY TRAN - EXEC CICS SET DB2CONN THREADLIMIT or EXEC CICS SET DB2ENTRY THREADLIMIT command issued.
- DSNC DISC - call passed to DFHD2CC to disconnect threads
- DSNC DISP PLAN - call passed to DFHD2CC to display information on threads for a particular DB2 plan
- DSNC DISP TRAN - call passed to DFHD2CC to display information on threads for a transaction.
- DSNC DISP STAT - call passed to DFHD2CC to write out statistics
- DSNC -db2command - DB2 IFI ccommand issued to send operator command to the connected DB2 subsystem.

CICS-DB2 shutdown quiesce program DFHD2CM2: Runs under transaction CDBQ. Issues an EXEC CICS SET DB2CONN NOTCONNECTED WAIT command to shutdown the CICS-DB2 Attachment facility.

CICS-DB2 shutdown force program DFHD2CM3: Runs under transaction CDBF. Issues an EXEC CICS SET DB2CONN NOTCONNECTED FORCE command to shutdown the CICS-DB2 Attachment facility.

CICS-DB2 table manager DFHD2TM: Handles installs, discards, inquire and set requests for the DFHD2GLB, DFHD2ENT and DFHD2TRN control blocks representing the DB2CONN, DB2ENTRY and DB2TRAN resources. In addition, it handles inquire requests for DB2 Package Sets. Callers of DFHD2TM are as follows:

- DFHAMD2 - for CEDA install and EXEC CICS CREATE
- DFHD2EX1 - to complete disablement of a DB2ENTRY or to complete Attachment facility shutdown
- DFHD2RP - to install objects from the Global Catalog during CICS restart
- DFHEIQD2 - for EXEC CICS INQUIRE,SET and DISCARD of DB2 objects
- DFHESE - for inquiry during EXEC CICS QUERY SECURITY processing.
- EYU0NLG4 and EYU0NLX2 - for inquiring on DB2 Package Sets during CPSM resource topology processing.
- EYU0NQDP - for inquiring on DB2 Package Sets for CPSM DB2PKGST requests.

CICS DB2 statistics program DFHD2ST: Called by AP domain statistics program DFHAPST to process CICS-DB2 statistics for EXEC CICS COLLECT STATISTICS and EXEC CICS PERFORM STATISTICS commands.

CICS DB2 connection control program DFHD2CC: DFHD2CC proceses the following requests:

- Start_db2_attachment - request routed on to DFHD2STR
- Stop_db2_attachment - request routed on to DFHD2STP
- Write_db2_statistics - statistics collected from control blocks and are written out to the terminal, to transient data or to SMF.
- disconnect_threads - CSB control blocks searched and marked so that threads are terminated when they are next released.

- display_plan and display_tran - thread information collected from control blocks and output to the terminal.

CICS DB2 EDF processor DFHD2EDF: Receives control from CICS-DB2 TRUE DFHD2EX1 when the TRUE is invoked for an EDF request. DFHD2EDF uses the RMI provided parameters to format the screen to be output by EDF before and after an EXEC SQL request is issued.

Control blocks

DFHD2SS (CICS-DB2 static storage)

CICS-DB2 static storage (D2SS) is acquired by DFHSIB1 and anchored off field SSZDB2 in the static storage address list DFHSSADS. The static storage is initialized by the CICS-DB2 restart program DFHD2RP. Its lifetime is that of the CICS region. CICS-DB2 static storage holds information such as storage manager, lock manager and directory manager tokens acquired during restart processing before any other CICS-DB2 control blocks are installed.

DFHD2GLB (CICS-DB2 global block)

The DFHD2GLB block represents an installed DB2CONN definition. DFHD2TM obtains it by using a getmain request when a DB2CONN is installed, and releases it by using a freemain request when a DB2CONN is discarded. It holds CICS-DB2 state data global to the connection and also the state data for pool threads and commands threads. The pool and command sections of the DFHD2GLB are mapped by a common type definition DFHD2RCT, which is also used to map the DFHD2ENT control block.

The DFHD2GLB block is anchored off CICS-DB2 static storage in field D2S_DFHD2GLB.

DFHD2ENT (CICS-DB2 DB2ENTRY block)

The DFHD2ENT block represents an installed DB2ENTRY definition. DFHD2TM obtains it by using a getmain request when a DB2ENTRY is installed and releases it by using a freemain request when a DB2ENTRY is discarded.

It uses a type definition DFHD2RCT, in common with the pool and command sections of the DFHD2GLB block, to achieve a common layout for all three areas. A DFHD2ENT block is located using a directory manager index that is keyed off the RDO name of the DB2ENTRY.

DFHD2TRN (CICS-DB2 DB2TRAN block)

The DFHD2TRN block represents an installed DB2TRAN definition. DFHD2TM obtains it by using a getmain request when a DB2TRAN is installed, and releases it by using a freemain request when a DB2TRAN is discarded. A DB2TRAN can be located in two ways:

- By a directory manager index keyed off the RDO name of the DB2TRAN
- By a directory manager index keyed off the transaction id associated with the DB2TRAN

DFHD2CSB (CICS-DB2 connection block)

The DFHD2CSB block represents a CICS-DB2 connection, with or without a thread. A DFHD2CSB is created by DFHD2EX1 prior being passed to DFHD2EX3 or DFHD2D2. A DFHD2CSB is freed by DFHD2EX1 after the DFHD2EX3 program has returned to MVS, or when DFHD2D2 indicates it should be freed. A DFHD2EX3 block is anchored off one of several CSB chains from a DB2ENTRY or the DFHD2GLB depending on the state of the connection and the DB2 thread.

DFHD2GWA (CICS-DB2 global work area)

The DFHD2GWA block is the global work area of the CICS-DB2 task-related user exit (TRUE) DFHD2EX1.

The DFHD2GWA block is obtained by using a getmain request when the TRUE is enabled, and released by using a freemain request when the TRUE is disabled. The D2GWA holds a chain of LOT control blocks representing the tasks currently using the CICS-DB2 interface.

DFHD2LOT (CICS-DB2 life of task block)

The DFHD2LOT block is the task local work area of the CICS-DB2 task-related user exit (TRUE) DFHD2EX1.

DFHERM obtains the DFHD2LOT block by using a getmain request when a task first calls the CICS-DB2 TRUE, and releases it by using a freemain request at end of task. Its address is passed to DFHD2EX1 by DFHERM in parameter UEPTAA in the DFHUEPAR RMI parameter list.

The DFHD2LOT holds CICS-DB2 state information for a CICS task using the CICS-DB2 interface.

DFHD2PKGSET (CICS-DB2 packageset block)

The DFHD2PKGSET block represents an installed packageset definition for a cloud application or for a platform. It is installed by DFHD2RL using a bundle definition.

A DFHD2PKGSET is located by a directory manager index that is keyed off the application context of platform name, application name, application major version, application minor version and application micro version. For a platform packageset the application name and version fields in the key are null.

Modules

Module	Description
DFHD2CC	CICS-DB2 connection control program
DFHD2CO	CICS-DB2 coordinator program
DFHD2CM0	CICS-DB2 PLTPI startup program
DFHD2CM1	CICS-DB2 command processor
DFHD2CM2	CICS-DB2 quiesce shutdown program
DFHD2CM3	CICS-DB2 force shutdown program
DFHD2D2	CICS-DB2 thread processor
DFHD2EDF	CICS-DB2 EDF processor
DFHD2EX1	CICS-DB2 task related user exit (TRUE)
DFHD2EX2	CICS-DB2 service task program
DFHD2EX3	CICS-DB2 subtask program
DFHD2INI	CICS-DB2 Initparm processor
DFHD2IN1	CICS-DB2 initialization gate
DFHD2IN2	CICS-DB2 recovery task
DFHD2MSB	CICS-DB2 master subtask program
DFHD2RP	CICS-DB2 restart program
DFHD2STP	CICS-DB2 shutdown program
DFHD2STR	CICS-DB2 startup program
DFHD2ST	CICS-DB2 statistics program
DFHD2TM	CICS-DB2 table manager
DSNCUEXT	CICS-DB2 sample dynamic plan exit

Exits

There are no global user exits provided by the CICS DB2 Interface.

The CICS DB2 interface does however provide a dynamic plan 'exit' in the form of a user-replaceable program. A sample default exit is provided called DSNCUEXT. A dynamic plan exit allows the name of the plan to chosen dynamically at execution time. For further information about dynamic plan exits, see the Overview of the CICS DB2 interface.

Trace

The CICS-DB2 Attachment facility outputs trace entries in the range AP 3100 to AP 33FF. Trace output from the CICS-DB2 TRUE (DFHD2EX1) and the thread processor (DFHD2D2), and GTF trace from the CICS-DB2 subtask is controlled by the RI (RMI) trace flag. Trace from the rest of the attachment and other CICS-DB2 modules is controlled by the RA (Resource Manager Adapter) trace flag.

Statistics

A limited set of CICS-DB2 statistics can be obtained by issuing the DSNC DISP STAT command, which will output the statistics to a CICS terminal. The same format of statistics is output to a nominated transient data queue when the CICS-DB2 Attachment facility is shut down For more information see the Overview of the CICS DB2 interface.

A more comprehensive set of CICS-DB2 statistics can be obtained by issuing an EXEC CICS PERFORM STATISTICS RECORD command with the DB2 keyword, or by issuing the EXEC CICS COLLECT STATISTICS command with the DB2CONN or DB2ENTRY keywords. CICS-DB2 Global statistics are mapped by DSECT DFHD2GDS. CICS-DB2 resource statistics are mapped by DSECT DFHD2RDS. For more information see the CICS statistics in Monitoring.

CICS Web support and the CICS business logic interface

CICS Web support is a collection of CICS services that enable a CICS region to act both as an HTTP server, and as an HTTP client. When CICS is an HTTP server, Web clients can use transaction processing services by calling CICS programs or by running CICS transactions. When CICS is an HTTP client, a user application program in CICS can initiate a request to an HTTP server, and receive a response from it. Web clients use TCP/IP to communicate with CICS Web support.

The CICS business logic interface allows other external users to use transaction processing services.

Control blocks

Figure 31 on page 1529 shows the control blocks used by CICS Web support for 3270 display applications.

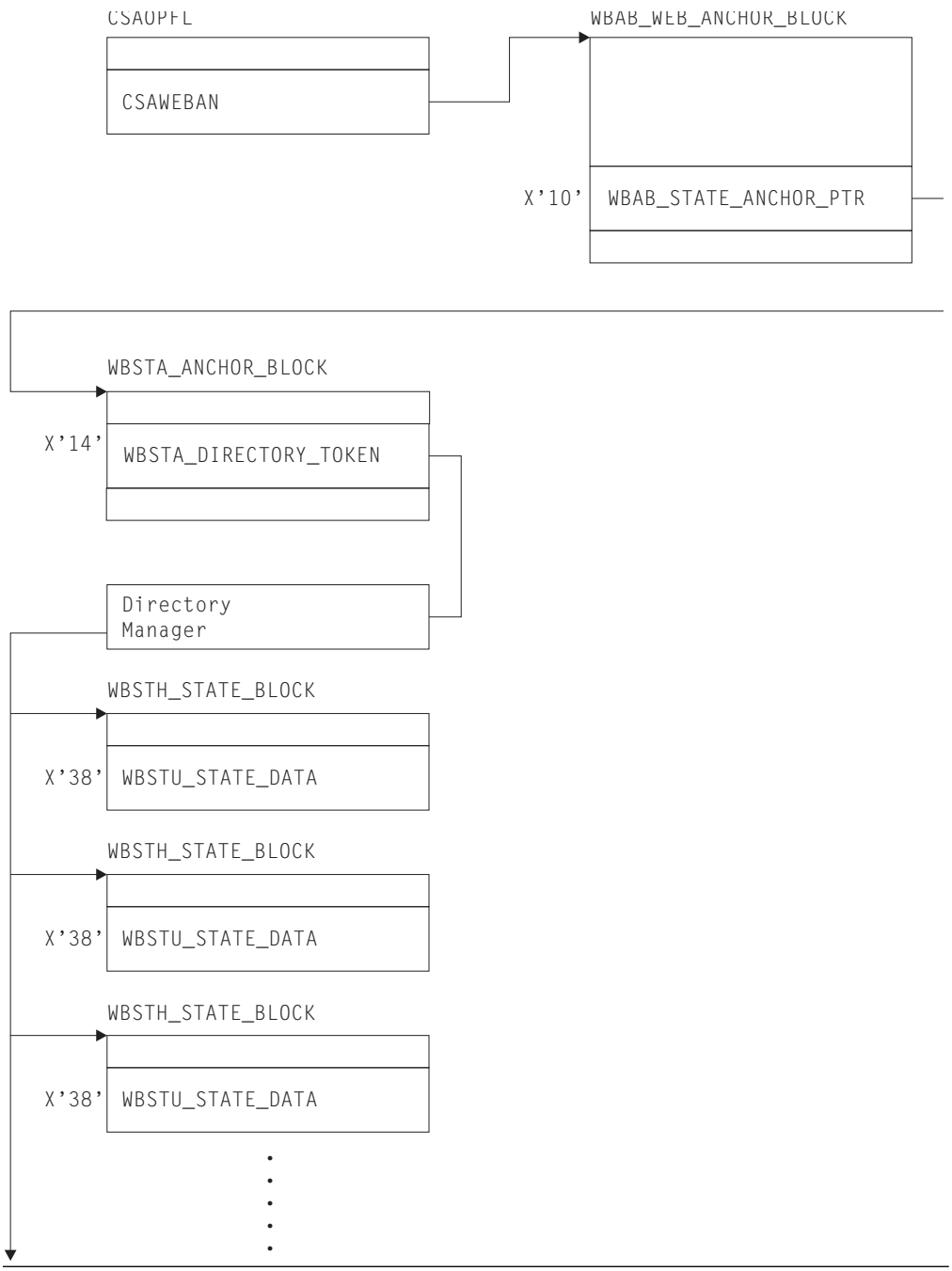


Figure 31. Web support module list

Modules

CICS Web support includes modules used for:

1. Initialization
2. Web attach processing
3. Default analyzer program
4. Alias transaction
5. Web error program

6. HTTP client processing
7. CICS business logic interface
8. CICS Web 3270 support
9. Unescaping function

Initialization, DFHWBIP: DFHWBIP initializes the Web environment at CICS startup.

Web attach processing, DFHWBXN: DFHWBXN is the web attach processing module. It is the initial program that is invoked for transaction CWXN (or an alias of CWXN), which is attached for a new sockets connection that is received on a port that is associated with a TCPIPSERVICE definition with PROTOCOL(HTTP). It is also started for transaction CWXU (or alias), which is attached when the TCPIPSERVICE definition specifies PROTOCOL(USER). It calls the web domain WBSR gate to process the incoming data.

Fast arriving requests might qualify for being processed by directly attached user transactions, and bypassing the web attach task. For more information, seeHTTP requests are processed by directly attached user transactions.

Default analyzer program, DFHWBAAX: DFHWBAAX is the default analyzer program for a TCPIPSERVICE definition that specifies PROTOCOL(HTTP). It does not carry out further processing when a matching URIMAP definition has been found for the request, even if the URIMAP specifies ANALYZER(YES). It tests for the presence of a URIMAP definition, and if the result is positive, returns without performing any analysis on the request URL. This means that the settings specified in the URIMAP definition for the alias transaction, converter program and application program are automatically accepted and used to determine subsequent processing stages.

If no matching URIMAP definition is found, DFHWBAAX gives control to the user-replaceable Web error application program DFHWBERX to produce an error response. This is achieved by setting DFHWBERX as the application program to handle the request.

An alternative analyzer program that has been specified on the TCPIPSERVICE definition, such as the CICS-supplied sample analyzer program DFHWBADX, might carry out analysis on the request and specify alternative settings for the alias transaction, converter program and application program.

When the TCPIPSERVICE definition specifies PROTOCOL(USER), an analyzer program is always required to determine processing for requests (which are treated as non-HTTP requests). DFHWBAAX is not suitable for PROTOCOL(USER). The CICS-supplied sample analyzer program DFHWBADX or a customized analyzer program must be used instead. URIMAP definitions are not used with PROTOCOL(USER).

Alias transaction, DFHWBA: DFHWBA is the alias program. An alias transaction is started by Web attach processing, or by Sockets listener task (CSOL) for each request received from TCP/IP. The transaction ID can be selected by a URIMAP definition or an analyzer program, and the default is CWBA. For CICS Web support, DFHWBA calls the user application program that is specified to process the request. This application program could be specified in a URIMAP definition, or by an analyzer program or converter program. For the CICS business logic interface, DFHWBA calls the CICS business logic interface program.

HTTP client processing, DFHWBCL: DFHWBCL is the HTTP client processing module. It is called by the command interface DFHEIWB (when EXEC CICS WEB commands with the SESSTOKEN option are used in application programs), and the COMMAREA interface DFHWBCLI, to handle outbound HTTP functions, such as opening a session and writing a request to the socket.

CICS business logic interface, DFHWBBLI: DFHWBBLI is the CICS business logic interface program. The interface to the CICS business logic interface program is described in Introduction to the CICS business logic interface.

The CICS business logic interface program is called by DFHWBA. It calls the **Decode** function of a converter program, a CICS application program, or the **Encode** function of a converter program, according to what is specified in its parameter list, and passes the data back to the caller.

DFHWBA1 is the business logic compatibility interface program. In earlier releases, it was the business logic interface program, but it is now a compatibility layer on DFHWBBLI. It accepts data from an old-format business logic interface parameter list, copies it to the new format parameter list, then links to DFHWBBLI.

CICS Web support for 3270 display applications: The modules used by CICS Web support for handling 3270 display applications (sometimes referred to as the CICS Web bridge) are:

DFHWBGB
Removes redundant state data from the system.

DFHWBST
Manages the state data.

DFHWBTC
Performs conversion between 3270 and HTML.

DFHWBTTA
The Web terminal translation application, which sets up the parameters for bridging to transactions from CICS Web support. The program has two aliases, DFHWBTTB and DFHWBTTC.

DFHWBLT
The CICS Web bridge exit.

Exits

Three global user exit points are provided in CICS Web support for HTTP client requests:

XWBAUTH, HTTP client send exit
XWBAUTH is called during processing of an EXEC CICS WEB SEND or EXEC CICS WEB CONVERSE command. It allows you to specify basic authentication credentials (username and password) for a target server. XWBAUTH passes these to CICS on request, to create an Authorization header. The host name and path information are passed to the user exit, with an optional qualifying realm.

XWBOPEN, HTTP client open exit
XWBOPEN is called during processing of an EXEC CICS WEB OPEN or **EXEC CICS INVOKE SERVICE** command. It allows you to specify proxy servers that should be used for HTTP requests by CICS as an HTTP client, and to apply a security policy to the host name specified for those requests.

XWBSNDO, HTTP client send exit

XWBSNDO is called during processing of an EXEC CICS WEB SEND or EXEC CICS WEB CONVERSE command. It allows you to specify a security policy for HTTP requests, in particular for the path component of the request.

Trace

The trace point IDs for this function are of the form WB xxxx. The trace levels are WB 1, WB 2, and Exc.

Command interpreter

The command interpreter demonstrates to the application programmer the syntax of CICS commands and the effects of their execution. It can also be used to perform simple one-off tasks whose nature does not justify the writing of a permanent application.

Design overview

The command interpreter is invoked by the CECI transaction and is an interactive, display-oriented tool that checks the syntax of CICS commands and executes them. Another transaction, CECS, performs only syntax checking.

The user enters a command that is analyzed in the same way as it would be by the command translator, which processes it as if it were part of an application program. The results of this analysis, including any messages, an indication of defaults assumed, and the entire syntax of the command, are then displayed.

When the command is syntactically valid, the user can request its execution. The interpreter calls DFHEIP, passing a parameter list precisely as would be passed during the execution of a program that contained the command.

The interpreter does all this using the same command-language tables as are used by the command translator. These tables contain data that define the syntax of CICS commands and the contents of the parameter lists required by DFHEIP to execute them.

Modules

Module	Function
DFHECIP	Invoked by CECI. Checks that the terminal is suitable. Obtains and initializes working storage. Loads the language tables. Links to DFHECID.
DFHECSP	Same as DFHECIP, but invoked by CECS.
DFHECID	Receives data from the terminal and sends back a display. Analyzes commands. Constructs parameter lists for DFHEIP, which it calls. Deals with PF keys.
DFHEITAB	Command-language table (application programmer commands).
DFHEITBS	Command-language table (system programmer commands).

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

CSD utility program (DFHCSDUP)

The CSD utility program, DFHCSDUP, provides offline services for you to list and modify the resource definitions in the CICS system definition (CSD) file.

DFHCSDUP can be invoked as a batch program, or from a user-written program running either in batch mode or under TSO. The second method provides a more flexible interface to the utility, allowing for the specification of up to five user exit routines to be called at various points during DFHCSDUP processing.

The following commands can be used with DFHCSDUP:

```
ADD
ALTER
APPEND
COPY
DEFINE
DELETE
EXTRACT
INITIALIZE
LIST
PROCESS
REMOVE
SCAN
SERVICE
UPGRADE
USERDEFINE
VERIFY
```

Design overview

When DFHCSDUP is invoked, control passes to the utility command processor (DFHCUCP), which validates commands and invokes the appropriate routine to execute the requested function. Unless DFHCSDUP has been invoked from a user program specifying a get-command exit, DFHCUCP takes a command from the input data set, using DFHCUCB to obtain the command and DFHCUCAB to analyze and parameterize it. When supplied, the get-command exit is invoked from the point during DFHCUCB's processing where commands would otherwise be read from SYSIN (or an alternatively named input data set when DFHCSDUP is invoked from a user program).

Some syntax errors are diagnosed and reported by DFHCUCAB, and further contextual validation takes place in DFHCUCV. Valid commands are then passed to the relevant service routine for execution. If command execution is successful, the next command is processed.

All commands are validated, but the execution of commands from the input data set stops when an incorrect command is encountered, and execution of subsequent commands is also suppressed if an error of severity 8 or higher occurs when the command is executed. When commands are supplied by a get-command exit, however, DFHCSDUP attempts to execute all commands, even if an error is detected in the command syntax or during processing (unless the error is serious enough to warrant an ABEND).

If errors occur while processing commands, error messages in the DFH51xx, DFH52xx, DFH55xx, and DFH56xx series are written to SYSPRINT (or an alternatively named output data set when DFHCSDUP is invoked from a user program).

An ESTAE environment is established by DFHCUCP shortly after the start of DFHCSDUP processing. If an operating system abend subsequently occurs, control passes to the ESTAE exit routine, which then returns to MVS requesting a dump and scheduling a retry routine to get control. This retry routine attempts clean up processing before returning to the caller of DFHCSDUP with a return code of '16'.

To protect the integrity of the CSD, DFHCUCP issues a STAX macro to defer the handling of any attention interrupts that may occur in a TSO environment until all processing associated with the current command has been completed.

DFHCSDUP uses batch versions of RDO routines from the parameter utility program (DFHPUP) and the CSD management program (DFHDMP) to read, write, and update resource definitions on the CSD file. All CSD control functions use the batch environment adapter (DFHDMPBA), which performs environment-dependent VSAM operations on the CSD file. DFHDMPBA also processes all interactions with operating system services.

Modules

DFHCSDUP is link-edited from a number of object modules, including batch versions of routines from DFHPUP and DFHDMP.

Exits

When invoked as a conventional batch program, DFHCSDUP supports only one user exit: the EXTRACT exit, which is invoked at various stages during the processing of an EXTRACT command. The name of the user-written program to get control must be specified by the USERPROGRAM keyword of the EXTRACT command. Details of selected CSD objects are passed to the user exit program so that users can analyze the contents of their CSD in any way they may choose.

When invoked from a user program, DFHCSDUP supports the following five user exits, the addresses of which can be specified in the EXITS parameter of DFHCSDUP's entry linkage:

1. Initialization exit—invoked by DFHCUCP
2. Termination exit—invoked by DFHCUCP
3. EXTRACT exit—invoked by DFHCULIS
4. Get-command exit—invoked by DFHCUCB
5. Put-message exit—invoked by DFHBEP.

Note: A user exit routine specified by the USERPROGRAM keyword of an EXTRACT command is used in preference to any EXTRACT exit routine specified on the entry linkage.

Trace

Trace points are not applicable to offline utilities.

Statistics

The following statistics are maintained by DFHCSDUP, and are written, when appropriate, to SYSPRINT (or alternatively named output data set):

```
CMDSEXOK   Commands executed OK
CMDSINER   Commands in error
CMDSNOTX   Commands not executed
CMDSWARN   Commands with warning messages.
```

These statistics are kept in DFHCUCP's static storage and are always output at the end of processing.

Database control (DBCTL)

An overall description of DL/I database support is given in "DL/I database support" on page 1559. This section gives information that is specific to database control (DBCTL).

Design overview

The CICS support that enables connection to DBCTL, via the database resource adapter (DRA), is based on the CICS resource manager interface (RMI), also known as the task-related user exit interface. However, because it is necessary to provide compatibility with the existing CICS-DL/I implementation (in terms of link-edit stubs, API return codes, and so on), a limited amount of support within CICS itself is provided, but there are no DBCTL release dependencies within the CICS modules.

The main components of the CICS-DBCTL interface are shown in Figure 32:

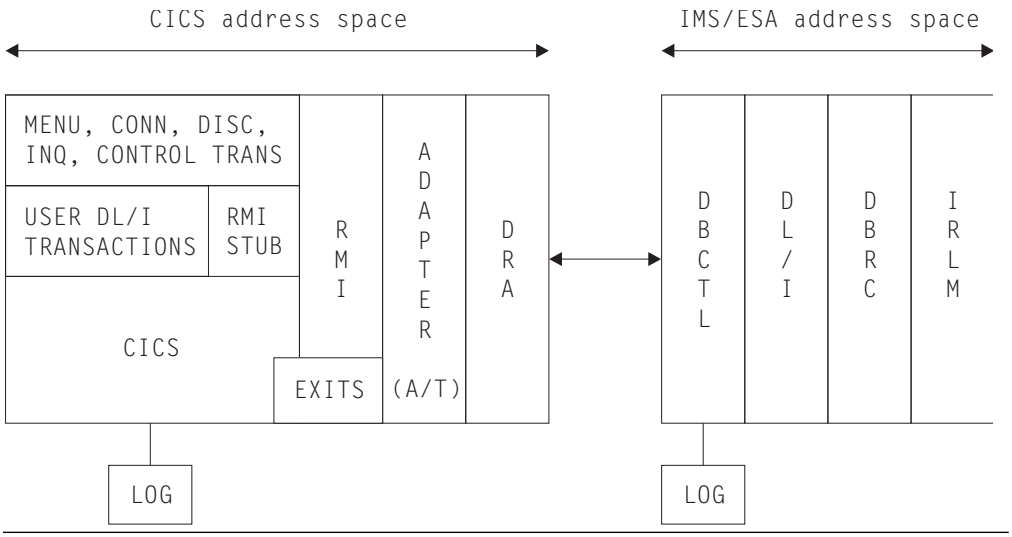


Figure 32. The major components of the CICS-DBCTL interface

- The connection process (CICS-DBCTL)

  CICS-DBCTL connection and disconnection programs

  These programs are used for establishing and terminating the connection with the DRA.

  CICS-DBCTL control program

  This program is responsible for resolving indoubt units of work after a CICS or DBCTL failure. It also outputs messages when DBCTL notifies CICS of a change in the status of the CICS-DBCTL interface.

  When the CICS disconnects from DBCTL, the control program is responsible for invoking the disable program which performs clean up.

DRA control exit
This exit is invoked by the DRA, when connection has been established with the DBCTL address space, to initiate the resynchronization process, that is, to initiate the resolution of indoubt units of work. It is also invoked to handle cases where connection to DBCTL cannot be achieved or when the connection has failed.

DBCTL user-replaceable program
This program is invoked whenever CICS successfully connects to DBCTL and whenever CICS disconnects from DBCTL.

Disable program
This program is invoked when CICS disconnects from DBCTL.

- The DBCTL call processor program

  The function of this program is to issue an RMI call to DBCTL and to maintain compatibility with the existing CICS-DL/I interface in areas such as application program return codes, and so on.
- The interface layer

  The adapter
  The adapter's primary responsibility is interfacing the RMI and DRA parameter lists. Other responsibilities include the issuing of DRA initialization and termination calls, when invoked by the CICS connection and disconnection programs, and the management of CICS tasks, in order to effect an orderly shutdown of the CICS-DBCTL interface.

  DRA suspend and resume exits
  These exits are invoked by the DRA in order to suspend and resume a CICS task while a DL/I call is processed by DBCTL.

  Adapter exits
  There are four exits for use by the adapter:
  - The statistics exit
  - The token exit
  - The monitoring exit
  - The status exit.

Details of these components are described in the following sections.

Note: CICS documentation uses the term "connecting and disconnecting from DBCTL". The DRA documentation refers to "initializing and terminating the CICS-DBCTL interface". In general, these two terms are synonymous.

The connection process:

Connection and disconnection programs: In order to initialize, terminate, and inquire on the status of the interface, a set of four programs is available:

1. Menu program
2. Connection program
3. Disconnection program
4. Inquiry program.

Menu program (DFHDBME): This permits a terminal user to display a menu, which offers the option of connecting and disconnecting from DBCTL.

The menu program passes control to either the connection or the disconnection program, as appropriate, using the COMMAREA to pass any overrides and parameters.

In the case of connection, it offers the ability to supply the suffix of the DRA startup parameter table and the name of the DBCTL region. The DRA startup parameter table contains various parameters, mostly relating to the initialization of the CICS-DBCTL interface, including the name of the DBCTL region and the minimum and maximum number of CICS-DBCTL threads. It also contains the length of time in seconds that the DRA waits after an unsuccessful attempt to connect to DBCTL, before attempting to connect again.

For disconnection, it offers the ability to specify whether an orderly or immediate disconnection from DBCTL is required.

The menu program is intended for use by CICS operators or network controllers, that is, users with special privileges.

BMS maps are used for both the menu and the inquiry programs. It should be noted that the bottom half of the menu screen includes all the items that appear on the inquiry screen, and the values are displayed on entry to the menu program, if they are known. The DRA startup table suffix is not included on the inquiry screen because the DRA startup table contains the application group name which is used for security checking.

After a connection request has been issued, it is possible to issue a disconnection request (orderly or immediate) from the menu program while the connection process is still in progress. After an orderly disconnection request has been issued, it is also possible to issue an immediate disconnection request while the orderly disconnection process is in progress. This has the effect of upgrading the orderly disconnection to an immediate disconnection.

Connection program (DFHDBCON): This program invokes the adapter requesting connection to DBCTL.

This program can be invoked either from the menu program or from the CICS PLT. It issues an ATTACH request of the CICS control program that later carries out resynchronization of indoubt units of work with DBCTL. The control program then issues a WAIT request.

The connection program continues by loading, activating (using the EXEC CICS ENABLE command), and then calling the adapter (using a DFHRMCAL request). A set of parameters is passed to the adapter which includes:

- The CICS applid
- The DRA startup parameter table suffix (optional)
- The DBCTL ID (optional)
- A set of exit addresses.

As a result of the DFHRMCAL request issued from the connection program, the adapter loads the DRA startup/router module from the CICS STEPLIB library and passes control to it, supplying it with various parameters including the CICS applid, DRA startup parameter table suffix, and DBCTL ID. The DRA startup/router module loads the DRA startup table. It then initiates the processes required to establish the DRA and then returns control to the adapter which, in turn, returns control to the connection program which then terminates. Until this point is reached, any DBCTL requests issued from CICS tasks are rejected by the CICS RMI stub (the DBCTL call processor).

The DRA startup/router module is responsible for establishing the DRA environment, using the parameters specified in the DRA startup table in the CICS STEPLIB library, overridden by any parameters passed to it.

The DRA establishes contact with the DBCTL address space and then invokes the control exit to initiate the resynchronization process.

Disconnection program (DFHDBDSC): This program invokes the adapter requesting disconnection from DBCTL.

The disconnection program is used to terminate the DRA environment. Two types of disconnection are available:

Orderly disconnection
: All existing CICS tasks using DBCTL are allowed to run to completion.

Immediate disconnection
: Existing DL/I requests are allowed to complete but no further DL/I requests are accepted.

In both cases a DBCTL U113 abend is avoided. (DBCTL can issue a U113 abend if CICS terminates while there is an active DL/I thread running on its behalf in DBCTL. The thread remains active for the duration of the PSB schedule, but DBCTL would issue a U113 abend if the thread is doing something for the CICS task.)

The disconnection program calls the adapter, using DFHRMCAL, supplying a parameter to indicate the type of termination required.

In the case of immediate disconnection, the adapter issues a DRA TERM call and returns to the disconnection program only when all existing DL/I threads have completed. In the case of orderly disconnection, the adapter assumes responsibility for managing CICS tasks, that is, it continues to accept requests for current tasks using DBCTL until they terminate, but does not allow new CICS tasks to use DBCTL. When the adapter detects that the count of permitted tasks has reached zero, it issues a DRA TERM call.

The disconnection program finally posts the control program to notify it of the fact that the CICS-DBCTL interface has been terminated. The control program then terminates after starting the disable program. The disable program issues a DISABLE command for the adapter, and performs clean up.

It should be noted that the terminal used to invoke the disconnection program is released after the input to the menu screen has been validated, enabling the terminal operator to use other programs. Any further messages from the disconnection process are generated centrally.

Inquiry program (DFHDBIQ): This program enables the user to inquire on the status of the interface. It is intended for a wider audience than the menu program; for example, application programmers.

Control program (DFHDBCT): The control program is invoked in the following circumstances:

- When the control exit is invoked by the adapter on behalf of the DRA
- When a CEMT FORCEPURGE command is issued for a CICS task executing in DBCTL

- When the disconnection program has received a response from the adapter as a result of a CICS-DBCTL interface termination request.

Its function in all cases is to issue messages. It then issues a WAIT after every invocation. Also, it has some special functions in three cases:

1. When contact has been made with DBCTL and resynchronization of in-doubts is required.

   In this case, the control program issues the following command:

   ```
   EXEC CICS RESYNC ENTRYNAME(adapter)
   IDLIST(DBCTL's in-doubts) ...
   ```

   This causes CICS to create tasks for each indoubt unit of work. Each task performs resynchronization and then informs the adapter via the CICS syncpoint manager as to whether the task has committed or backed out. The adapter then notifies the DRA on a task basis.

   The possible calls to the adapter from the CICS syncpoint manager are as follows:

   - Prepare to commit
   - Commit unconditionally[1]
   - Backout[1]
   - Unit of recovery is lost to CICS cold start[2]
   - DBCTL should not be indoubt about this unit of recovery[2].

   Notes:

   [1] These items can be issued as a result of a RESYNC request.

   [2] These items can be issued as a result of a RESYNC request only.
2. When /CHECKPOINT FREEZE has been requested.

   In this case, the control program invokes the disconnection program requesting an orderly disconnection from DBCTL. Generally, an orderly disconnection from DBCTL allows CICS tasks already using DBCTL to continue until task termination. However, when a /CHECKPOINT FREEZE has been requested, DBCTL prevents any PSB schedules from taking place. Thus, in this case, some tasks might be terminated before end of task is reached with a 'DBCTL not available' return code, if they issue a subsequent PSB schedule request.
3. When the disconnection program invokes the control program.

   In this case, the control program starts the disable program.

DRA control exit (DFHDBCTX): The control exit is invoked in the DRA environment in the following circumstances:

- When contact has been established with the DBCTL address space, in order to initiate resynchronization.

  The control exit is invoked in the DRA environment whenever contact has been established with DBCTL, whether invoked by the user or due to the DRA automatically reestablishing contact after a DBCTL failure. The control exit receives an input parameter list that includes the DBCTL ID, DBCTL's list of indoubt units of work, and the DBCTL RSE name. The control exit posts the control program, which performs the resynchronization.
- When the MVS subsystem interface (SSI) rejects the IDENTIFY request to DBCTL, thereby causing the IDENTIFY to fail.

This could occur if the DRA was trying to issue an IDENTIFY request to a DBCTL subsystem that was not running. In this case the control exit sets a response code of '0'. The first time in a connection attempt that the DRA receives a '0' response after an MVS SSI failure, the DRA outputs message DFS690A inviting the operator to reply WAIT or CANCEL. On subsequent failures when a response code of '0' is returned, the DRA waits for the length of time specified in the DRA startup table before attempting the IDENTIFY request again.

- When DBCTL rejects the IDENTIFY request to DBCTL; for example, incorrect application group name (AGN) supplied.

  In this case, the control exit asks the DRA to terminate.
- When the operator replies CANCEL to the DFS690A message during DRA initialization, because contact cannot be established with DBCTL.

  In this case, the control exit notifies the DRA to terminate immediately.
- When DBCTL abnormally terminates.

  In this case, the control exit invokes the control program and then it asks the DRA to issue an IDENTIFY request to DBCTL.
- When the DRA abnormally terminates.

  In this case, it is not possible to access DBCTL from the same CICS session without initializing the CICS-DBCTL interface using the menu program.
- When a /CHECKPOINT FREEZE request has been issued to DBCTL.

  Note that /CHECKPOINT FREEZE is the command used to close down a DBCTL subsystem. In this case the control exit invokes the control program which, in turn, invokes the disconnection program requesting an orderly disconnection from DBCTL. The control exit notifies the DRA to wait for a termination request.

DBCTL user-replaceable program (DFHDBUEX):

The DBCTL user-replaceable program, DFHDBUEX, is invoked whenever CICS successfully connects or disconnects from DBCTL. It provides the opportunity for the customer to supply code to enable and disable CICS-DBCTL transactions at these times.

The program runs as a CICS application and can thus issue **EXEC CICS** requests. The program is invoked with a CICS COMMAREA containing the following parameters:

- Request type: CONNECT | DISCONNECT
- Reason for disconnection: MENU DISCONNECTION | /CHECKPOINT FREEZE | DRA FAILURE | DBCTL FAILURE
- DRA startup table suffix
- DBCTL ID.

Disable program (DFHDBDI): The disable program, DFHDBDI, is invoked when CICS disconnects from DBCTL. It performs clean up, which includes disabling the adapter.

The DBCTL call processor program (DFHDLIDP): Among the functions of the DBCTL call processor program, DFHDLIDP, are:

Issuing DFHRMCAL requests to the adapter: DL/I requests issued from application programs that have been routed to this module are passed on to the adapter. The DBCTL call processor constructs a register 1 parameter list that includes the DL/I parameter list and a thread token. It then issues a DFHRMCAL request.

It is the responsibility of this module to generate the thread token required by the DRA.

Maintaining return code compatibility: If any calls are made to the RMI before the first part of the connection process has completed, that is, before the DFHDBCON program has received a "successful" response code from the DRA via the adapter, error return codes are set in the task control area (TCA) to indicate that DBCTL is unavailable. These codes are put in the user interface block (UIB) by the DL/I call router program, DFHDLI.

Similarly, the DBCTL call processor informs application programs when DBCTL is no longer available; for example, after a DBCTL abend.

Another function of the call processor is to set up the TCA fields, TCADLRC and TCADLTR, with response and reason codes respectively for the call. This ensures that the application program continues to receive responses indicating normal response, NOTOPEN, and INVREQ conditions, with the appropriate response and reason codes in the corresponding UIB fields, UIBFCTR and UIBDLTR, after NOTOPEN and INVREQ conditions have been raised.

Initiating PC abends: If an 'unsuccessful' return code is passed back to CICS as a result of a DBCTL request, indicating that the CICS thread must be abended, the DBCTL call processor issues a PC ABEND, which invokes syncpoint processing to back out changes made to recoverable resources. Various abend codes can be issued. Note that, in the case of a deadlock abend (abend code ADCD) it may be possible to restart the program.

Exception trace entries are output in the case of transaction abends.

Writing CICS messages: For any thread abend in DBCTL, a CICS message is written indicating the abend code passed back to CICS in the field PAPLRETC. Similarly, for any scheduling failures, where the application program receives the UIBRCODE field (UIBFCTR and UIBDLTR fields combined) set to X'0805', the scheduling failure subcode is contained in a CICS message.

The interface layer:

Adapter (DFHDBAT):

Control is passed to the database adapter transformer, DFHDBAT, through the CICS resource manager interface (RMI). It is the responsibility of the adapter to construct the DRA INIT, DRA TERM, and DRA THREAD parameter lists from the RMI parameter list passed to it. It must also transform the DRA parameter list that was passed back after a DL/I call to the format expected by CICS.

DFHDBAT is defined as a threadsafe program that CICS runs as an OPENAPI TRUE on an L8 open TCB.

Part of the DRA parameter list requires two tokens to be generated by CICS:

- A thread token
- A recovery token

The thread token is generated by the DBCTL call processor, and enables a CICS unit of work to be related to a DBCTL unit of work. It is used by the asynchronous RESUME exit to identify the CICS thread to be resumed after a DL/I call.

The 16-byte recovery token is constructed by concatenating an 8-byte unique CICS subsystem name (the CICS APPLID) with the 8-byte CICS RMI recovery token (also known as the unit of work ID).

A further responsibility of the adapter is to manage CICS tasks when an orderly termination of the CICS-DRA interface has been requested by means of the CICS termination program. In this case, the adapter continues to accept DL/I requests from CICS tasks currently using DBCTL, but does not allow new CICS tasks to use DBCTL. When the adapter detects that the count of current tasks has reached zero, it issues a DRA TERM call to shut down the interface.

Table 67 summarizes the types of invocations of the adapter code from CICS, and how the adapter reacts to the individual invocation.

Table 68 summarizes the types of invocations of the adapter code from the DRA, and how the adapter reacts to each individual invocation.

Table 69 on page 1543 summarizes the cases when the adapter invokes the adapter exits.

Table 67. CICS-adapter request summary

Invocation	Invoker	Adapter action
Initialize	Connection program	Issues DRA INIT
Terminate-Orderly	Disconnection program	Issues DRA TERM after waiting for CICS-DBCTL tasks to quiesce
Terminate-Fast	Disconnection program	Issues DRA TERM
PSB Schedule	DBCTL call processor	Issues THREAD SCHED
DL/I request	DBCTL call processor	Issues THREAD DLI
Prepare	CICS sync point manager	Issues THREAD PREP
Commit	CICS sync point manager	Issues THREAD COMTERM
Abort	CICS sync point manager	Issues THREAD ABTTERM
Lost To CICS cold start	CICS sync point manager	Issues COLD request
DBCTL should not be in doubt	CICS sync point manager	Issues UNKNOWN request
Task is terminating	CICS task manager	Issues TERMTHRD
Force Purge Task	Control program	Issues PURGE THREAD
Orderly CICS Term	CICS termination	Issues DRA TERM after waiting for CICS-DBCTL tasks to quiesce
Immediate CICS Term	CICS termination	Issues DRA TERM
CICS is abending	CICS termination	Issues DRA TERM
CICS has been canceled	CICS termination	Returns to CICS

Table 68. DRA-adapter request summary

Invocation from the DRA	Adapter action
CICS-DBCTL connection is complete	Invoke the control exit
MVS SSI has rejected the IDENTIFY request to DBCTL	Invoke the control exit

Table 68. DRA-adapter request summary (continued)

Invocation from the DRA	Adapter action
DBCTL has rejected the IDENTIFY request	Invoke the control exit
Operator has replied CANCEL to message DFS690A	Invoke the control exit
DBCTL has terminated abnormally	Invoke the control exit
DRA has terminated abnormally	Invoke the control exit
/CHECKPOINT FREEZE has been issued	Invoke the control exit
PSB schedule, DL/I, sync point, thread termination, thread purge, or interface termination request is to be suspended	Invoke the suspend exit
PSB schedule, DL/I, sync point, thread termination, thread purge, or interface termination request is to be resumed	Invoke the resume exit

Table 69. Adapter exit summary

Circumstances	Adapter action
Successful completion of THREAD SCHED request	Invoke the monitoring exit
Completion of THREAD COMTERM or THREAD ABTTERM request	Invoke the monitoring exit
DRA thread failure	Invoke the status exit
Resynchronization request issued from CICS recovery manager	Invoke the token exit
CICS orderly or immediate term	Invoke the token exit
CICS ABEND	Invoke the token exit
Completion of DRA TERM issued as a result of a termination request from disconnection program	Invoke the statistics exit
Completion of DRA TERM issued as a result of a CICS orderly termination request	Invoke the statistics exit

Suspend exit (DFHDBSPX): The suspend exit is invoked by the adapter on behalf of the DRA so that a CICS thread can be suspended during the processing of a DL/I call. The suspend exit outputs a trace entry immediately before issuing a WAIT, and a trace entry immediately after it is posted by the resume exit.

The suspend exit is also invoked by the adapter when a disconnection request from the menu is being processed.

Resume exit (DFHDBREX): The resume exit is invoked asynchronously by the adapter on behalf of the DRA, and it is executed in the DRA environment. It handles both normal resume and abnormal resume after an abend of the thread. The resume exit issues an MVS POST.

When a thread fails, the resume exit is invoked and an 'unsuccessful' return code is passed back to the DBCTL call processor, indicating that CICS must issue an abend for that thread (task).

Adapter exits: The following sections describe the adapter exits.

The adapter statistics exit (DFHDBSTX): The statistics exit is invoked by the adapter when the CICS-DBCTL interface has been terminated by the CICS operator using the menu program to request disconnection from DBCTL. The exit is also invoked by the adapter when CICS is terminated in an orderly way.

The function of the exit is to invoke the CICS statistics domain supplying the data that has been returned from the DRA relating to the individual CICS-DBCTL session.

For a /CHECKPOINT FREEZE command, the exit is not invoked, but the statistics domain is called by DFHCDBCT.

The adapter token exit (DFHDBTOX): The token exit is invoked by the adapter when a task is encountered which has not been allocated a thread token, that is, it has not been through the DBCTL call processor module. This occurs for resynchronization tasks and for the CICS termination invocation.

The adapter monitoring exit (DFHDBMOX): The monitoring exit is invoked by the adapter when monitoring data has been returned by DBCTL as a result of a PSB schedule request, and a CICS SYNCPOINT or DLI TERM request. The exit passes the data on to the CICS monitoring domain to update the tasks monitoring information.

The adapter status exit (DFHDBSSX): The status exit is invoked by the adapter in the event of a DRA thread failure, so that resources owned by the failing thread can be transferred to CICS, which then releases the transferred resources during syncpoint processing.

DBCTL system definition: DBCTL system definition is described in the IMS™ *System Definition Reference*.

DBCTL PSB scheduling: When a CICS task requests the scheduling of a DL/I PSB by means of an EXEC DLI SCHEDULE request or DL/I PCB call, and the request is for a DBCTL PSB, control is passed to DFHDLIDP.

Database calls: For DBCTL, DFHDLIDP invokes the CICS RMI to pass control to DBCTL.

DBCTL PSB termination: DBCTL PSB termination is performed during the syncpoint when the resource manager interface (RMI) communicates with DBCTL.

System termination: Support is provided to close down the CICS-DBCTL interface during CICS termination. This should avoid the possibility of causing DBCTL to terminate with a U113 abend because of CICS terminating while DL/I threads are running on its behalf in DBCTL.

To provide the support, there is an extension to the RMI to invoke active adapters at CICS termination.

If CICS termination hangs because the CICS-DBCTL interface does not close down, the operator should type in a /DISPLAY ACTIVE command on the DBCTL console and identify the threads corresponding to the CICS system being terminated. This is possible because the threads' recovery tokens, which are displayed, start with the CICS applid. The operator should then issue /STOP THREAD requests for each thread.

Control blocks

The following diagram shows the major control blocks used to support the CICS-DBCTL interface:

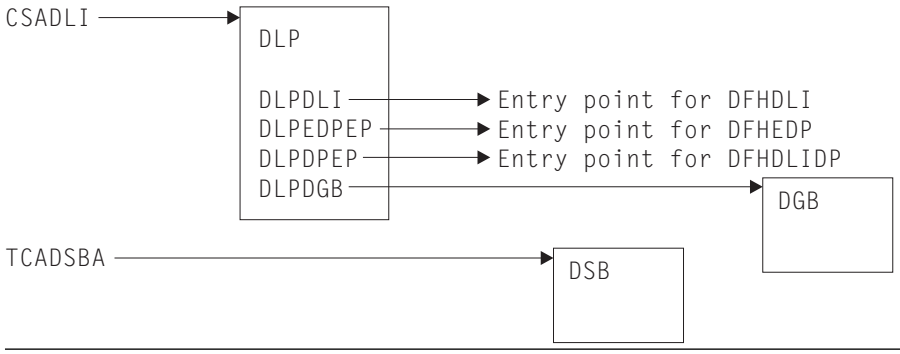


Figure 33. Some control blocks used for DBCTL support

The DL/I interface parameter list (DLP) is described in "DL/I interface parameter list (DLP)" on page 1561.

The DBCTL global block (DGB) is acquired, from storage above the 16MB line, when the CICS-DBCTL interface is first initialized. It lasts for the remainder of the CICS execution.

The DBCTL scheduling block (DSB) is acquired, from storage above the 16MB line, when a task issues a PSB schedule request to DBCTL; that is, the PSB used does not appear in the remote PDIR. The DSB is freed at task termination.

Modules

Module	Description
DFHDBAT	Adapter
DFHDBCON	Initialization program
DFHDBCT	Control program
DFHDBCTX	Control exit
DFHDBDI	Disable program
DFHDBDSC	Termination program
DFHDBIE	Inquiry screens
DFHDBIQ	Inquiry program
DFHDBME	Menu program
DFHDBMOX	Monitoring exit
DFHDBNE	Menu screens
DFHDBREX	Resume exit
DFHDBSPX	Suspend exit
DFHDBSSX	Status exit
DFHDBSTX	Statistics exit
DFHDBTOX	Token exit
DFHDBUEX	DBCTL user exit
DFHDLI	DL/I router program
DFHDLIDP	DBCTL call processor

Exits

The following global user exit points are provided for DBCTL:

- In DFHDBCR: XXDFB and XXDTO
- In DFHDBCT: XXDFA.

Data interchange program

The data interchange program (DFHDIP) supports the batch controller functions of the IBM 3790 Communication System and the IBM 3770 Data Communication System. Support is provided for the transmit, print, message, user, and dump data sets of the 3790 system.

Design overview

The data interchange program is designed as a function manager for Systems Network Architecture (SNA) devices. It is invoked via DFHEDI for command-level requests, or internally by the basic mapping support (BMS) routines using the DFHDI macro. DFHDIP performs the following actions:

1. Determines whether a new output destination has been specified (it retains information about the previous destinations in the data interchange control block) and, if so, builds appropriate FMHs to select the new destination, and outputs these FMHs to the SNA device via terminal control.
2. Invokes the appropriate subroutine to perform the intended function:

   ADD Builds ADD FMH, transmits it and the user data

   REPLACE
   Builds REPLACE FMH, transmits it and the user data

   ERASE
   Builds ERASE FMH and RECID FMH and transmits them

   NOTE Builds NOTE FMH, transmits it, and returns the reply to the user

   QUERY
   Builds QUERY FMH, transmits it, and outputs END FMH

   SEND Outputs user data

   WAIT Waits for completion of the I/O

   END Builds END FMH and transmits it

   ABORT
   Builds ABORT FMH and transmits it

   ATTACH
   Removes FMH from initial input

   DETACH
   Frees the storage used by DFHDIP

   RECEIVE
   Reads a complete record from the logical device.
3. Sets the appropriate return code.

Figure 34 on page 1547 shows the data interchange program interfaces.

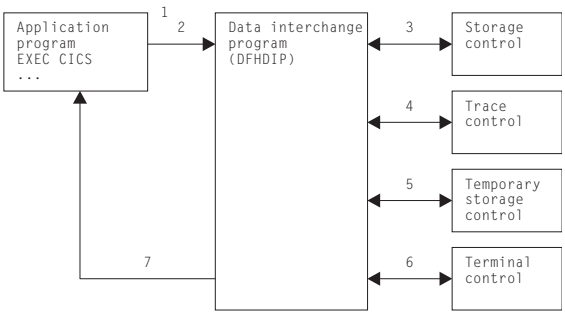


Figure 34. Data interchange program interfaces

Note:

1. The application program invokes DFHEDI (via DFHEIP) which then communicates with DFHDIP by setting fields in the TCA.
2. DFHDIP receives control.
3. If no storage has been obtained for the data interchange block (DIB), storage control is invoked. The storage is chained to the TCTTE. Significant status information, such as the currently selected destination, is remembered in the data interchange block, which is freed at the end of task processing.
4. A trace entry is made.
5. If logging is present (protected task and message integrity) and if a destination change or function change occurs on output, temporary-storage control is invoked to write the DIB to recoverable temporary storage.
6. Terminal control is invoked to output any built FMH and also to output the user data. (DFHTC TYPE=WRITE is issued.) For input requests, DFHTC TYPE=READ requests are issued to obtain a non-null input record.
7. Any errors obtained from the device are decoded and placed in the TCA return code slot. If no errors were detected, a return code of '0' (zero) is returned.

Modules

DFHEDI, DFHDIP

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for the data interchange program:

- AP 00D7, for which the trace level is DI 1.

Distributed program link

Distributed program link enables a program (the client program) in one CICS region to issue an **EXEC CICS LINK** command to link to a program (the server program) running in another CICS region (the resource region). The link can be through intermediate CICS regions. This topic does not contain information about CICS distributed program link (DPL) using IP interconnectivity (IPIC).

For information about IPIC support for distributed program link, see CICS distributed program link.

The communication in distributed program link processing is, from the CICS side, synchronous, which means that it occurs during a single invocation of the client program, and that requests and replies between two programs can be directly correlated.

CICS distributed program link includes information about distributed program link processing.

Figure 35 gives an overview of distributed program link operation.

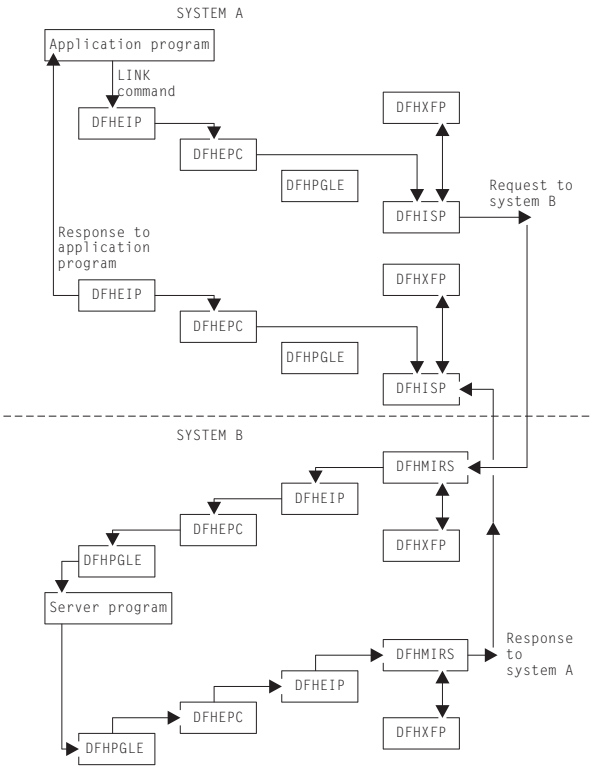


Figure 35. Overview of program link

The DFHEIP module is described in “EXEC interface” on page 1570. This routes all program control requests to DFHEPC. DFHEPC passes all remote LINK requests to the program manager domain (PGLE_LINK_EXEC request). For local programs, program manager links to the program and, on return, it returns to DFHEPC. For remote programs, program manager returns to DFHEPC with and exception response, with a reason code indicating "remote program", and DFHEPC passes the request to the intersystems program, DFHISP. The operation of DFHISP for distributed program link is the same as for function shipping, but only the DFHXFP transformations are used. (See “Function shipping” on page 1706.) The operation of DFHPEP is described in “Program control” on page 1755; the interface to DFHPGLE LINK_EXEC is described in “PGLE gate, LINK_EXEC function” on page 857.

CICS handles session failures and systems failures for distributed program link processing by returning a TERMERR condition to the program that issued the LINK request.

If the server program terminates abnormally and does not handle the abend itself, DFHMIRS returns the abend code to the program that issued the LINK request. This code is the last abend code to occur in the server program, which may have handled other abends before terminating.

A client program using distributed program link can specify that a SYNCPOINT is to be taken in the resource region on successful completion of the server program. That is, any resources updated by the server program (or any associated program) are treated as if they are a separate unit of work.

Modules

The following modules are involved in the distributed program link:

DFHEIP
EXEC interface (see "EXEC interface" on page 1570)

DFHEPC
DFHEIP program control interface (see "Program control" on page 1755)

DFHISP
ISC converse (see "Function shipping" on page 1706)

DFHMIRS
Mirror transaction (see "Function shipping" on page 1706)

DFHPGLE
PG domain - link exec function (see "PGLE gate, LINK_EXEC function" on page 857)

DFHXFP
Online data transformation program (see DFHXFP)

Exits

There are three global user exit points in DFHEPC: XPCERES, XPCREQ and XPCREQC.

Trace

No trace points are provided for this function.

Distributed transaction processing

Distributed transaction processing enables a CICS transaction to communicate with a transaction running in another system. The transactions are designed and coded explicitly to communicate with each other, and thereby to use the intersystem link with maximum efficiency.

The communication in distributed transaction processing is, from the CICS side, synchronous, which means that it occurs during a single invocation of the CICS transaction and that requests and replies between two transactions can be directly correlated.

For more information, see Multiregion operation, ISC and IPIC intercommunications facilities, and Distributed transaction processing overview.

Design overview

CICS handles session failures and systems failures for distributed transaction processing in the same way as for CICS function shipping. See the relevant sections in "Function shipping" on page 1706 for further information.

Distributed transaction processing with MRO and LU6.1

Figure 36 gives an overview of the modules involved with distributed transaction processing for MRO and LU6.1 ISC.

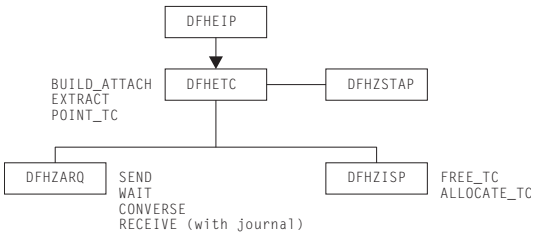


Figure 36. Distributed transaction processing for MRO and LU6.1

The DFHEIP module is described in "EXEC interface" on page 1570. This routes all terminal control requests to DFHETC. DFHETC handles BUILD_ATTACH, EXTRACT, and POINT_TC requests itself. It routes all other requests (SEND, WAIT, CONVERSE, RECEIVE (with journal)), to DFHZARQ, except for FREE_TC and ALLOCATE_TC requests, which are routed to DFHZISP. If the request requires that the user conversation state be returned, DFHETC calls DFHZSTAP. All these modules are described in detail under "Modules" on page 1551.

Mapped and unmapped conversations (LU6.2)

In **mapped** conversations, the data passed to and received from the LU6.2 application programming interface (API) is user data. Mapped conversations use the normal CICS API. Application programs and function shipping requests written for LU6.1 operate using mapped conversations when transferred to LU6.2.

Figure 37 gives an overview of the modules involved with the processing of mapped conversations in LU6.2. ISC.

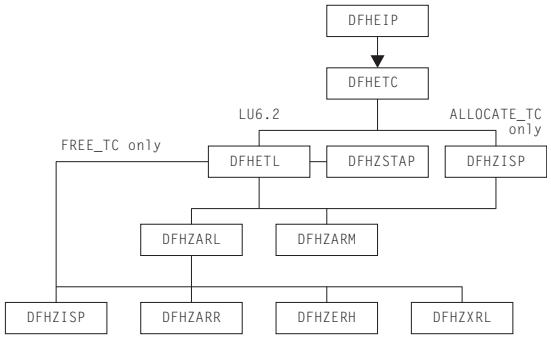


Figure 37. Distributed transaction processing for mapped conversations in LU6.2

The DFHEIP module is described in "EXEC interface" on page 1570. This routes all terminal control requests to DFHETC. DFHETC routes all requests relating to an LU6.2 session to DFHETL except for ALLOCATE_TC requests, which are routed to DFHZISP.

In turn, DFHETL calls DFHZARL to process most requests; it calls DFHZISP to handle FREE_TC requests, and DFHZARM to handle the receipt of unrecognized or unsupported IDs. If the request requires that the user conversation state be returned, DFHETL calls DFHZSTAP.

DFHZARL's processing depends on the type of request; for example, it calls DFHZISP to allocate a TCTTE, DFHZARR to receive data, and DFHZERH for outbound or inbound FMH7 processing. If the request needs to be transaction routed, DFHZARL calls DFHZXRL to route the request to the terminal-owning region (see "Transaction routing" on page 1845).

With the exception of DFHZXRL, all these modules are described in detail under "Modules."

Unmapped conversations (also known as **basic** conversations), are used principally for communication with device-level products that do not support mapped conversations, and which possibly do not have an API open to the user. In unmapped conversations, the data passed to and received from the LU6.2 API contains GDS headers.

Figure 38 gives an overview of the modules involved with the processing of unmapped conversations in LU6.2 ISC.

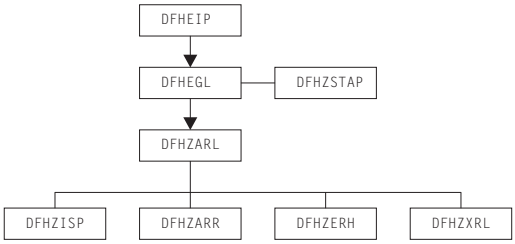


Figure 38. Distributed transaction processing for unmapped conversations in LU6.2

The DFHEIP module is described in "EXEC interface" on page 1570. This passes control to DFHEGL to process GDS commands. DFHEGL routes all GDS conversation-related commands directly to DFHZARL. Some validation of application-provided parameters is performed, and errors are reflected back to the application. If the request requires that the user conversation state be returned, DFHEGL calls DFHZSTAP.

DFHZARL's processing depends on the type of request; for example, it calls DFHZISP to allocate a TCTTE, DFHZARR to receive data, and DFHZERH for outbound or inbound FMH7 processing. If the request needs to be transaction routed, DFHZARL calls DFHZXRL to route the request to the terminal-owning region (see "Transaction routing" on page 1845).

Modules

DFHEGL

DFHEGL processes GDS commands. It is an EXEC interface processor module, and receives control directly from DFHEIP or DFHEIG.

The TCTTE for the session is located and checked for validity. All GDS conversation-related commands are mapped into a DFHLUC macro call and routed

directly to DFHZARL. There is no mapping or unmapping of data, state indicators are not maintained, and there are no FMHs to process.

DFHETC and DFHETL

DFHEIP routes all terminal control requests to DFHETC (the EXEC interface processor for terminal control). DFHETC handles BUILD_ATTACH, EXTRACT, and POINT_TC requests itself. It routes all other requests relating to an MRO or LU6.1 session to DFHZARQ except for FREE_TC and ALLOCATE_TC requests, which are routed to DFHZISP. It routes all other requests relating to an LU6.2 session to DFHETL except for ALLOCATE_TC, which is routed to DFHZISP.

DFHETL performs the following actions:

1. Maps an application request into a form suitable for the DFHZCP and DFHZCC application request modules. This includes mapping application data into GDS records.
2. Detects errors and returns error codes to the application.
3. Unmaps data from GDS records.
4. Maintains state indicators.

For ISSUE CONFIRMATION, CONNECT PROCESS, EXTRACT PROCESS, ISSUE ERROR, ISSUE ABEND, and ISSUE SIGNAL commands, DFHETL:

1. Maps application requests into DFHLUC macro calls.
2. Updates state indicators in the TCTTE (for example, the TCTTE indicator that shows that a CONNECT PROCESS command has been issued).

For SEND and CONVERSE commands, DFHETL:

1. Obtains storage for the processing of outbound application data.
2. Creates attach FMHs, if appropriate.
3. Calls DFHZARL to transmit data.

For RECEIVE commands, DFHETL:

1. Obtains storage for the processing of inbound data.
2. Calls DFHZARL to receive inbound data.
3. Extracts inbound FMHs, as appropriate.
4. Unmaps inbound data.
5. Validates LLs and rejects them if not valid.
6. Manages the passing of data back to the application.
7. If the application issues a RECEIVE NOTRUNCATE request in order to receive only part of the chain, retains the residual data for subsequent RECEIVE requests. DFHETL receives one complete chain of data at a time from DFHZARL.

For WAIT commands, DFHETL calls DFHZARL.

For FREE commands, DFHETL:

1. Checks that the terminal is in the correct state to be freed.
2. Frees the storage used to hold RECEIVE data and the ETCB.
3. Calls DFHZISP to free the session.

DFHZARL

DFHZARL is always invoked through the DFHLUC macro. The DFHLUCDS DSECT maps a parameter list that is set up to pass information to and return information from DFHZARL. DFHZARL manages data in buffers, not in TIOAs. SEND commands cause data to be assembled by DFHZARL into a buffer until a WAIT, or other event, causes the data in the buffer to be transmitted.

DFHZARL invokes DFHZSDL to send data to z/OS Communications Server, by placing requests on the activate chain. However, for optimization, DFHZARL can invoke DFHZSDL directly. Receive requests are handled by DFHZARR.

DFHZARL invokes DFHZUSR to manage the conversation state. The LU6.2 states for each session are stored in the TCTTE for that session.

If the request needs to be transaction routed, DFHZARL calls DFHZXRL to route the request to the terminal-owning region (see "Transaction routing" on page 1845).

Details of DFHZARL's processing for the principal functions of the DFHLUC macro that is used to invoke DFHZARL are in the following topics.

INITIAL_CALL function: This function is requested by DFHZSUP. DFHZARL acquires LU6.2 send and receive buffers. If the transaction is being started as a result of an ATTACH request received from a remote system, DFHZARL transfers any data received with the attach header from the TIOA into the receive buffer.

ALLOCATE function: DFHZARL performs the following actions:

1. If the request passed the address of a profile entry, puts this address in the TCA. If the request passed the name of a profile, calls transaction manager to locate the entry and then puts the address of the entry in the TCA.
2. If the request passed a netname rather than a specific sysid, calls DFHZLOC to locate the TCTTE for the netname and then puts the sysid into the DFHLUC parameter list (as if the caller had the specified sysid).
3. Copies the DFHLUC parameter list to LIFO storage.
4. Calls DFHZISP to allocate a TCTTE.
5. Addresses the TCTTE allocated.
6. Acquires LU6.2 send and receive buffers.
7. Sets the user state machine (DFHZUSRM), request = ALLOCATE_RESOURCE.
8. Returns results to the caller.

SEND function: DFHZARL performs the following actions:

1. Checks the user state machine (DFHZUSRM).
2. Checks the LL count and maintains a record of the outstanding LL count.
3. If the command is SEND LAST, INVITE, or CONFIRM, and the outstanding LL count is nonzero, issues an error message.
4. Sets the user state machine (DFHZUSRM).
5. Issues RECEIVE IMMEDIATE requests, as required, to pick up any negative responses sent by the partner program.

The caller must specify WAIT in the request to force the data to be sent immediately. SEND CONFIRM has an implicit WAIT, and control is not returned until a response has been received, when the state machine is set.

For a SEND request with WAIT, DFHZARL then:

1. Sets the user state machine (DFHZUSRM), request=WAIT.
2. Invokes DFHZSDL for transmission of the data in application area or send buffer.

For a SEND request without WAIT, DFHZARL then:

1. If there is sufficient space in the send buffer for all the data, transfers the data from the application area to the send buffer, and returns control to the caller.
2. Saves the INVITE and LAST indicators.
3. If the send buffer cannot hold all the data, invokes DFHZSDL for an implicit SEND.

If data or a CONFIRM command was sent (or both), DFHZARL then:

1. Checks for a signal received.
2. Checks for exception (negative) response received. If found, calls DFHZERH to handle the error. On return, sets the state machine.
3. Returns results to the caller.

When an implicit send is required, DFHZARL passes the data to DFHZSDL for transmission, passing the address of the data in the send buffer and in the application buffer. The total length of data passed to DFHZSDL is a multiple of the request unit size. On return to DFHZARL, the remaining data is transferred to the send buffer. The parameters passed to DFHZARL, such as INVITE and LAST, are not transmitted by DFHZSDL.

RECEIVE function: DFHZARL passes the DFHLUC parameter list, specifying the type of receive required, to DFHZARR for processing (see "DFHZARR" on page 1556).

ISSUE ERROR or ABEND function: DFHZARL is called as a result of an ISSUE ERROR or ISSUE ABEND command, and performs the following actions:

1. Sets the user state machine
2. Calls DFHZERH.

DFHZARM

DFHETL may invoke DFHZARM to provide service functions. DFHZARQ passes control to DFHZARM instead of initiating DFHZSDS, DFHZRVS, and so on, if DFHZARQ finds that it is an LU6.2 session. This applies to the SEND, WAIT, RECEIVE, and SIGNAL commands. The same applies to DFHZISP for the FREE command.

DFHZARM translates the data stream to and from a format suitable for invoking DFHZARL. In particular:

- An LU6.2 attach FMH may have to be requested.
- Data must be passed in GDS record format (structured fields preceded by an LLID).

DFHZARM is invoked via the DFHLUCM macro, which has seven executable options:

- DFHLUCM TYPE =
  - SEND
  - RECEIVE

– WAIT
– SIGNAL
– FREE
– INVALID_ID

DFHLUCM TYPE=STORAGE defines the storage in LIFO for passing primary input and output. The DSECT name is DFHLUMDS. TCTTE contains the secondary input and output. The principal functions are described in the following sections.

SEND function: DFHZARM performs the following actions:

1. Maps the data into GDS record format. The IDs used are:
   - X'12F1'
   - X'12F2'
   - X'12FF'.
2. Examines bits set in the TCTTE by DFHZARL to determine which DFC to apply.
3. Invokes DFHZARL (using a DFHLUC TYPE=SEND,LIST=... macro call) to pass the GDS records and DFC indicators.
4. Updates the state bits in TCTTE as necessary.
5. Interrogates the LU6.2 ATTACH_FMH_BUILT bit in the TCTTE, which was set by DFHZSUP or DFHETL. This bit indicates whether this is first SEND. If an LU6.2 attach header has not already been built as a result of a CONNECT PROCESS command, DFHZARM issues CONNECT_PROCESS to DFHZARL, assuming synclevel 2, before sending the data.

RECEIVE function: DFHZARM performs the following actions:

1. Calls DFHZARL using TYPE=BUFFER. Two calls are made. On the first call, the first 4 bytes (LLID) are retrieved into LIFO. These are examined and the LL is used to determine the TIOA size and to specify the length required in the second call.
2. On the second call, retrieves the remainder of the data directly into the TIOA. If the LL indicates concatenated data, a series of calls is made to retrieve all the data.

FREE function: The FREE function is used, for example, by DFHZISP to ensure that I/O has completed and CEB sent, using null data if necessary.

INVALID_ID function: The INVALID_ID function is used by DFHETL and DFHZARM itself. It handles the receipt of unrecognized or unsupported IDs. DFHZARM calls DFHZARL with ISSUE_ERROR (X'0889010x'), and sends a record with ID X'12F4' followed by the unrecognized ID. If the remote system responds, DFHZARM turns the flows around so that the local system can try again.

LU6.1 chains: An LU6.1 chain corresponds to one SEND command. LU6.2 chains are bigger, so:

- For outbound data, DFHZARM maps one SEND into one structured field (concatenated if necessary).
- For inbound data, DFHZARM retrieves one (possibly concatenated) field and calls it a chain, thus preserving compatibility.

DFHZARQ

DFHETC routes SEND, WAIT, CONVERSE, and some RECEIVE commands to DFHZARQ. RECEIVE commands are passed to DFHZARQ if input journaling is in effect. Otherwise, the call is routed to DFHZARL directly.

DFHZARQ passes control to DFHZARM instead of initiating DFHZSDS, DFHZRVS, and so on, if DFHZARQ finds that it is an LU6.2 session. This applies to the SEND, WAIT, RECEIVE, and SIGNAL commands.

Reasons for calling DFHZARQ are:

- To avoid duplication of existing code
- So that DFHZCP performs journaling of outbound data
- To perform an implicit CONNECT PROCESS if SEND or CONVERSE is the next session-related command after ALLOCATE
- To enable the SNA change direction (CD) and end bracket (EB) indicators to flow with the data.

DFHZARR

DFHZARR is called by DFHZARL to handle receive requests. Details of the processing follow.

RECEIVE function: This function must be able to handle receipt of the following:

- Application data
- FMH7s and ER1s (negative responses)
- PS_headers (Prepares, Request_commits)
- Indicators such as CD, CEB, and RQD2
- Signal.

Figure 39 gives an overview of the modules involved with the processing of receive requests. These modules are described in CICS executable modules.

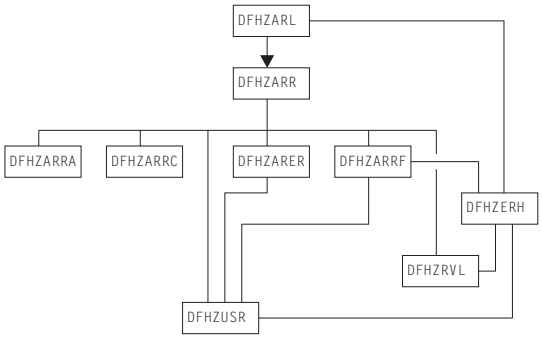


Figure 39. Distributed transaction processing of LU6.2 receive requests

DFHZARL passes the DFHLUC parameter list, specifying the type of receive required, to DFHZARR.

DFHZARR then performs the following actions:

1. Checks that request is valid; if not, returns error codes.
2. Initializes the application and LU6.2 receive buffers (by calls to DFHZARRA and the DFHZARR0 subroutine of DFHZARR respectively).
3. Calls DFHZARRC to determine what to process next.

4. Depending on DFHZARRC's response, calls the relevant subroutine.
5. If "enough" (or all that can be) has not been received, loops back to step 3 on page 1556; otherwise step 6.
6. Tests for (and returns) signal when it has been received.

The results of the receive are passed back to the caller in the DFHLUC parameter list.

To control this processing, DFHZARR uses the variables **receive_type** and **what_next**, as follows.

receive_type can have the following values:

RECEIVE_WAIT
: Request was a receive and wait.

RECEIVE_IMMEDIATE
: Request was a receive immediate.

LOOK_AHEAD
: All the allowed user data has been received, but only one receive immediate call to the DFHZARR1 subroutine of DFHZARR is permitted to attempt to pick up indicators such as CD, CEB, or a PS_header.

NO_MORE_RECEIVES
: No more calls to DFHZARR1 are permitted, but processing may continue with what has already been received.

NO_RECEIVE_LOOK_AHEAD
: All the allowed user data has been received. An attempt must be made to pick up indicators such as CD, CEB, or a PS_header without a call to DFHZARR1. This value is only required for a receive immediate request.

RECEIVE_COMPLETE
: Receive processing is finished.

The first two values are possible initial values of receive_type, and the other four are used as the receive progresses.

what_next is an output of DFHZARRC, and represents what is next to be processed. It can have the following values:

DATA_RECORD
: Application data

FMH_RECORD
: FMH7 in the buffer

PS_HEADER_RECORD
: Prepare or Request_commit

PARTIAL_LL
: First byte of a logical record only, therefore cannot tell whether it is a DATA_RECORD or PS_HEADER_RECORD

CD Change Direction

CEB Conditional End Bracket

RQD2 RQD2 without CD or CEB

RQD2_CD
: RQD2 with CD

RQD2_CEB
: RQD2 with CEB

ER1 Negative response

EMPTY_BUFFER
: Nothing available to receive.

DFHZERH

DFHZERH is called by DFHZARL or DFHZARRF, when it is required to transmit error information or when error information has been received.

Outbound errors: For outbound errors, DFHZERH is invoked by DFHZARL following an ISSUE_ERROR, ISSUE_ABEND, or SYNC_ROLLBACK request.

An FMH7 must be transmitted, but can only be transmitted if the session is in the send state.

If the session is in the receive state, DFHZERH:

1. Sends a negative response
2. Purges the remaining data to end of chain.

In all cases, DFHZERH then:

1. Checks that the session is still in bracket
2. Clears the send buffers
3. Calls DFHZARL to send the FMH7.

Inbound errors: For inbound errors, DFHZERH is invoked by DFHZARL or DFHZARRF when a process-level exception response or an FMH7 has been received.

If an exception response is received while in the send state, DFHZERH purges the present output buffer and sends 'LIC,CD,RQE1' to put the conversation into receive state—so that the following FMH7 can be received.

If an FMH7 is received, DFHZERH examines the associated sense code and any GDS error log data, then returns to its caller.

DFHZISP

DFHZISP is called by DFHETC to perform ALLOCATE_TC requests. (ALLOCATE commands are passed to DFHZISP because DFHETC cannot check the session type until the session is allocated.)

DFHZISP is also called to perform FREE_TC requests.

DFHZSTAP

DFHZSTAP provides a means of determining the conversation state of an MRO or LU6.2 session from the application side. This function is required if the application issues an EXEC CICS EXTRACT ATTRIBUTES command with the STATE option, or a conversation-based command with the STATE option.

For MRO, modules that invoke MVS services via the DFHTC macro also update the conversation state information with a DFHZCNVM TYPE=PUT macro call. When an application requires the conversation state of a session, DFHETC calls DFHZSTAP using a DFHZSTAM TYPE=GETCURRSTATE macro, which returns a value representing the conversation state of the session.

For LU6.2, DFHZUSR is called to maintain the user conversation state machine. (See "z/OS Communications Server LU6.2" on page 1885 for further details.) When an application requires the conversation state of a session, DFHETL (mapped) or DFHEGL (unmapped) calls DFHZSTAP using a DFHZSTAM

TYPE=GETCURRSTATE macro. DFHZSTAP examines the DFHZUSR state machine and maps the information into a value representing the conversation state of the session.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for distributed transaction processing:

- AP FDxx, for which the trace level is TC 1.
- AP FExx (LU6.2 application receive requests), for which the trace levels are TC 2 and Exc.

DL/I database support

Facilities for accessing DL/I databases and database control (DBCTL) support are available only with IMS.

Within a single CICS system, the following types of support can be available:

- DBCTL support present. For specific information about DBCTL, see "Database control (DBCTL)" on page 1535.
- Remote DL/I and DBCTL support present (the PDIR system initialization parameter is specified). For specific information about remote DL/I, see "Remote DL/I" on page 1760.

The rest of this section covers DL/I database support in general.

Design overview

The following types of DL/I requests can be made by a CICS system:

- EXEC DLI statements (converted into standard CALL DLI statements by DFHEDP)
- CALL DLI statements.

CICS support for DL/I is provided as follows:

1. A router component

   This component determines whether the call is using a remote or DBCTL PSB, and passes control to the appropriate call processor. This component is described in more detail later in this section.
2. A DL/I call processor

   This component is subdivided into:

   - A remote DL/I call processor
   - A DBCTL DL/I call processor.

   Each call processor deals with a specific interface that is described in the appropriate section of this book for the remote DL/I function and the DBCTL function.

Figure 40 on page 1560 shows the relationships between the components of the CICS-DL/I interface.

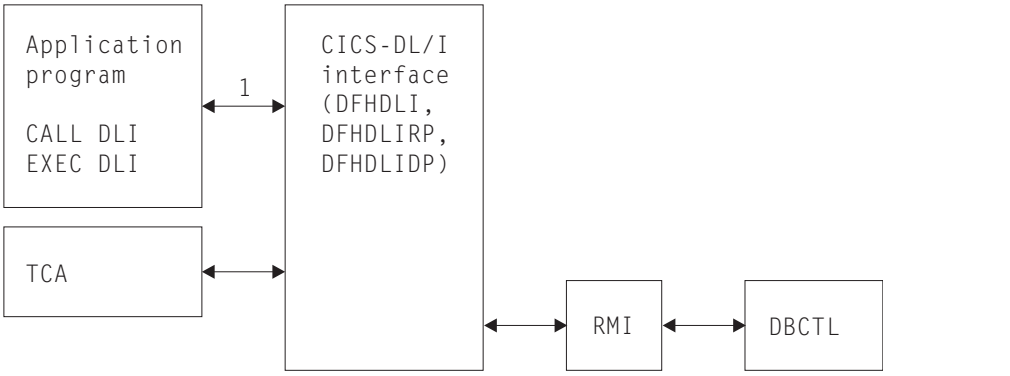


Figure 40. CICS-DL/I interfaces

Note:

1. When DL/I functions are requested by an application program or a CICS control module through execution of a CALL or CALLDLI macro, DFHDLI sets the required fields in the TCA. EXEC DLI statements are converted into standard CALL DLI statements by DFHEDP.

   If the request is for a remote database, DFHDLI passes control to DFHDLIRP. If the request is for a DBCTL database, DFHDLI passes control to DFHDLIDP.

   In addition to processing DL/I input/output requests, the DL/I interface, on request, schedules and terminates DL/I program specification blocks (PSBs).

The remainder of this section is concerned with the router component.

The router component (DFHDLI)

The router component receives a request in standard CALL DLI parameter lists. At schedule time, it determines whether the request is a remote or DBCTL request.

Among the functions of the router are the following:

Deciding where to process a request: At PSB schedule time, the router determines whether the DL/I requests issued from the application program should be routed to DBCTL or another CICS system (remote). The presence (or absence) of the PSB used in the PDIR determines where the call gets routed.

If no PDIR exists (that is, the PDIR=NO system initialization parameter is specified or is allowed to default), the request is routed to the DBCTL call processor.

If a PDIR has been specified, the router module scans the PDIR. All entries in the PDIR have a SYSIDNT option specified. If the PSB is not found in the PDIR, or if the PDIR entry specifies a SYSIDNT that is the SYSIDNT of the CICS system that is currently running, the request is routed to the DBCTL call processor. Otherwise, the request is routed to the remote call processor.

All DL/I requests are routed to the same DL/I call processor as the corresponding PSB schedule request in the same unit of work.

Initiating synchronization processing: The router provides special handling of the DL/I TERM call. When the router detects a TERM call, it forces a syncpoint, causing CICS to carry out syncpoint processing for the task.

Generating CICS trace records: The router module generates CICS trace records at DL/I call entry and DL/I call exit.

Control blocks

DL/I database support uses the control blocks DIB, DLP, and UIB, which are shown in Figure 41.

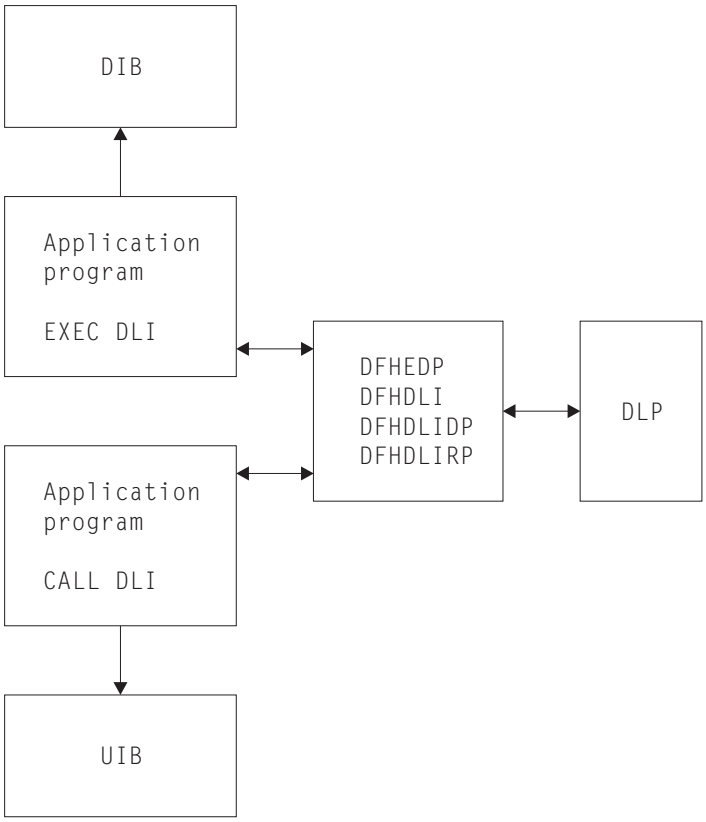


Figure 41. Control blocks for DL/I database support

DL/I interface block (DIB)

When an application program issues EXEC DLI requests, it uses the user DL/I interface block (DIB) instead of the user interface block (UIB). On return, DFHEDP extracts data from the UIB to place in the DIB. The storage for the user DIB is part of the application program. The definition of the user DIB is automatically inserted by the CICS translator for an EXEC DLI application program.

DL/I interface parameter list (DLP)

The DL/I interface parameter list (DLP) is a global DL/I interface control block that lasts for the duration of a CICS session, and contains information relating to the type of DL/I support present in the CICS system. The DLP is created during CICS startup and is addressed by CSADLI in the CSA optional features list.

User interface block (UIB)

The user interface block (UIB) is the control block used by the CALL and CALL DL/I interfaces to pass response codes and the PCB address list to application programs using CALL DL/I services. The UIB is acquired when a task issues its first PSB schedule request specifying that it requires a UIB. The UIB is freed at task termination. TCADLIBA points to the UIB.

See Data areas for a detailed description of these control blocks.

Modules

Figure 42 shows the module flow of DL/I requests to the DL/I call processors. DL/I requests from application programs made using CALL or CALL DL/I are handled by DFHEIP. Requests made using EXEC DLI are passed from DFHEIP, to the RMI, to DFHEDP. Next, three main CICS-DL/I interface modules process the requests. The first module, DFHDLI, determines what DL/I request is being made and then passes control to one of two call processors. These are the DBCTL DL/I call processor, DFHDLIDP, and the remote call processor, DFHDLIRP. DFHDLIDP routes the requests to the RMI, then DFHDBAT, to IMS modules. DFHDLIRP routes the request to DFHISP.

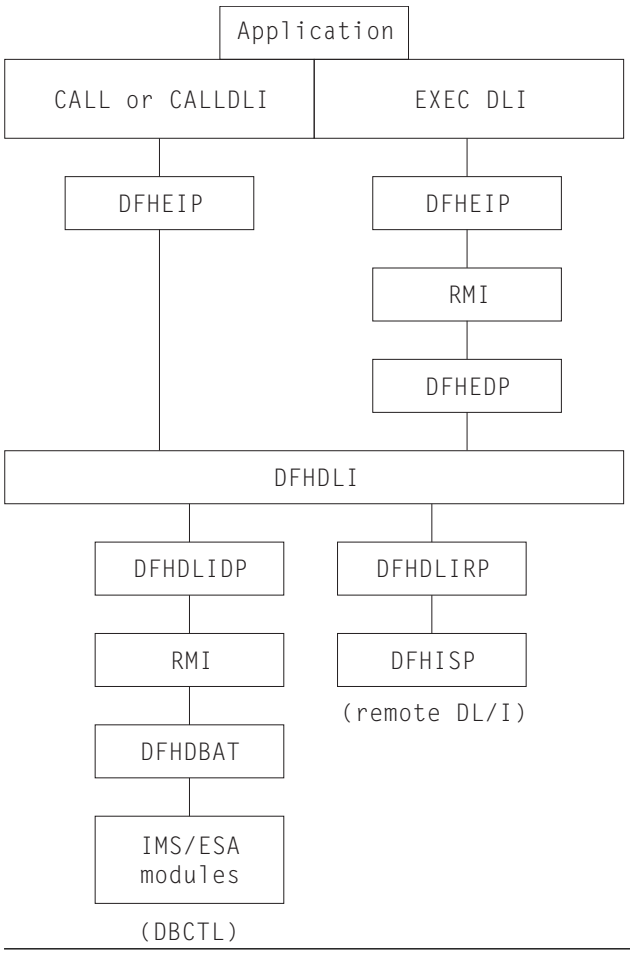


Figure 42. Module flow of DL/I requests to the DL/I call processors

The common CICS-DL/I interface modules consist of the following:

- DFHDLI—contains the code for routing requests to DFHDLIRP and DFHDLIDP
- DFHDLIDP—contains the code for DBCTL requests.
- DFHDLIRP—contains the code for remote DL/I requests

Exits

The following global user exit points are provided in DFHDLI: XDLIPRE and XDLIPOST.

For further information about these, see CICS action on return from the control program and the Overview of Database Control (DBCTL).

Trace

The following point ID is provided for DL/I and DBCTL:

- AP 03xx, for which the trace levels are RA 1, RA 2, and Exc.

Dump utility program (DFHDU710)

The dump utility program (DFHDU710) runs offline (in batch mode) to produce a printout of the CICS transaction dumps from a CICS transaction dump data set (DFHDMPA or DFHDMPB).

Design overview

DFHDU710 operates in batch mode while one of the dump data sets is closed. Each area, program, and table entry is identified, formatted, and printed separately, with both actual and relative addresses to facilitate analysis. You can select single or double spacing of dumps when the dump utility program is executed.

The CICS dump data set (DFHDMPA or DFHDMPB) contains a number of CICS transaction dumps. These are produced as the result of a transaction abend or a user-application **EXEC CICS DUMP TRANSACTION** request.

DFHDU710 runs as a stand-alone program in batch mode to format and print the contents of a transaction dump data set. Parameters specified on the SYSIN data set can be used to print only selected dumps or an index of the dumps in the data set.

For further details about DFHDU710, see Dump utilities (DFHDU680 and DFHPD680).

Data sets

There are three sources of data for DFHDU710:

Parameters on JCL EXEC statement
: A character string of keywords that can be specified to control the layout and format of the dumps.

SYSIN
: Records specifying the criteria to be used in selecting which of the dumps on the data set are to be printed.

DFHDMPDS
: The transaction dump data set.

There are two output files:

DFHPRINT
: The print file for the formatted transaction dump.

DFHTINDX
: The print file for the index of dumps on the data set.

Processing

Figure 43 on page 1564 shows the dump utility program interfaces.

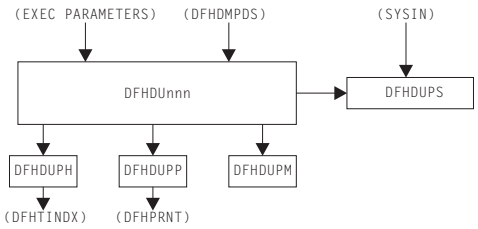


Figure 43. Dump utility program interfaces

The overall flow of the processing within DFHDU710 is as follows. Unless otherwise indicated, all processing is performed by DFHDUPR, the main component of DFHDU710.

1. Process the EXEC parameters if they are present.
2. Call DFHDUPP to open the print data set DFHPRINT.
3. Open the dump data set DFHDMPDS.
4. Read the dumps from DFHDMPDS. For each dump there are four categories of records:
   Dump header record
   Call DFHDUPS to see whether this dump is required for printing. On the first time through, DFHDUPS reads the selective print information from SYSIN. DFHDUPS also calls DFHDUPH to add the dump to the dump index data set DFHTINDX. DFHDUPH opens DFHTINDX on its first invocation.
   Module index records
   DFHDUPM is called to accumulate the module index information in a table in main storage.
   Other data records
   The data is formatted into print lines and DFHDUPP is invoked to write them to DFHPRINT.
   Dump trailer record
   DFHDUPM is invoked to sort and format the module index records. DFHDUPP is called to write them to DFHPRINT.
5. When the end of the dump data set is encountered:
   a. DFHDUPP is called to close DFHPRINT.
   b. DFHDUPH is called to close DFHTINDX.
   c. DFHDUPR closes DFHDMPDS.
6. DFHDU710 terminates.

Modules

Module	Function
DFHDUPR	Controlling routine, responsible for reading information from the dump data set DFHDMPDS.
DFHDUPS	Receives the address of a dump header record from the dump data set, and decides whether this dump fulfils the criteria for printing. On first entry, reads and stores the selective print parameters from SYSIN.
DFHDUPP	Is responsible for all access to the print file DFHPRINT, namely for OPEN, CLOSE, and PUT requests.
DFHDUPH	Writes line to dump index for each dump header record encountered. On first entry, opens the index file DFHTINDX.

Module	Function
DFHDUPM	Invoked for each module index entry found to save information. Invoked when dump trailer record found to format and print the complete module index.

Copy books

Copy book	Function
DFHDUPSC	Contains the definition of the parameter list passed to DFHDUPS.
DFHDUPMC	Contains the definition of the parameter list passed to DFHDUPM.
DFHDUPPC	Contains the definition of the parameter list passed to DFHDUPP.

Exits

Global user exit points are not applicable to offline utilities.

Trace

Trace points are not applicable to offline utilities.

Dynamic allocation sample program (IBM 3270 only)

Any data set defined to file control can be allocated to CICS dynamically when the file is opened, rather than at CICS job initiation time. This allocation takes place automatically if job control statements for the data set are not included in the CICS job stream, and if both the data set name and the disposition have been specified in the resource definition when the data set is opened.

The dynamic allocation sample program provides an alternative way to perform dynamic allocation. When used with a terminal of the IBM 3270 Information Display System, it gives the user access to the functions of DYNALLOC (SVC 99) in MVS. This can be used, in conjunction with master terminal functions and suitable operating procedures, to allocate and deallocate any file that CICS can dynamically open and close.

Design overview

The program runs as a CICS transaction, using CICS functions at the command level wherever possible. It does not modify any CICS control blocks. Only the DYNALLOC function is available through the program; any manipulation of the environment before or after the DYNALLOC request must be done by other means.

CICS supplies sample resource definitions for the program load module, DFH99, and the transaction, ADYN, that invokes it. These definitions are in the group DFH$UTIL. Note that DFH99 ***must*** be defined with EXECKEY(CICS).

The flow in a normal invocation is as follows. The main program, DFH99M, receives control from CICS, and carries out initialization. This includes determining the screen size and allocating input and output buffer sections, and issuing initial messages. It then invokes DFH99GI to get the input command from the terminal. Upon return, if the command was null, the main program terminates, issuing a final message.

The command obtained has its start and end addresses stored in the global communication area, COMM. The main program allocates storage for tokenized text, and calls DFH99TK to tokenize the command. If errors were detected at this stage, further analysis of the command is bypassed.

Following successful tokenizing, the main program calls DFH99FP to analyze the verb keyword. DFH99FP calls DFH99LK to look up the verb keyword in the table, DFH99T. DFH99LK calls DFH99MT if an abbreviation is possible. Upon finding the matching verb, DFH99FP puts the address of the operand section of the table into COMM, and puts the function code into the DYNALLOC request block.

The main program now calls DFH99KO to process the operand keywords. Each keyword in turn is looked up in the table by calling DFH99LK, and the value coded for the keyword is checked against the attributes in the table. DFH99KO then starts off a text unit with the appropriate code, and, depending on the attributes the value should have, calls a conversion routine

- For character and numeric strings, DFH99CC is called. It validates the string, and puts its length and value into the text unit.
- For binary variables, DFH99BC is called. It validates the value, converts it to binary of the required length, and puts its length and value into the text unit.
- For keyword values, DFH99KC is called. It looks up the value in the description part of the keyword table using DFH99LK, and puts the coded equivalent value and its length into the text unit.

When a keyword specifying a returned value is encountered, DFH99KO makes an entry on the returned value chain, which is anchored in COMM. This addresses the keyword entry in DFH99T, the text unit where the value is returned, and the next entry. In this case the conversion routine is still called, but it only reserves storage in the text unit, setting the length to the maximum and the value to zeros.

When all the operand keywords have been processed, DFH99KO returns to the main program, which calls DFH99DY to issue the DYNALLOC request.

DFH99DY sets up the remaining parts of the parameter list, and if no errors too severe have been detected, a subtask is attached to issue the DYNALLOC SVC. A WAIT EVENT is then issued against the subtask termination ECB. When the subtask ends, and CICS dispatches the program again, the DYNALLOC return code is captured from the subtask ECB, with the error and reason codes from the DYNALLOC request block, and a message is issued to give these values to the terminal.

DFH99DY then returns to the main program, which calls DFH99RP to process returned values. DFH99RP scans the returned value chain, and for each element issues a message containing the keyword and the value found in the text unit. If a returned value corresponds to a keyword value, DFH99KR is called to look up the value in the description, and issue the message.

Processing of the command is now complete, and the main program is reinitialized for the next one, and loops back to the point where it calls DFH99GI.

Messages are issued at many places, using macros. The macro expansion ends with a call to DFH99MP, which ensures that a new line is started for each new message, and calls DFH99ML, the message editor. Input to the message editor is a list of tokens, and each one is picked up in turn and converted to displayable text. For

each piece of text, DFH99TX is called, which inserts the text into the output buffer, starting a new line if necessary. This ensures that a word is never split over two lines.

When the command has been processed, the main program calls DFH99MP with no parameters, which causes it to send the output buffer to the terminal, and initialize it to empty.

Control blocks

The sample program does not have any control blocks.

Modules

Module	Function
DFH99BC	Convert to binary target
DFH99CC	Character and number string conversion
DFH99DY	Issue SVC 99 and analyze result
DFH99FP	Process function keyword
DFH99GI	Format display and get input
DFH99KC	Keyword value conversion
DFH99KH	List keywords for help
DFH99KO	Process operator keywords
DFH99KR	Convert returned value to keyword
DFH99LK	Search key set for given token
DFH99ML	Build message text from token list
DFH99MM	Main control program (entry point DFH99M)
DFH99MP	Message filing routine
DFH99MT	Match abbreviation with keyword
DFH99RP	Process returned values
DFH99T	Table of keywords
DFH99TK	Tokenize input command
DFH99TX	Text display routine
DFH99VH	List description for help

Exits

No global user exit points are provided for this function.

Trace

This sample program makes no entries in the trace other than the normal entries for a CICS user transaction.

External interfaces

SVC 99—MVS DYNALLOC SVC.

ECI over TCP/IP

The IP ECI (IE) domain processes external call interface (ECI) requests that arrive from a CICS client that is connected to CICS by a TCP/IP network. It attaches a mirror task to issue the appropriate program link request, and returns the results to the client.

Mirror tasks attached to service ECI requests have the facility type of SOCKET in their association data. For information about tracking origin data, see Association data.

Design overview

The CICS code that processes external call interface (ECI) requests that arrive from a TCP/IP network via the Sockets Domain (SO) is mostly contained within the IP ECI (IE) domain. Some code that is logically part of the function runs in AP domain.

SO domain works by attaching a listener task (CIEP for IPECI) to handle incoming data, and IE domain attaches a mirror task (CPMI) to issue the program link request and return any resulting output.

There are five logically separate pieces of code for this function:

- IE domain initialization and termination code in DFHIEDM.
- The AP domain part of the listener task, in program DFHIEP.
- The IE domain part of the listener task, in the PROCESS_ECI_FLOW function of program DFHIEIE.
- The AP domain part of the mirror task, in programs DFHMIRS and DFHIEXM.
- The IE domain part of the mirror task, in the SEND, RECEIVE and SEND_ERROR functions of program DFHIEIE.

See "IP ECI (IE) domain" on page 543 for more information.

Listener task, CIEP

The CIEP task is attached by SO domain when it receives data on the port specified in the IPECI TCPIPSERVICE. The CIEP transaction handles control flows directly, or attaches a mirror task to issue the ECI program link request.

The valid flows that may be received by CIEP are:

- Attach FMH for CCIN INSTALL

  The initial flow from a client is an attach for the CCIN transaction to install the client. No attach is done as IE domain handles the installation processing internally.
- Attach FMH for CCIN UNINSTALL

  A client can terminate its connection with CICS by sending a CCIN UNINSTALL transaction request. No attach is done as IE domain handles the installation processing internally.
- Attach FMH for some other transid, assumed to be a mirror
- FMH7 indicating the client wants to abend a conversation.
- Connection level PING request/reply
- Conversation level PING request/reply
- Connection status 01, last transmission from client (equivalent to UNINSTALL)
- User data in extended conversation (Link request or SYNCPOINT RU)

All other flows are rejected by CIEP; conversation errors with an FMH7, control errors by closing the socket.

The different flows are distinguished by testing various fields in the flow headers, including the SNA format RH.

Request header settings: Response headers are never sent. All flows have request headers. Errors are returned by sending FMH7 with CEB.

All flows are OIC,RQE1.

The link requests to a long running mirror are packaged as FMH43s but, because they are within a GDS, should not cause the RH FMH bit to be set on.

Direction	Type of flow	Request header flags					
in	CCIN INSTALL FMH5	BB		OIC	CD	RQE1	FMH
out	CCIN INSTALL reply		CEB	OIC		RQE1	
in	CCIN UNINSTALL request	BB	CEB	OIC		RQE1	FMH
in	Mirror FMH5 + link request	BB		OIC	CD	RQE1	FMH
out	Non long-running mirror link reply		CEB	OIC		RQE1	
out	Long-running mirror link reply			OIC	CD	RQE1	
in	Long-running mirror link request			OIC	CD	RQE1	
in	Long-running mirror sync flow			OIC	CD	RQE1	
out	Long-running mirror sync reply		CEB	OIC		RQE1	
out	Conversation failure (FMH7)		CEB	OIC		RQE1	FMH
in	FMH7		CEB	OIC		RQE1	FMH

Mirror task, CPMI

A mirror task is attached by the listener task to handle a particular client conversation. The transaction attach callback module for IE mirrors is DFHIEXM. It sets the IECCB (IP ECI Conversation Control Block) to be the mirror task's facility token and establishes security context for the mirror task, using userid and password sent from the client where required.

The mirror task main program, DFHMIRS, issues the IEIE RECEIVE for the available data, and then performs the same functions as it does for ECI requests received in other environments. It then issues the IEIE SEND to return the output from the linked program to the client. For a conversation marked by the client as 'extended', the mirror then issues another IEIE RECEIVE which causes it to be suspended, waiting for more data. For a non-extended conversation, the mirror terminates after the SEND.

PING

CICS TS 2.2 supports full connection and conversation level PING as designed for the CICS family. This consists of defined flows to allow CICS to determine whether specified connections, or particular conversations on a connection, should still be considered active. CICS TS sends a PING request if the RTIMOUT interval is exceeded when waiting for data from a client:

- Send conversation level PING if the client install indicated this was supported.
- Send connection level PING otherwise.
- If it is a conversation PING that has timed out abend the task after sending a connection level PING to see if the client is still active.

- If a connection level PING times out, uninstall the client.

Notes

1. The socket is full duplex, so SENDs and RECEIVEs can be issued in any order, and asynchronously by different CICS tasks. This is necessary for multiple conversations on the same socket, and means that the CIEP task can issue a SOCK RECEIVE as soon as it has attached the mirror. The SOCK SEND will be done under the mirror task.
2. Sending tasks ENQ on the socket to prevent the data from multiple conversations being interleaved. The ENQ is issued by SO domain.
3. The SO socket token is the second part of the user token but is never required in the CIEP task. The sends and receives issued from CIEP use the socket implicit in the task's state.
4. If the connection is lost or closed by TCP/IP and there are long running mirrors waiting on receives, SO domain is notified, attaches CIEP and returns a bad response on the SO receive issued by CIEP.

Modules

DFHIEP
The initial program for the IP ECI listener transaction, CIEP.

DFHIEXM
The IPECI mirror transaction attach callback module.

Sets the IECCB to be the mirror task's facility token.

Establishes security context for the mirror task, using userid and password sent from client where required.

EXEC interface

The EXEC interface provides the support for application programs containing **EXEC CICS** commands.

Design overview

The relevant parts of the EXEC interface are as follows:

- The initial command processors, DFHEIP or DFHEIG.
  - DFHEIP is called when an **EXEC CICS** command is executed in an AMODE(24) or AMODE(31) user application program.
  - DFHEIG is called when an **EXEC CICS** command is executed in an AMODE(64) user application program.

  A parameter list is passed, in which the first argument (referred to as arg-zero) contains a group code and a function code as the first two bytes.
  - The group code in general indicates the CICS component associated with the command being executed. In subsequent processing, it is this code alone that determines which EXEC processor module is called from DFHEIP or DFHEIG.
  - The function code identifies the actual command being executed.

  DFHEIP and DFHEIG are link-edited with other modules to form the application interface program (DFHAIP) load module.
- A set of EXEC processor modules that the initial command processor calls, which perform the first level of analysis of the command being executed. The processor then calls the appropriate CICS domain to complete the execution of the command.

- A set of EXEC stubs. For COBOL, PL/I, and C, there is one stub for each application language. For assembler language, there is one stub for AMODE(24) or AMODE(31) applications, and one stub for AMODE(64) applications. The appropriate stub must be link-edited at the front of each CICS application program, and provides the mechanism for getting to the correct entry points in the command processor.
- The DFHAPLI module, which is called at the initialization and termination of each application program.

Control blocks

The control blocks associated with the EXEC interface are as follows:

EXEC interface block (EIB) (DSECT name: DFHEIBLK).

Each task in a command-level environment has a control block called the EXEC interface block (EIB) associated with it. The EIB is used for direct communication between command-level programs and CICS.

The EIB contains information that is useful during the execution of an application program, such as the transaction identifier, the time and date (initially when the task is started, and subsequently, if updated by the application program), and the cursor position on a display device. The EIB also contains information that is helpful when a dump is being used to debug a program. DFHEIBLK defines the layout of an EIB, and is included automatically in the application program, giving access to all of the fields in the EIB by name.

A further EIB, known as the "system" EIB, exists for each task. The system EIB has the same format as the "user" (or "application") EIB. It is intended for use mainly by CICS system code. In general, application programs have addressability to the user EIB only, which is a copy taken of the system EIB at appropriate times. However, any service programs translated with the SYSEIB option have addressability to the system EIB also, so that they can issue EXEC CICS commands without causing the user EIB to be updated. (See Defining translator options for further information about the SYSEIB translator option.)

Figure 44 on page 1572 shows the format of an EIB.

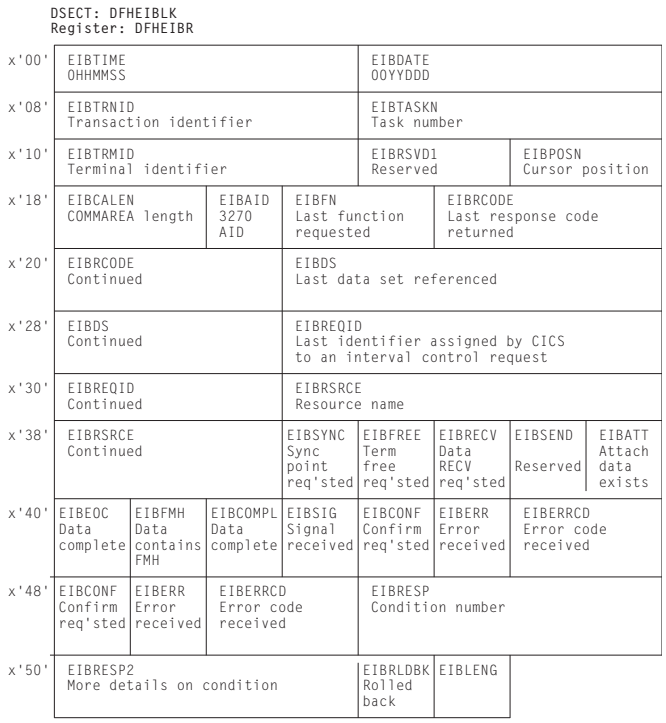
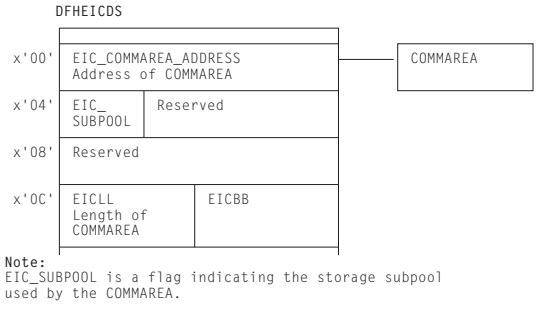


Figure 44. EXEC interface block (EIB)

EXEC interface communication area (DSECT name: DFHEICDS).

The EXEC interface communication area describes the storage that is used to pass the COMMAREA from one command-level transaction to another using an EXEC CICS RETURN command with the TRANSID, COMMAREA, and LENGTH options.

Figure 45 shows the format of the EXEC interface communication area.



Note: EIC_SUBPOOL is a flag indicating the storage subpool used by the COMMAREA.

Figure 45. EXEC interface communication area (EIC)

EXEC interface storage (EIS) (DSECT name: DFHEISDS).

The EXEC interface storage is used by DFHEIP as the interface between the application program and CICS control blocks. It contains a system area used by DFHEIP only. EIS is storage acquired by the DFHAPXM module (part of the transaction manager), along with other task-lifetime storage such as the TCA and both system and user EIBs. There is one EIS per transaction (not per program), and it is addressed by TCAEISA in the TCA.

(See Figure 46.)

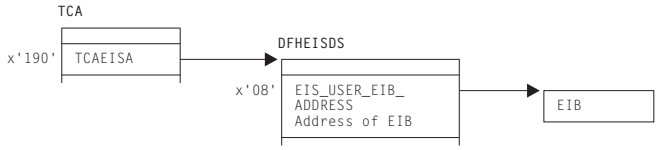


Figure 46. EXEC interface storage (EIS)

Modules

The EXEC interface comprises the following modules:

- The main interface modules (DFHEIP and DFHEIG)
- Prologue and epilogue code for assembler-language programs
- EXEC interface processors
- EXEC stubs

These processor modules support the EXEC CICS commands that are listed in Table 70.

The DFHEIP main interface module also supports EXEC DLI commands. It passes these commands through the external resource manager interface program, DFHERM, on their way to DFHEDP for conversion to standard CALL parameter lists that are acceptable to DL/I.

The following tables list all the EXEC CICS commands and show the class of each command (basic or special), its group and function codes, and the name and language of the associated EXEC interface processor. Table 70 is ordered by command name. Table 71 on page 1578 is ordered by group/function code.

The group and function codes used by the Front End Programming Interface (FEPI) feature are not listed in these tables. However, the EXEC CICS FEPI commands use group codes of 82 (API-type commands) and 84 (SPI-type commands). For details about the EXEC CICS FEPI commands, see The FEPI API commands.

The tables use the following abbreviations:

Class of command

B = basic

S = special

Language of module

A = assembler

O = other

Table 70. EXEC CICS commands ordered by command name

Command	Class	Group/function code	Module DFH...	Language
ABEND	B	0E 0C	EPC	A
ACQUIRE TERMINAL	S	86 02	EIACQ	O
ADDRESS	B	02 02	EEI	A
ADDRESS SET	B	02 10	EEI	A

Table 70. EXEC CICS commands ordered by command name (continued)

Command	Class	Group/function code	Module DFH...	Language
ALLOCATE	B	04 20	ETC	A
ASKTIME	B	10 02	EIIC	O
ASKTIME ABSTIME	B	4A 02	EIDTI	O
ASSIGN	B	02 08	EEI	A
BIF DEEDIT	B	20 02	EBF	A
BUILD ATTACH	B	04 26	ETC	A
CANCEL	B	10 0C	EIIC	O
CHANGE TASK	B	5E 06	EIQSK	O
COLLECT STATISTICS	S	70 08	EIQMS	O
CONNECT PROCESS	B	04 32	ETC	A
CONVERSE	B	04 06	ETC	A
CREATE CONNECTION	S	30 0E	EICRE	O
CREATE FILE	S	30 14	EICRE	O
CREATE JOURNALMODEL	S	30 1E	EICRE	O
CREATE LSRPOOL	S	30 16	EICRE	O
CREATE MAPSET	S	30 04	EICRE	O
CREATE PARTITIONSET	S	30 06	EICRE	O
CREATE PARTNER	S	30 18	EICRE	O
CREATE PROFILE	S	30 0A	EICRE	O
CREATE PROGRAM	S	30 02	EICRE	O
CREATE SESSIONS	S	30 12	EICRE	O
CREATE TDQUEUE	S	30 1C	EICRE	O
CREATE TERMINAL	S	30 10	EICRE	O
CREATE TRANCLASS	S	30 1A	EICRE	O
CREATE TRANSACTION	S	30 08	EICRE	O
CREATE TYPETERM	S	30 0C	EICRE	O
DELAY	B	10 04	EIIC	O
DELETE	B	06 08	EIFC	O
DELETEQ TD	B	08 06	ETD	A
DELETEQ TS	B	0A 06	ETS	A
DEQ	B	12 06	EKC	A
DISCARD AUTINSTMODEL	S	42 10	EIQTM	O
DISCARD FILE	S	4C 10	EIQDS	O
DISCARD JOURNALMODEL	S	92 10	EIQSL	O
DISCARD JOURNALNAME	S	60 10	EIQSJ	O
DISCARD PARTNER	S	44 10	EIQPN	O
DISCARD PROFILE	S	46 10	EIQPF	O
DISCARD PROGRAM	S	4E 10	EIQSP	O
DISCARD TRANSACTION	S	50 10	EIQSX	O
DISABLE	B	22 04	UEM	A
DUMP	B	1C 02	EDC	A
DUMP SYSTEM	B	7E 04	EDCP	O
DUMP TRANSACTION	B	7E 02	EDCP	O
ENABLE	B	22 02	UEM	A
ENDBR	B	06 12	EIFC	O
ENQ	B	12 04	EKC	A
ENTER TRACEID	B	1A 04	ETR	A
ENTER TRACENUM	B	48 02	ETRX	O

Table 70. EXEC CICS commands ordered by command name (continued)

Command	Class	Group/function code	Module DFH...	Language
EXTRACT ATTACH	B	04 28	ETC	A
EXTRACT ATTRIBUTES	B	04 3E	ETC	A
EXTRACT EXIT	B	22 06	UEM	A
EXTRACT LOGONMSG	B	04 3C	ETC	A
EXTRACT PROCESS	B	04 2E	ETC	A
EXTRACT TCT	B	04 2A	ETC	A
FORMATTIME	B	4A 04	EIDTI	O
FREE	B	04 22	ETC	A
FREEMAIN	B	0C 04	EISC	O
FREEMAIN64	B	0C 14	EISC	O
GDS ALLOCATE	B	24 02	EGL	A
GDS ASSIGN	B	24 04	EGL	A
GDS CONNECT PROCESS	B	24 0C	EGL	A
GDS EXTRACT ATTRIBUTES	B	24 1C	EGL	A
GDS EXTRACT PROCESS	B	24 06	EGL	A
GDS FREE	B	24 08	EGL	A
GDS ISSUE ABEND	B	24 0A	EGL	A
GDS ISSUE CONFIRMATION	B	24 0E	EGL	A
GDS ISSUE ERROR	B	24 10	EGL	A
GDS ISSUE PREPARE	B	24 1A	EGL	A
GDS ISSUE SIGNAL	B	24 12	EGL	A
GDS RECEIVE	B	24 14	EGL	A
GDS SEND	B	24 16	EGL	A
GDS WAIT	B	24 18	EGL	A
GETMAIN	B	0C 02	EISC	O
GETMAIN64	B	0C 12	EISC	O
HANDLE ABEND	B	0E 0E	EPC	A
HANDLE AID	B	02 06	EEI	A
HANDLE CONDITION	B	02 04	EEI	A
IGNORE CONDITION	B	02 0A	EEI	A
INQUIRE AUTINSTMODEL	S	42 02	EIQTM	O
INQUIRE AUTOINSTALL	S	68 12	EIQVT	O
INQUIRE CONNECTION	S	58 02	EIQSC	O
INQUIRE DCE	S	8E 02	EIQDE	O
INQUIRE DSNAME	S	7A 02	EIQDN	O
INQUIRE DUMPDS	S	66 02	EIQDU	O
INQUIRE EXITPROGRAM	S	88 02	EIQUE	O
INQUIRE FILE	S	4C 02	EIQDS	O
INQUIRE IRC	S	6E 02	EIQIR	O
INQUIRE JOURNALMODEL	S	92 02	EIQSL	O
INQUIRE JOURNALNAME	S	60 12	EIQSJ	O
INQUIRE JOURNALNUM	S	60 02	EIQSJ	O
INQUIRE MODENAME	S	5A 02	EIQSM	O
INQUIRE MONITOR	S	70 12	EIQMS	O
INQUIRE NETNAME	S	52 06	EIQST	O
INQUIRE PARTNER	S	44 02	EIQPN	O
INQUIRE PROFILE	S	46 02	EIQPF	O

Table 70. EXEC CICS commands ordered by command name (continued)

Command	Class	Group/function code	Module DFH...	Language
INQUIRE PROGRAM	S	4E 02	EIQSP	O
INQUIRE REQID	S	8A 02	EIQRQ	O
INQUIRE STATISTICS	S	70 02	EIQMS	O
INQUIRE STREAMNAME	S	92 12	EIQSL	O
INQUIRE SYSDUMPCODE	S	66 22	EIQDU	O
INQUIRE SYSTEM	S	54 02	EIQSA	O
INQUIRE TASK	S	5E 02	EIQSK	O
INQUIRE TCLASS	S	5E 12	EIQSK	O
INQUIRE TDQUEUE	S	5C 02	EIQSQ	O
INQUIRE TERMINAL	S	52 02	EIQST	O
INQUIRE TRACEDEST	S	78 02	EIQTR	O
INQUIRE TRACEFLAG	S	78 12	EIQTR	O
INQUIRE TRACETYPE	S	78 22	EIQTR	O
INQUIRE TRANDUMPCODE	S	66 12	EIQDU	O
INQUIRE TRANSACTION	S	50 02	EIQSX	O
INQUIRE TSQUEUE	S	0A 08	EIQTS	O
INQUIRE VTAM [1]	S	68 02	EIQVT	O
ISSUE ABEND	B	04 30	ETC	A
ISSUE ABORT	B	1E 08	EDI	A
ISSUE ADD	B	1E 02	EDI	A
ISSUE CONFIRMATION	B	04 34	ETC	A
ISSUE COPY	B	04 0A	ETC	A
ISSUE DISCONNECT	B	04 14	ETC	A
ISSUE END	B	1E 0C	EDI	A
ISSUE ENDFILE	B	04 1A	ETC	A
ISSUE ENDOUTPUT	B	04 16	ETC	A
ISSUE EODS	B	04 08	ETC	A
ISSUE ERASE	B	1E 04	EDI	A
ISSUE ERASEAUP	B	04 18	ETC	A
ISSUE ERROR	B	04 36	ETC	A
ISSUE LOAD	B	04 0E	ETC	A
ISSUE NOTE	B	1E 10	EDI	A
ISSUE PASS	B	04 3A	ETC	A
ISSUE PREPARE	B	04 38	ETC	A
ISSUE PRINT	B	04 1C	ETC	A
ISSUE QUERY	B	1E 0A	EDI	A
ISSUE RECEIVE	B	1E 0E	EDI	A
ISSUE REPLACE	B	1E 06	EDI	A
ISSUE RESET	B	04 12	ETC	A
ISSUE SEND	B	1E 14	EDI	A
ISSUE SIGNAL	B	04 1E	ETC	A
ISSUE WAIT	B	1E 12	EDI	A
LINK	B	0E 02	EPC	A
LOAD	B	0E 06	EPC	A
MONITOR	B	48 04	ETRX	O
PERFORM RESETTIME	S	72 02	EIPRT	O
PERFORM SECURITY	S	64 02	EIPSE	O
PERFORM SHUTDOWN	S	76 02	EIPSH	O
PERFORM STATISTICS	S	70 06	EIQMS	O
POINT	B	04 24	ETC	A

Table 70. EXEC CICS commands ordered by command name (continued)

Command	Class	Group/function code	Module DFH...	Language
POP	B	02 0E	EEI	A
POST	B	10 06	EIIC	O
PURGE MESSAGE	B	18 0A	EMS	A
PUSH	B	02 0C	EEI	A
QUERY SECURITY	B	6A 02	ESE	O
READ	B	06 02	EIFC	O
READNEXT	B	06 0E	EIFC	O
READPREV	B	06 10	EIFC	O
READQ TD	B	08 04	ETD	A
READQ TS	B	0A 04	ETS	A
RECEIVE	B	04 02	ETC	A
RECEIVE MAP	B	18 02	EMS	A
RECEIVE PARTN	B	18 0E	EMS	A
RELEASE	B	0E 0A	EPC	A
RESETBR	B	06 14	EIFC	O
RESYNC	B	16 04	ESP	A
RETRIEVE	B	10 0A	EIIC	O
RETURN	B	0E 08	EPC	A
REWRITE	B	06 06	EIFC	O
ROUTE	B	18 0C	EMS	A
SEND	B	04 04	ETC	A
SEND CONTROL	B	18 12	EMS	A
SEND MAP	B	18 04	EMS	A
SEND PAGE	B	18 08	EMS	A
SEND PARTNSET	B	18 10	EMS	A
SEND TEXT	B	18 06	EMS	A
SET AUTOINSTALL	S	68 14	EIQVT	O
SET CONNECTION	S	58 04	EIQSC	O
SET DCE	S	8E 04	EIQDE	O
SET DSNAME	S	7A 04	EIQDN	O
SET DUMPDS	S	66 04	EIQDU	O
SET FILE	S	4C 04	EIQDS	O
SET IRC	S	6E 04	EIQIR	O
SET JOURNALNAME	S	60 14	EIQSJ	O
SET JOURNALNUM	S	60 04	EIQSJ	O
SET MODENAME	S	5A 04	EIQSM	O
SET MONITOR	S	70 14	EIQMS	O
SET NETNAME	S	52 08	EIQST	O
SET PROGRAM	S	4E 04	EIQSP	O
SET STATISTICS	S	70 04	EIQMS	O
SET SYSDUMPCODE	S	66 24	EIQDU	O
SET SYSTEM	S	54 04	EIQSA	O
SET TASK	S	5E 04	EIQSK	O
SET TCLASS	S	5E 14	EIQSK	O
SET TDQUEUE	S	5C 04	EIQSQ	O
SET TERMINAL	S	52 04	EIQST	O
SET TRACEDEST	S	78 04	EIQTR	O
SET TRACEFLAG	S	78 14	EIQTR	O
SET TRACETYPE	S	78 24	EIQTR	O
SET TRANDUMPCODE	S	66 14	EIQDU	O
SET TRANSACTION	S	50 04	EIQSX	O

Table 70. EXEC CICS commands ordered by command name (continued)

Command	Class	Group/function code	Module DFH...	Language
SET VTAM [1]	S	68 04	EIQVT	O
SIGNOFF	B	74 04	ESN	O
SIGNON	B	74 02	ESN	O
SPOOLCLOSE	B	56 10	EPS	O
SPOOLOPEN	B	56 02	EPS	O
SPOOLREAD	B	56 04	EPS	O
SPOOLWRITE	B	56 06	EPS	O
START	B	10 08	EIIC	O
STARTBR	B	06 0C	EIFC	O
SUSPEND	B	12 08	EKC	A
SYNCPOINT	B	16 02	ESP	A
TRACE	B	1A 02	ETR	A
UNLOCK	B	06 0A	EIFC	O
WAIT CONVID	B	04 2C	ETC	A
WAIT EVENT	B	12 02	EKC	A
WAIT EXTERNAL	B	5E 22	EIQSK	O
WAIT JOURNALNAME	B	14 08	EJC	A
WAIT JOURNALNUM	B	14 04	EJC	A
WAIT SIGNAL	B	04 10	ETC	A
WAIT TERMINAL	B	04 0C	ETC	A
WAITCICS	B	5E 32	EIQSK	O
WRITE FILE	B	06 04	EIFC	O
WRITE JOURNALNAME	B	14 06	EJC	A
WRITE JOURNALNUM	B	14 02	EJC	A
WRITE OPERATOR	B	6C 02	EOP	O
WRITEQ TD	B	08 02	ETD	A
WRITEQ TS	B	0A 02	ETS	A
XCTL	B	0E 04	EPC	A

Table 71. EXEC CICS commands ordered by group/function code

Command	Class	Group/function code	Module DFH...	Language
ADDRESS	B	02 02	EEI	A
HANDLE CONDITION	B	02 04	EEI	A
HANDLE AID	B	02 06	EEI	A
ASSIGN	B	02 08	EEI	A
IGNORE CONDITION	B	02 0A	EEI	A
PUSH	B	02 0C	EEI	A
POP	B	02 0E	EEI	A
ADDRESS SET	B	02 10	EEI	A
RECEIVE	B	04 02	ETC	A
SEND	B	04 04	ETC	A
CONVERSE	B	04 06	ETC	A
ISSUE EODS	B	04 08	ETC	A
ISSUE COPY	B	04 0A	ETC	A
WAIT TERMINAL	B	04 0C	ETC	A
ISSUE LOAD	B	04 0E	ETC	A
WAIT SIGNAL	B	04 10	ETC	A
ISSUE RESET	B	04 12	ETC	A
ISSUE DISCONNECT	B	04 14	ETC	A

Table 71. EXEC CICS commands ordered by group/function code (continued)

Command	Class	Group/function code	Module DFH...	Language
ISSUE ENDOUTPUT	B	04 16	ETC	A
ISSUE ERASEAUP	B	04 18	ETC	A
ISSUE ENDFILE	B	04 1A	ETC	A
ISSUE PRINT	B	04 1C	ETC	A
ISSUE SIGNAL	B	04 1E	ETC	A
ALLOCATE	B	04 20	ETC	A
FREE	B	04 22	ETC	A
POINT	B	04 24	ETC	A
BUILD ATTACH	B	04 26	ETC	A
EXTRACT ATTACH	B	04 28	ETC	A
EXTRACT TCT	B	04 2A	ETC	A
WAIT CONVID	B	04 2C	ETC	A
EXTRACT PROCESS	B	04 2E	ETC	A
ISSUE ABEND	B	04 30	ETC	A
CONNECT PROCESS	B	04 32	ETC	A
ISSUE CONFIRMATION	B	04 34	ETC	A
ISSUE ERROR	B	04 36	ETC	A
ISSUE PREPARE	B	04 38	ETC	A
ISSUE PASS	B	04 3A	ETC	A
EXTRACT LOGONMSG	B	04 3C	ETC	A
EXTRACT ATTRIBUTES	B	04 3E	ETC	A
READ	B	06 02	EIFC	O
WRITE FILE	B	06 04	EIFC	O
REWRITE	B	06 06	EIFC	O
DELETE	B	06 08	EIFC	O
UNLOCK	B	06 0A	EIFC	O
STARTBR	B	06 0C	EIFC	O
READNEXT	B	06 0E	EIFC	O
READPREV	B	06 10	EIFC	O
ENDBR	B	06 12	EIFC	O
RESETBR	B	06 14	EIFC	O
WRITEQ TD	B	08 02	ETD	A
READQ TD	B	08 04	ETD	A
DELETEQ TD	B	08 06	ETD	A
WRITEQ TS	B	0A 02	ETS	A
READQ TS	B	0A 04	ETS	A
DELETEQ TS	B	0A 06	ETS	A
INQUIRE TSQUEUE	S	0A 08	EIQTS	O
GETMAIN	B	0C 02	EISC	O
FREEMAIN	B	0C 04	EISC	O
GETMAIN64	B	0C 12	EISC	O
FREEMAIN64	B	0C 14	EISC	O
LINK	B	0E 02	EPC	A
XCTL	B	0E 04	EPC	A
LOAD	B	0E 06	EPC	A
RETURN	B	0E 08	EPC	A
RELEASE	B	0E 0A	EPC	A
ABEND	B	0E 0C	EPC	A
HANDLE ABEND	B	0E 0E	EPC	A
ASKTIME	B	10 02	EIIC	O
DELAY	B	10 04	EIIC	O

Table 71. EXEC CICS commands ordered by group/function code (continued)

Command	Class	Group/function code	Module DFH...	Language
POST	B	10 06	EIIC	O
START	B	10 08	EIIC	O
RETRIEVE	B	10 0A	EIIC	O
CANCEL	B	10 0C	EIIC	O
WAIT EVENT	B	12 02	EKC	A
ENQ	B	12 04	EKC	A
DEQ	B	12 06	EKC	A
SUSPEND	B	12 08	EKC	A
WRITE JOURNALNUM	B	14 02	EJC	A
WAIT JOURNALNUM	B	14 04	EJC	A
SYNCPOINT	B	16 02	ESP	A
RESYNC	B	16 04	ESP	A
RECEIVE MAP	B	18 02	EMS	A
SEND MAP	B	18 04	EMS	A
SEND TEXT	B	18 06	EMS	A
SEND PAGE	B	18 08	EMS	A
PURGE MESSAGE	B	18 0A	EMS	A
ROUTE	B	18 0C	EMS	A
RECEIVE PARTN	B	18 0E	EMS	A
SEND PARTNSET	B	18 10	EMS	A
SEND CONTROL	B	18 12	EMS	A
TRACE	B	1A 02	ETR	A
ENTER TRACEID	B	1A 04	ETR	A
DUMP	B	1C 02	EDC	A
ISSUE ADD	B	1E 02	EDI	A
ISSUE ERASE	B	1E 04	EDI	A
ISSUE REPLACE	B	1E 06	EDI	A
ISSUE ABORT	B	1E 08	EDI	A
ISSUE QUERY	B	1E 0A	EDI	A
ISSUE END	B	1E 0C	EDI	A
ISSUE RECEIVE	B	1E 0E	EDI	A
ISSUE NOTE	B	1E 10	EDI	A
ISSUE WAIT	B	1E 12	EDI	A
ISSUE SEND	B	1E 14	EDI	A
BIF DEEDIT	B	20 02	EBF	A
ENABLE	B	22 02	UEM	A
DISABLE	B	22 04	UEM	A
EXTRACT EXIT	B	22 06	UEM	A
GDS ALLOCATE	B	24 02	EGL	A
GDS ASSIGN	B	24 04	EGL	A
GDS EXTRACT PROCESS	B	24 06	EGL	A
GDS FREE	B	24 08	EGL	A
GDS ISSUE ABEND	B	24 0A	EGL	A
GDS CONNECT PROCESS	B	24 0C	EGL	A
GDS ISSUE CONFIRMATION	B	24 0E	EGL	A
GDS ISSUE ERROR	B	24 10	EGL	A
GDS ISSUE SIGNAL	B	24 12	EGL	A
GDS RECEIVE	B	24 14	EGL	A
GDS SEND	B	24 16	EGL	A
GDS WAIT	B	24 18	EGL	A

Table 71. EXEC CICS commands ordered by group/function code (continued)

Command	Class	Group/function code	Module DFH...	Language
GDS ISSUE PREPARE	B	24 1A	EGL	A
GDS EXTRACT ATTRIBUTES	B	24 1C	EGL	A
CREATE PROGRAM	S	30 02	EICRE	O
CREATE MAPSET	S	30 04	EICRE	O
CREATE PARTITIONSET	S	30 06	EICRE	O
CREATE TRANSACTION	S	30 08	EICRE	O
CREATE PROFILE	S	30 0A	EICRE	O
CREATE TYPETERM	S	30 0C	EICRE	O
CREATE CONNECTION	S	30 0E	EICRE	O
CREATE TERMINAL	S	30 10	EICRE	O
CREATE SESSIONS	S	30 12	EICRE	O
CREATE FILE	S	30 14	EICRE	O
CREATE LSRPOOL	S	30 16	EICRE	O
CREATE PARTNER	S	30 18	EICRE	O
CREATE TRANCLASS	S	30 1A	EICRE	O
CREATE TDQUEUE	S	30 1C	EICRE	O
CREATE JOURNALMODEL	S	30 1E	EICRE	O
INQUIRE AUTINSTMODEL	S	42 02	EIQTM	O
DISCARD AUTINSTMODEL	S	42 10	EIQTM	O
INQUIRE PARTNER	S	44 02	EIQPN	O
DISCARD PARTNER	S	44 10	EIQPN	O
INQUIRE PROFILE	S	46 02	EIQPF	O
DISCARD PROFILE	S	46 10	EIQPF	O
ENTER TRACENUM	B	48 02	ETRX	O
MONITOR	B	48 04	ETRX	O
ASKTIME ABSTIME	B	4A 02	EIDTI	O
FORMATTIME	B	4A 04	EIDTI	O
INQUIRE FILE	S	4C 02	EIQDS	O
SET FILE	S	4C 04	EIQDS	O
DISCARD FILE	S	4C 10	EIQDS	O
INQUIRE PROGRAM	S	4E 02	EIQSP	O
SET PROGRAM	S	4E 04	EIQSP	O
DISCARD PROGRAM	S	4E 10	EIQSP	O
INQUIRE TRANSACTION	S	50 02	EIQSX	O
SET TRANSACTION	S	50 04	EIQSX	O
DISCARD TRANSACTION	S	50 10	EIQSX	O
INQUIRE TERMINAL	S	52 02	EIQST	O
SET TERMINAL	S	52 04	EIQST	O
INQUIRE NETNAME	S	52 06	EIQST	O
SET NETNAME	S	52 08	EIQST	O
INQUIRE SYSTEM	S	54 02	EIQSA	O
SET SYSTEM	S	54 04	EIQSA	O
SPOOLOPEN	B	56 02	EPS	O
SPOOLREAD	B	56 04	EPS	O
SPOOLWRITE	B	56 06	EPS	O
SPOOLCLOSE	B	56 10	EPS	O
INQUIRE CONNECTION	S	58 02	EIQSC	O
SET CONNECTION	S	58 04	EIQSC	O
INQUIRE MODENAME	S	5A 02	EIQSM	O
SET MODENAME	S	5A 04	EIQSM	O

Table 71. EXEC CICS commands ordered by group/function code (continued)

Command	Class	Group/function code	Module DFH...	Language
INQUIRE TDQUEUE	S	5C 02	EIQSQ	O
SET TDQUEUE	S	5C 04	EIQSQ	O
INQUIRE TASK	S	5E 02	EIQSK	O
SET TASK	S	5E 04	EIQSK	O
CHANGE TASK	B	5E 06	EIQSK	O
INQUIRE TCLASS	S	5E 12	EIQSK	O
SET TCLASS	S	5E 14	EIQSK	O
WAIT EXTERNAL	B	5E 22	EIQSK	O
WAITCICS	B	5E 32	EIQSK	O
INQUIRE JOURNALNUM	S	60 02	EIQSJ	O
SET JOURNALNUM	S	60 04	EIQSJ	O
INQUIRE JOURNALNAME	S	60 12	EIQSJ	O
SET JOURNALNAME	S	60 14	EIQSJ	O
PERFORM SECURITY	S	64 02	EIPSE	O
INQUIRE DUMPDS	S	66 02	EIQDU	O
SET DUMPDS	S	66 04	EIQDU	O
INQUIRE TRANDUMPCODE	S	66 12	EIQDU	O
SET TRANDUMPCODE	S	66 14	EIQDU	O
INQUIRE SYSDUMPCODE	S	66 22	EIQDU	O
SET SYSDUMPCODE	S	66 24	EIQDU	O
INQUIRE VTAM [1]	S	68 02	EIQVT	O
SET VTAM [1]	S	68 04	EIQVT	O
INQUIRE AUTOINSTALL	S	68 12	EIQVT	O
SET AUTOINSTALL	S	68 14	EIQVT	O
QUERY SECURITY	B	6A 02	ESE	O
WRITE OPERATOR	B	6C 02	EOP	O
CICSMESSAGE [2]	S	6C 12	EOP	O
INQUIRE IRC	S	6E 02	EIQIR	O
SET IRC	S	6E 04	EIQIR	O
INQUIRE STATISTICS	S	70 02	EIQMS	O
SET STATISTICS	S	70 04	EIQMS	O
PERFORM STATISTICS	S	70 06	EIQMS	O
COLLECT STATISTICS	S	70 08	EIQMS	O
INQUIRE MONITOR	S	70 12	EIQMS	O
SET MONITOR	S	70 14	EIQMS	O
PERFORM RESETTIME	S	72 02	EIPRT	O
SIGNON	B	74 02	ESN	O
SIGNOFF	B	74 04	ESN	O
PERFORM SHUTDOWN	S	76 02	EIPSH	O
INQUIRE TRACEDEST	S	78 02	EIQTR	O
SET TRACEDEST	S	78 04	EIQTR	O
INQUIRE TRACEFLAG	S	78 12	EIQTR	O
SET TRACEFLAG	S	78 14	EIQTR	O
INQUIRE TRACETYPE	S	78 22	EIQTR	O
SET TRACETYPE	S	78 24	EIQTR	O
INQUIRE DSNAME	S	7A 02	EIQDN	O
SET DSNAME	S	7A 04	EIQDN	O
DUMP TRANSACTION	B	7E 02	EDCP	O
DUMP SYSTEM	B	7E 04	EDCP	O
INQUIRE JOURNALMODEL	S	92 02	EIQSL	O

Table 71. EXEC CICS commands ordered by group/function code (continued)

Command	Class	Group/function code	Module DFH...	Language
INQUIRE STREAMNAME	S	92 12	EIQSL	O

1. VTAM is the previous name for z/OS Communications Server.
2. This command is for CICS internal use only.

Exits

The following global user exit points are provided in DFHEIP:

Exit programs and the CICS storage protection facility

User-replaceable programs and the storage protection facility

EXEC interface program exits XEIIN, XEIOUT, XEISPIN, and XEISPOUT

For further information, see .

Trace

The following point IDs are provided for DFHEIG:

- AP A500 (entry), for which the trace level is EI 1.
- AP A501 (exit), for which the trace level is EI 1.

The following point ID is provided for DFHEIP:

- AP 00E1, for which the trace level is EI 1.

The following point IDs are provided for DFHEISR:

- AP E110 (entry), for which the trace level is EI 2.
- AP E111 (exit), for which the trace level is EI 2.

Trace entries are made before and after the execution of a command by its EXEC interface processor module.

Execution diagnostic facility (EDF)

When you use the CICS command-level programming interface, you can use the execution diagnostic facility (EDF) to step through the CICS commands of an application program. This program can be part of a local or remote transaction. At each step, you can check the validity of each command and make temporary modifications to the program.

Design overview

Application programmers can use EDF to test a command-level application program online without making any modifications to the source program or the program preparation procedure. EDF intercepts execution of the application program at certain points and displays relevant information about the program at these points.

The EDF can be called at the following points in the EXEC interface program (DFHEIP or DFHEIG):

- When program initialization has completed, just before control is passed to the application entry point

- During program termination, just after control is been received from the application
- Before a normal EXEC command is passed to its processor module
- When a normal EXEC command returns to DFHEIP or DFHEIG

Modules

Module	Function
DFHEDFBR	You can use the temporary-storage browse transaction (CEBR) to browse, copy, or delete items in a queue. CEBR calls DFHEDFBR to run the required action.
DFHEDFD	EDF display program that provides the user interaction.
DFHEDFM	EDF map set that consists of three maps: DFHEDFM, DFHEDFN, and DFHEDFP.
DFHEDFP	EDF control program that provides the CEDF transaction for starting EDF. It controls the debugging task and sets debug mode on or off.
DFHEDFR	EDF response table that is used by DFHEDFD to interpret the responses obtained by EXEC commands.
DFHEDFX	EDF task switch program that attaches the debugging task, providing all required information about the status of the user task. It also suspends the user task until the debugging task allows it to resume.

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

External CICS interface

The external CICS interface (EXCI) is an integral part of CICS Transaction Server for z/OS. The function enables non-CICS application programs (*client programs*) running in MVS to call programs (*server programs*) running in a CICS Transaction Server for z/OS region, and to pass and receive data by means of a communications area or by means of channels and containers.

Design overview

This section provides an overview of the design of the external CICS interface (EXCI).

The external CICS interface is an application programming interface that enables a non-CICS program (a *client program*) running in MVS to call a program (a *server program*) running in a CICS region, and to pass and receive data by means of a communications area or by means of channels and containers. The CICS application program is invoked as if linked-to by another CICS application program.

The EXCI programming interface allows a user to allocate and open sessions (or *pipes*) to a CICS region, and to pass distributed program link (DPL) requests over

them. The multiregion operation (MRO) facility of CICS interregion communication (IRC) facility supports these requests, and each pipe. (A pipe is a one-way communication path between a sending process and a receiving process. In an external CICS interface implementation, each pipe maps onto one MRO session, where the client program represents the sending process and the CICS server region represents the receiving process.)

Unless the CICS region is running in a sysplex under MVS/ESA 5.1 and therefore able to use cross-system MRO (XCF/MRO), the client program and the CICS server region (the region where the server program runs or is defined) must be in the same MVS image. Although the external CICS interface does not support the cross-memory access method, it can use the XCF access method provided by XCF/MRO. For information about XCF/MRO, see Getting started with intercommunication.

A client program that uses the external CICS interface can operate multiple sessions for different users (either under the same or separate TCBs) all coexisting in the same MVS address space without knowledge of, or interference from, each other.

Where a client program attaches another client program, the attached program runs under its own TCB.

The programming interfaces

The external CICS interface provides two forms of programming interface: the EXCI CALL interface and the EXEC CICS interface.

The EXCI CALL interface

This interface consists of six commands that allow you to:

- Allocate and open sessions to a CICS system from non-CICS programs running under MVS
- Issue DPL requests on these sessions from the non-CICS programs
- Close and deallocate the sessions on completion of the DPL requests.

The six EXCI commands are:

1. Initialize_User
2. Allocate_Pipe
3. Open_Pipe
4. DPL call
5. Close_Pipe
6. Deallocate_Pipe

The processing of an EXCI CALL-level command is shown in Figure 47 on page 1586.

The EXEC CICS interface

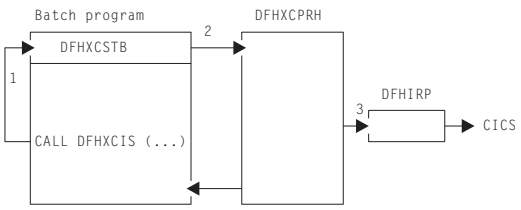
The external CICS interface provides a single, composite command–EXEC CICS LINK PROGRAM– that performs all six commands of the EXCI CALL interface in one invocation. The processing of an EXEC CICS LINK command is shown in Figure 48 on page 1586.

This command takes the same form as the distributed program link command of the CICS command-level application programming interface.

API restrictions for server programs

A CICS server program invoked by an external CICS interface request is restricted to the DPL subset of the CICS application programming interface. This subset (the DPL subset) of the API commands is the same as for a CICS-to-CICS server program.

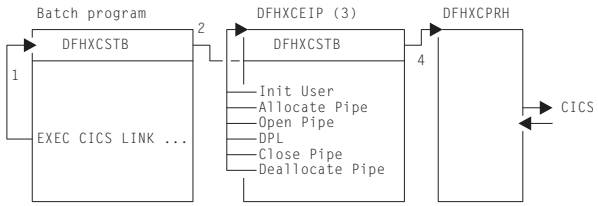
For details about the DPL subset for server programs, see the Distributed Program Link (DPL).



Note:

1. An EXCI CALL API request is issued, and invokes the DFHXCIS entry point in the EXCI stub, DFHXCSTB.
2. DFHXCSTB locates DFHXCPRH, and invokes it to process the EXCI request. If DFHXCPRH is not found, DFHXCSTB loads DFHXCPRH before invoking it.
3. DFHXCPRH sets up the control blocks needed for the EXCI request. For a DPL request, DFHXCPRH invokes DFHIRP to pass control to CICS.

Figure 47. External CICS interface, CALL-level API



Note:

1. An EXCI EXEC API request is issued, and invokes the DFHXCEI entry point in the EXCI stub, DFHXCSTB.
2. DFHXCSTB locates DFHXCEIP, and invokes it to process the EXCI request. If DFHXCEIP is not found, DFHXCSTB loads DFHXCEIP before invoking it.
3. DFHXCEIP converts the EXCI EXEC-level request into a series of EXCI CALL-level requests.
4. The CALL-level requests result in calls to the EXCI stub, DFHXCSTB (as in Figure 47).

Figure 48. External CICS interface, EXEC-level API

Modules

Module	Function
DFHXCALL	EXEC-level API macro. Invoked by the CICS translator when processing EXCI EXEC-level requests.
DFHXCDMP	dump services. Calls the CICS SVC to issue SDUMP macro requests, to take an SDUMP of the EXCI address space.
DFHXCSTB	stub link-edited with applications that want to use EXCI.
DFHXCEIP	EXEC-level API handler. The main EXCI module that processes EXCI EXEC-level requests.
DFHXCO	options macro for generating the DFHXCOPT options table.
DFHXCOPT	options table to customize the EXCI environment.

Module	Function
DFHXCPLD	Assembler-language parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.
DFHXCPLH	C parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.
DFHXCPLL	PL/I parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.
DFHXCPLO	COBOL parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.
DFHXCPRH	program request handler The main EXCI module that processes EXCI CALL-level requests.
DFHXCRCD	Assembler-language return code definitions. Copybook defining the return codes for use with the EXCI APIs.
DFHXCRCH	C return code definitions. Copybook defining the return codes for use with the EXCI APIs.
DFHXCRCL	PL/I return code definitions. Copybook defining the return codes for use with the EXCI APIs.
DFHXCRCO	COBOL return code definitions. Copybook defining the return codes for use with the EXCI APIs.
DFHXCSVC	SVC services. Invoked by the CICS SVC to issue an SDUMP macro to take an SDUMP of the EXCI address space.
DFHXCTAB	language table. Copybook defining the syntax of the EXCI EXEC language for use by the CICS translator.
DFHXCTRA	global trap program. The EXCI equivalent of the DFHTRAP module, providing the service with ability to collect extra diagnostic information.
DFHXCTRD	local trap parameter list definition. Defines the parameter list passed to DFHXCTRA and all EXCI trace points used by DFHXCTRA.
DFHXCTRP	trace services. Writes EXCI trace entries to the EXCI internal trace table.
DFHXCTRI	trace initialization. Initializes EXCI trace services.
DFHXCURM	User-replaceable program that allows the user to modify the applid of the CICS region to which the EXCI request is to be issued.

Exits

There are no exit points for the EXCI.

Trace

The EXCI has its own internal trace table in the EXCI address space where the client program is running. EXCI trace entries can also be written to the MVS GTF trace data set.

EXCI trace point IDs are EXxxxx, with a trace level of 1, 2, or Exc.

For more information about EXCI tracing, see Problem determination for the external CICS interface in Troubleshooting.

Field engineering program

The field engineering program (DFHFEP) is a CICS system service function primarily designed for an IBM field engineer to use when installing new terminals. When CICS is running, this program (invoked by the CSFE transaction) transmits a set of characters to the requesting terminal. In addition, the program can be used to echo a message; that is, it repeats exactly what is keyed at the terminal.

This program also supports some general debugging functions.

Design overview

When used for testing terminals, DFHFEP first prepares for device-dependent conditions. It then issues a storage control FREEMAIN, followed by a GETMAIN for storage for the ENTER message, which it writes using terminal control WRITE, READ, and WAIT macros. Finally, if **print** was requested, the character set is printed; if **end** was requested, the completion message is issued; otherwise the input is echoed.

DFHFEP performs all the requests made by the CSFE transaction. In addition to the terminal test function, CSFE can request the activation or deactivation of:

- System spooling interface trace
- Terminal builder trace
- Storage freeze
- Storage violation trap
- Global trap/trace exit.

See for details of the command syntax and functions provided.

Modules

DFHFEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

File control

File control provides a facility for accessing data sets, files, and data tables, using keyed or relative-byte-address (RBA) access through the virtual storage access method (VSAM), the basic direct access method (BDAM), shared data table services and the coupling facility data tables server.

VSAM data sets can be accessed in either RLS or non-RLS mode. RLS mode allows sharing of data sets across a parallel sysplex. File control allows updates, additions, deletions, random retrieval, and sequential retrieval (browsing) of logical data in the data sets. If VSAM is used, access to logical data can be through a VSAM alternate index path, as well as through the base data set.

File control reads from, and writes to, user-defined data sets and data tables, gathers statistics, and acquires dynamic storage for I/O operations. File control uses control information defined by the user in FILE resource definitions, or for

BDAM files, DFHFCT macro resource definitions. The control information for all file types is held by CICS in the internal file control table (FCT). This table describes the physical characteristics of all the data sets, and any logical relationships that may exist between them.

Design overview

File control provides the following services and features:

- Random record retrieval
- Random record update
- Random record addition
- Random record deletion (VSAM only)
- Sequential record retrieval
- BDAM deblocking
- Enabling and disabling of files, making them accessible to applications
- Opening and closing of files for the access method
- Exclusive control of records during update operations
- Mass record insertion (VSAM only)
- Automatic journaling and logging.

Deblocking services for BDAM data sets

CICS provides deblocking of logical records on a direct-access (BDAM) data set. This service is provided for both fixed-length and variable-length records. The data set must have been created according to standard operating system record-formatting conventions.

Concurrency control

Protection is provided against the concurrent updating (adding, deleting or changing) of a data set record by two or more transactions (or strictly speaking, two or more units of work; a transaction may optionally consist of a sequence of units of work). This protection is in most cases achieved using locking. If a second unit of work attempts to update a record which has been locked by another unit of work, the second unit of work is normally queued until the first releases its lock. If the lock has been converted into a retained lock (this is done if a syncpoint failure occurs) then the second unit of work gets an error response rather than being queued. An optimized alternative to locking is used to achieve concurrency control for coupling facility data tables. This is described in the section 'Concurrency control for coupling facility data tables'.

For a VSAM data set being accessed in non-RLS mode, CICS acquires locks (or enqueues) using the NQ domain that prevent the same record from being updated by more than one unit of work at a time. If the file is recoverable, then the lock is not released until syncpoint (that is, the end of the unit of work), otherwise it is released when the request thread completes. A request thread consists, for example, of a read update followed by a rewrite. In non-RLS mode, VSAM also provides a form of concurrency control known as **exclusive control**. The sphere of exclusive control is the control interval (CI), and this means that two different records cannot be concurrently updated if they are both within the same CI. Exclusive control is only maintained while a record is being updated, and is released as soon as the operation is complete.

For a VSAM data set being accessed in RLS mode, VSAM acquires locks at the record level to prevent the same record from being updated by more than one unit of work within the sysplex at a time. If the data set is recoverable, then the lock is

not released until syncpoint, otherwise it is released when the request sequence completes. There is no CI locking with RLS mode.

For a recoverable BDAM file, CICS acquires locks using the NQ domain that prevent the same record from being updated by more than one unit of work at a time.

Concurrency control for coupling facility data tables: Concurrency control for coupling facility data tables is provided by using one of two update models provided by coupling facility data tables support (CFDT support).

The default is the locking update model, in which the CFDT server acquires locks at the record level to prevent the same record from being updated by more than one unit of work within the sysplex at a time. If the data set is recoverable, then the lock is not released until syncpoint, otherwise it is released when the request sequence completes.

The contention update model is an optimized alternative to using locking to achieve update integrity (concurrency control). With this model, which can be specified on a per-data table basis, no locks are acquired when a record is read for update, but if another unit of work subsequently changes or deletes this record, then the first unit of work will be informed that the record has changed (or been deleted) when it comes to rewrite or delete the record itself. The occurrence of such a contention is detected by the CFDT server, and the contention update model is only available for coupling facility data tables.

Sequential retrieval

A facility supported by CICS file control is the sequential retrieval of records from the database. This facility is known as browsing. To initiate a browse operation, the user provides either a specific or generic (partial) record reference (key) for the point at which sequential retrieval is to begin. Each subsequent get request by the user initiates retrieval of the next sequential record. The application, while in browse mode, can issue random get for update requests to a different data set, without interrupting the browse operation. For VSAM files accessed in RLS mode, the application can update the records that it is browsing. For VSAM files accessed in non-RLS mode, and BDAM files, in order to update a record of the same data set, the application must first terminate the browse operation. The same application can concurrently browse several different data sets and browse the same data set with multiple tasks.

With VSAM data sets, the application can skip forward during a browse operation to bypass unwanted data.

All types of CICS data tables (CICS-maintained, user-maintained and coupling facility) can be browsed.

Read Integrity

When a file is accessed in RLS mode, three levels of read integrity are supported:

- UNCOMMITTED read integrity is the same level of read integrity as is supported for non-RLS requests. With this level of read integrity, read requests can return data which has not yet been committed, and which might subsequently be backed out.
- CONSISTENT read integrity. With this level of read integrity, read requests are serialized with concurrent update activity for the record, so that a read request will wait until data which is being updated has been committed (or until the

update has completed, for a non-recoverable data set). This means that read requests will always see commit-consistent data.

- REPEATABLE read integrity. With this level of read integrity, additional locking is used so that in addition to waiting for updates to be committed, records that have been read within a unit of work cannot be updated until the unit of work completes. This means that if a read is repeated within a unit of work, the same data will be returned.

Backout logging

File control will perform automatic logging of file operations which update recoverable files. This logging is written to the CICS system log stream. In the event of either a system or a transaction failure, the information can subsequently be used to restore the recoverable data set as though the current transaction had never run.

For coupling facility data tables, the CFDT server performs its own logging, and is responsible for backing out updates in the event of a failure.

Forward Recovery Logging

If a file (non-RLS VSAM) or data set (RLS or non-RLS VSAM) is defined to be forward recoverable, then CICS will perform automatic logging of file operations which update it. This logging is written to the forward recovery log stream specified on the file definition or data set. In the event of a failure, the information can be used to forward recover from a backup copy of the data set.

Forward recovery support is not provided for user-maintained data tables or coupling facility data tables.

Automatic journaling and logging

Except in the case of user-maintained data tables and coupling facility data tables, CICS provides optional automatic journaling and logging facilities for records that are updated, deleted from, or added to a file control data set. Automatic journaling is specified by the user in the resource definition for each data set affected. For a specified data set, a record read for update, a new record added, or an existing record deleted is automatically written to the specified journal. To allow journaled records to be associated with the appropriate data set (instead of with the CICS file name), a special record is journaled showing the current data set allocation whenever it changes.

Use of concurrent tasks

The file control non-RLS VSAM interface program (DFHFCVR) uses a change-mode request to the dispatcher to allow VSAM I/O requests and VSAM UPAD exit code to run under a concurrent task. This provides overlapping of processing in a multiprocessor environment.

RLS requests use a different mechanism: SMSVSAM assigns each request its own SRB, allowing MVS to concurrently schedule requests in an analogous way to that provided by subtasking for non-RLS.

Shared Data table services

Shared data tables (that is, CICS-maintained and user-maintained data tables) are managed by a set of OCO modules, referred to in this book as “data table services”. The services are invoked by a branch-and-link interface passing a parameter block.

Services provided include the following:

- Initialization
- Open, close, and load of tables
- Retrieval and update of table records
- Backout and commit of table changes
- Statistics.

For files that are defined by the user as CICS-maintained or user-maintained data tables, file control invokes these services at appropriate points in the processing of application requests.

Coupling facility data tables server

Coupling facility data tables are managed by OCO modules within the CICS address space, and in a separate address space that is known as the *Coupling Facility Data Tables server* (CFDT server). The CFDT server provides access to coupling facility data tables residing in a coupling facility data tables pool, so that they can be shared by CICS regions across a parallel sysplex. Refer to Coupling facility data table structures and servers for more details about CFDT servers.

For files that are defined by the user as accessing coupling facility data tables, file control makes calls to the CFDT server at appropriate points in the processing of application requests.

How CICS processes file control requests

CICS receives file control requests from applications through the EXEC interface. This section describes only the mainstream processing for such requests. It does not describe exception conditions. For programming interface information about exception conditions, see the Exception conditions for LINK command in Reference -> Application development. This section also does not provide details about the specific processing for requests to any kind of data table.

Processing using VSAM

For VSAM data sets, this section describes the processing followed when the file is being accessed in non-RLS mode. For RLS mode, the processing is broadly similar, although it differs in some of the interfaces used to VSAM, and the locking mechanisms are very different.

Note: File control processing is constrained by the availability of buffers, CICS strings and (for local shared resource (LSR) files) LSR strings. Tasks can get suspended during the execution of any file control request if there are not enough strings or buffers available for the immediate processing that is to be done.

With VSAM RLS, a task waiting for buffers will be suspended in VSAM rather than in CICS.

Processing using Data Tables

For shared data tables (CICS-maintained and user-maintained data tables), processing is broadly similar to that for non-RLS VSAM. The main differences are that, for remote files, non-update requests may be processed locally instead of being function shipped, and that, in cases where a request cannot be satisfied from a data table, it may be converted into a non-RLS or RLS VSAM request to be processed by DFHFCVS or DFHFCRS, or function shipped via DFHFCDTX.

For coupling facility data tables, processing is also broadly similar to that for non-RLS VSAM. The main difference is that instead of issuing the request to VSAM, a call or calls are made to entry points within the CFDT server, which then

processes the request and returns the results. A task accessing a coupling facility data table may occasionally be suspended in the CFDT server.

Note that the following processing sections do not describe data table processing explicitly.

General request processing

All file requests, whatever the request and whatever the file access method, follow the same general sequence of steps:

1. User exit XFCREQ is called.
2. The request is converted from EXEC parameter list form to FCFR interface form.
3. If this is the first file access request by the transaction, a FRAB is obtained and its address stored in Recovery Manager's FC Token. The FRAB provides the anchor for file request state for this transaction.
4. If this is the first request to this file by the transaction, a FLAB is obtained and the file control table entry is located. If the file is remote or an explicit SYSID has been specified on the request, the FLAB is marked with a remote indicator. If this is not the first request to the file, then the FLAB is located that repressents accesses made to the file by this transaction.
5. If this is the first, or only, request of a request sequence, a FRTE is obtained. If this is not the first request in a request sequence, the FRTE that represents the sequence is located. rather than being function shipped.
6. If the request is to a local file, and if resource security is active, the security check is made, unless a check has already been made within the current UOW for this file.
7. If the request is to a local file and the file is not already open, it is opened and its access method dependent attributes are saved in its file control table entry.
8. The SERVREQ attributes of the file are checked.
9. For READ and browse requests, SET storage is released or obtained, as necesssary.
10. The access method specific request processor is called as follows:
    - DFHFCVS for non-RLS VSAM files
    - DFHFCRS for RLS VSAM files
    - DFHFCBD for BDAM files
    - DFHFCDR for coupling facility data tables
    - DFHFCDTS for user-maintained data tables
    - DFHFCDTS for non-update requests to CICS maintained data tables
    - DFHFCVS for update requests to CICS maintained data tables
    - DFHFCRF for requests that are to be shipped to a remote region
11. CICS has checked whether the file is defined as local or remote. If it is remote, the request is function-shipped to the file-owning region, where CICS processes the request as if it had originated locally.

    There is an exception for CICS-maintained and user-maintained data tables, for which non-update requests are treated as local rather than being function shipped.

    Note that RLS support and coupling facility data tables support both provided shared access within a parallel sysplex without the use of function shipping. Files which use either of these types of sharing will be defined as local on all systems which want to share the data set (in the case of RLS support) or data table (CFDT support).

12. SET storage is obtained for BDAM files or below the line READ requests.
13. The FRTE is released if the request sequence has ended and the file is closed if a close is pending, this FRTE is the last user and the FLAB indicates that the file can be closed.
14. The FCFR responsed are converted to EXEC parameter list responses. In particular, the EIBRCODE and RESP2 values are constructed.
15. User exit XFCREQC is called.

READ request processing: The course of READ request processing depends on the access method, and whether or not the UPDATE option is specified on the request:

VSAM processing:

1. The supplied keylength is validated.
2. A VSAM work area (VSWA) is created. This includes the request parameter list (RPL) that will be passed to VSAM.

   The processing that follows depends on whether the UPDATE option was specified on the READ request.

   UPDATE option not specified:

   a. The RPL is completed, and a call made to VSAM to get the record.
   b. If the request specifies INTO and the record is too large for the user-specified area, the request is reissued specifying a work area large enough to hold the record. The record is then copied to the user-specified area in truncated form, and the LENGERR condition is raised.
   c. The VSWA is freed.
   d. The read is journaled if specified in the resource definition.

   UPDATE option specified:

   a. The UPDATE flag is set in the RPL.
   b. An attempt is made to read the record by issuing the VSAM request. READ UPDATE requires exclusive control of the control interval (CI) containing the record. VSAM manages the locking mechanism for control intervals. If the CI is already locked, VSAM returns an error and the requesting task is forced to wait on resource type FCXCWAIT.
   c. CICS file control acquires a record lock on the record just read, using a CICS ENQUEUE request. The record lock prevents any other transaction from updating the record before the owning transaction has reached a syncpoint (for recoverable files), or before the REWRITE, DELETE, UNLOCK or syncpoint that completes the request sequence (non-recoverable files).
   d. Exclusive control of the CI is retained until the REWRITE, DELETE, or UNLOCK request that follows the READ UPDATE has been completed, or until the next syncpoint.

      The CICS record lock (if any) is retained until the next syncpoint, in case the transaction updating the record abends and dynamic transaction backout processing is necessary.
   e. If the file is recoverable the request is logged. If required, the request is also recorded in a user-specified journal.

   BDAM processing:

   a. A file I/O area (FIOA) is obtained.
   b. If the UPDATE option has been specified:
      1) The address of the RIDFLD is saved in the FIOA.

2) If the data set is recoverable, the RIDFLD is ENQUEUEd on to lock the record against other updates. The ENQUEUE is retained until the next syncpoint.

c. The KEYLENGTH is checked for validity.

d. The key field is converted from character string format (TTTTTTRR) to binary format (TTR), if necessary.

e. A BDAM READ request is issued. If the READ is successful, the required block is returned in the FIOA.

f. The key field returned by BDAM is converted from binary format to character string format, if necessary.

g. If the file is recoverable and UPDATE has been specified, the request is logged. If required, the request is also recorded in a user-specified journal.

h. If deblocking is required, the required record is located in the block that has been returned by BDAM:

1) If DEBREC has been specified, the record number is used to locate the record.

2) If DEBKEY has been specified, the embedded key is used to locate the record.

WRITE request processing:

The course of WRITE request processing depends on the access method, and for VSAM access, on whether the file is a data table.

VSAM processing

1. The KEYLENGTH is checked for validity. If it is incorrect, the INVREQ condition is raised.
2. A VSAM work area (VSWA) is created. This includes the request parameter list (RPL) that will be passed to VSAM.

   Different paths are now followed depending on the type of file.

   ESDS file:

   a. If the file is recoverable, or writes are to be journaled:

   1) If this is not the first write of a sequence and the ESDS write lock is being waited for by another transaction, release the lock and end this sequence, logging the completion if recoverable.

   2) If this is (or has become) the first write of a sequence, acquire the ESDS write lock for the data set.

   b. If the file is recoverable, the WRITE ADD request is recorded in the CICS system log.

   c. If required, the WRITE ADD request is recorded in a user-specified journal.

   d. Any fields in the RPL not supplied when the VSWA was created are completed.

   e. The RPL is set to point to the user-specified data area. If the user specified a record that is too large for the file, the length in the RPL is set to the maximum length, so that the record is truncated.

   f. A VSAM PUT request is issued to write the record.

   g. If the file is recoverable, a CICS record lock is obtained for the record that has just been written. The record lock will be retained until the next syncpoint, in case the transaction writing the record abends and dynamic transaction backout processing has to be performed.

h. If the file is recoverable, the after-image of the record is logged for forward recovery and a write complete record is written on the system log.

i. If not a MASSINSERT, the ESDS write lock is released, if held.

KSDS or RRDS file:

a. For KSDS requests, the RIDFLD key specified in the request is checked against the key field in the record to be written. (The record is currently in the application FROM data area.) If it does not match, the INVREQ condition is raised.

b. If the file is recoverable and not in load mode:

   1) A CICS lock is obtained on the record that is to be written, and an attempt is made to read the record (by means of a VSAM GET request) to discover whether it already exists in the file. If it does, the DUPREQ condition will be raised on the write to VSAM.

   2) If the file is a KSDS, and if this request is part of a MASSINSERT, or if a MASSINSERT is in progress, the read is issued with GTEQ to find the next record in the base data set. A lock is created, using the key of this next record, to prevent other transactions from inserting records into the empty range.

   3) If there is no existing record with the given key, the WRITE ADD request to VSAM is recorded in the CICS system log and, if required, in a user-specified journal.

c. If the file is not recoverable or in load mode, the WRITE request is recorded, if required, in the user-specified journal, and if recoverable a record lock is obtained and the write logged.

d. Any fields in the RPL not supplied when the VSWA was created are completed.

e. If a data table is associated with the base cluster (the data table will be a CICS-maintained table, as user-maintained and coupling facility data tables follow a separate processing path which is not described here). a data table pre-add is issued to place the record in the table as a not-yet-valid entry. If the file is recoverable, a record lock is already held; if not, a lock is acquired before the data table service is called.

f. A VSAM request is issued to write the record.

g. If the file is recoverable, the after-image of the record is logged for forward recovery.

h. If required, the after-image is recorded in a user-specified journal.

i. If the file is a data table, a data table request is issued to complete the add to the data table by validating the record. If a record lock was obtained for a non-recoverable file, it is released.

3. If the MASSINSERT option has *not* been specified on the WRITE request, the VSWA for the operation is released.

   If MASSINSERT has been specified, the VSWA is not released, because it is likely to be needed for subsequent WRITE operations. In this case, the end of MASSINSERT processing is notified to VSAM by the CICS UNLOCK function. (See "UNLOCK request processing" on page 1598.)

   Specifying MASSINSERT causes exclusive control of the CI to be acquired. Exclusive control is released by issuing an UNLOCK request. To avoid deadlocks, this should be done immediately after the last WRITE MASSINSERT request.

BDAM processing

1. The KEYLENGTH is checked for validity. If it is incorrect, the INVREQ condition is raised.
2. The WRITE command input is checked to ensure that MASSINSERT has not been specified—BDAM does not support MASSINSERT processing. If it has, condition INVREQ is raised.
3. A file I/O area (FIOA) is obtained.
4. If the file is recoverable, the record to be written is ENQUEUEd on. The lock is retained until the next syncpoint.
5. The record to be written is copied from the user-supplied data area to the FIOA. If the record is too large, it is truncated.
6. If the file is recoverable, the request is logged. If required, the request is also recorded in a user-specified journal.
7. The key field is converted from character string format to binary format, if necessary, and the BDAM I/O request issued.
8. The key returned by BDAM is converted from binary format to character string format, if necessary, and passed to the application.
9. A supervisor call (SVC 53) is issued to release BDAM exclusive control, if necessary.
10. The FIOA is released by using a FREEMAIN request.

REWRITE request processing: The REWRITE request is used to write a record back to a file following a READ UPDATE request. **VSAM processing:**

1. The RPL is set to point to the user-specified data area. If the user specified a record that is too large for the file, the length in the RPL is set to the maximum length, so that the record is truncated.
2. The RPL is completed.
3. If there is a data table associated with the base cluster (this data table is a CICS-maintained table, as user-maintained tables follow data table processing):
   a. If the file is unrecoverable, a record lock is obtained. (If the file is recoverable, a lock is already held).
   b. A data table request is issued to invalidate the record in the table before the VSAM update.
4. VSAM is called to PUT(UPDATE) the record. Exclusive control of the CI, which was obtained for the preceding READ UPDATE request, is released, but the CICS record lock (for recoverable files) is retained until the next sync point, in case the transaction abends and dynamic transaction backout processing is necessary.
5. If there is a data table associated with the data set, the table record is updated and its validity is reinstated, by issuing a call to data table services. If the file is unrecoverable, the record lock is released.
6. If the file is recoverable, and if the record is successfully rewritten, the after-image is written to the log for forward recovery.
7. The VSWA for the operation is released.

   Note: When a record is updated by way of a path, the corresponding alternate index is updated by VSAM to reflect the change. However, if the record is updated directly by way of the base, or by a different path, the alternate index is only updated by VSAM if it has been defined to VSAM (when created) to belong to the **upgrade set** of the base data set.

BDAM processing:

1. The FIOA that was used in the corresponding READ UPDATE request is located, and the modified record read into it from the user-specified area. If the record is too long, it is truncated.
2. A FREEMAIN call is issued to release the FWA.
3. If the file is recoverable, the request is logged. If required, the request is also recorded in a user-specified journal.
4. The key field is converted from character string format to binary format, if necessary, and the BDAM I/O request issued.
5. The key returned by BDAM is converted from binary format to character string format, if necessary, and passed to the application.
6. A supervisor call (SVC 53) is issued to release BDAM exclusive control, if necessary.
7. A FREEMAIN call is issued to release the FIOA.

UNLOCK request processing: The UNLOCK request is used to release exclusive control obtained during a READ UPDATE (VSAM or BDAM) or WRITE MASSINSERT (VSAM only) request.

VSAM processing (including CICS-maintained data tables):

1. The VSWA for the operation is released, together with associated storage.
2. An ENDREQ request is sent to VSAM. This releases exclusive control of the CI, if it is held, and frees any VSAM strings.

BDAM processing:

1. A supervisor call (SVC 53) is issued to release BDAM exclusive control, if necessary.
2. A FREEMAIN call is issued to release the FIOA.

DELETE request processing:

The course of DELETE request processing depends on whether a RIDFLD has been specified. The processing for user-maintained data tables differs from that for CICS-maintained data tables. DELETE requests are not valid for VSAM ESDS or for BDAM files.

VSAM processing (including CICS-maintained data tables)

1. If a RIDFLD has been specified:
   a. If a KEYLENGTH has been specified, it is checked for validity.
   b. If the GENERIC option has been specified, and the file is *not* a KSDS, condition INVREQ is raised.
   c. A VSWA is created.
2. If no RIDFLD was specified, the SERVREQ attribute of the file is checked to ensure that DELETE requests are valid for this file. If not, the INVREQ condition is raised.
3. If a RIDFLD has been specified and GENERIC is not specified, the following actions are performed once. If a RIDFLD has been specified and GENERIC is specified, the following actions are repeated until there are no more records that contain the generic key.
   a. VSAM is requested to GET for UPDATE a record with the specific or generic key. GET UPDATE processing requires exclusive control of the CI. The record is read into an internal buffer.

The generic key value, if supplied, is checked against the key contained in the record. If it does not match, there are no more records containing the generic key in the file.

b. If the file is recoverable:
   1) A CICS record lock is obtained for the record. This will be held until the next syncpoint.
   2) The VSAM GET UPDATE request is recorded synchronously on the system log.
   3) A CICS range lock is obtained for the record to be deleted if a MASSINSERT is in progress. This is to prevent an end-of-range record from being deleted while the range is in use for a MASSINSERT sequence.

c. If there is a data table (which will be CICS-maintained) associated with the base cluster, a record lock is acquired if the file is nonrecoverable, and a data table pre-update call is issued to invalidate the record before the VSAM update.

d. A VSAM ERASE request is issued, to delete the record from the file.

e. If there is a data table associated with the base cluster, the record is deleted from the table by issuing a call to data table services. If the file is nonrecoverable, the record lock is released.

f. If a range lock was acquired, it is released.

g. If the file is recoverable, a WRITE DELETE record is written in the system log for forward recovery.

h. If required, a WRITE DELETE record is written to a user-specified journal.

4. The VSWA is released.

STARTBR and RESETBR request processing: STARTBR and RESETBR request processing are very similar, and are described together.

VSAM processing:

1. A VSWA is created if STARTBR.
2. The user key is recorded in the VSWA for use in subsequent BROWSE processing.
3. A call is made to VSAM to point to the record, and to acquire shared control of the CI.

BDAM processing:

1. An FIOA is obtained and initialized if STARTBR.
2. The initial key is saved in the FIOA, converting the key from character string format to binary format if necessary.
3. If deblocking is required, the deblocking indicator (DEBREC or DEBKEY) is saved in the FIOA.

READNEXT and READPREV request processing: READNEXT and READPREV request processing are very similar, and are described together.

VSAM processing:

1. A check is made that READPREV with a generic key was not requested. If it was, condition INVREQ is raised.
2. If KEYLENGTH was specified, it is checked for validity. If it is incorrect, the INVREQ condition is raised.
3. The RPL options are set.

4. If SET is specified, an internal work area is obtained and the RPL is set to point to the work area. The area is either above or below the 16MB line, depending on the requirements of the application.
5. If INTO is specified, the RPL is set to point to the user-specified area.
6. A VSAM request is issued to read the record. Shared control of the CI is needed, and the request will not succeed if some other task already has exclusive control. In such a case, a call is made to VSAM to reestablish the correct position in the file. The task then waits until VSAM informs CICS that the CI is available to the task. CICS resumes the task, which can now acquire shared control and obtain the required record.
7. If SET is specified, the SET pointer points to the work area.
8. If INTO is specified, a check is made to see if the record is too large to fit into the user-specified area. If it is too large, the request is reissued using an internal work area, the data is copied from the work area into the user-specified area and truncated, and the LENGERR condition is raised.
9. If required, the request is recorded in a user-specified journal.

BDAM processing—READNEXT requests:

1. A check is made that READPREV was not issued. If it was, condition INVREQ is raised.
2. The FIOA that was created on STARTBR is located.
3. If a new block is required, a BDAM I/O request is issued to get it.
4. If deblocking is required, the required record is located in the block that has been returned by BDAM:
   a. If DEBREC has been specified, the record number is used to locate the record.
   b. If DEBKEY has been specified, the embedded key is used to locate the record.
5. If INTO is specified, the record or block is moved from the FIOA to the user-specified area. If the record is longer than the user-specified area, it is truncated, and the LENGERR condition is raised.
6. If SET is specified, the SET pointer points to the record in the FIOA.
7. The RIDFLD of the record is returned to the application.
8. The current browse position is recorded in the FIOA.

ENDBR request processing:

The ENDBR request is used to end a browse session on a file. To avoid deadlocks, ENDBR must be issued when the browse session is complete.

VSAM processing:

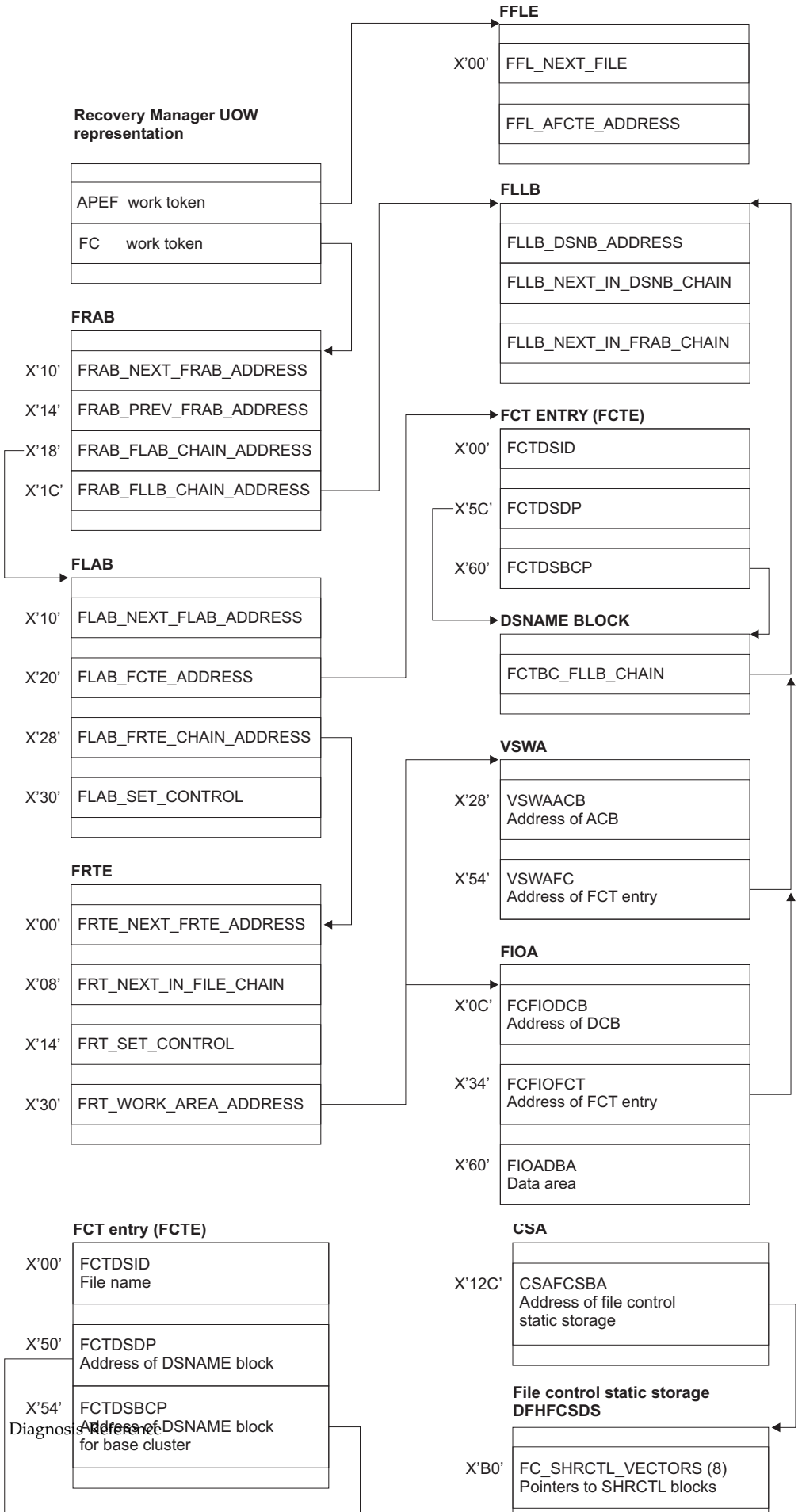
1. An ENDREQ request is sent to VSAM. This frees any VSAM strings that are held, and relinquishes shared control of the CI.
2. The VSWA for the operation is released.

BDAM processing:
The FIOA that was used for the browse session is released by using a freemain request.

Control blocks

Figure 49 on page 1602 shows the major control blocks associated with file control. Control blocks which are not shown in this diagram include those relating to

coupling facility data tables support.


FFLE
X'00'
FFL_NEXT_FILE
FFL_AFCTE_ADDRESS
Recovery Manager UOW representation
APEF work token
FC work token
FLLB
FLLB_DSNB_ADDRESS
FLLB_NEXT_IN_DSNB_CHAIN
FLLB_NEXT_IN_FRAB_CHAIN
FRAB
X'10'
FRAB_NEXT_FRAB_ADDRESS
X'14'
FRAB_PREV_FRAB_ADDRESS
X'18'
FRAB_FLAB_CHAIN_ADDRESS
X'1C'
FRAB_FLLB_CHAIN_ADDRESS
FCT ENTRY (FCTE)
X'00'
FCTDSID
X'5C'
FCTDSDP
X'60'
FCTDSBCP
DSNAME BLOCK
FCTBC_FLLB_CHAIN
FLAB
X'10'
FLAB_NEXT_FLAB_ADDRESS
X'20'
FLAB_FCTE_ADDRESS
X'28'
FLAB_FRTE_CHAIN_ADDRESS
X'30'
FLAB_SET_CONTROL
VSWA
X'28'
VSWAACB
Address of ACB
X'54'
VSWAFC
Address of FCT entry
FRTE
X'00'
FRTE_NEXT_FRTE_ADDRESS
X'08'
FRT_NEXT_IN_FILE_CHAIN
X'14'
FRT_SET_CONTROL
X'30'
FRT_WORK_AREA_ADDRESS
FIOA
X'0C'
FCFIODCB
Address of DCB
X'34'
FCFIOFCT
Address of FCT entry
X'60'
FIOADBA
Data area
FCT entry (FCTE)
X'00'
FCTDSID
File name
X'50'
FCTDSDP
Address of DSNAME block
X'54'
FCTDSBCP
Address of DSNAME block
for base cluster
CSA
X'12C'
CSAFCSBA
Address of file control
static storage
File control static storage
DFHFCSDS
X'B0'
FC_SHRCTL_VECTORS (8)
Pointers to SHRCTL blocks

Note: The pointer to the DSNAME block, FCTDSDP, is different from the pointer to the base cluster DSNAME block, FCTDSBCP, only when the FCT entry does not represent a base. DSNAME blocks that do not correspond to bases do not have the base cluster information, although the space is allocated.

These control blocks are described in "Access method control block (ACB)" through "VSAM work area (VSWA)" on page 1611.

Access method control block (ACB)

The ACB identifies the file that is associated with this VSAM request to VSAM.

It is passed to VSAM by DFHFCRV for RLS, or DFHFCVR for non-RLS (it is the RPL, which points to the ACB, that is passed) to initiate a VSAM request. The ACB lasts as long as the associated CICS file is open; that is, it is created at file open time and deleted at file close time by DFHFCN for non-RLS, or DFHFCRO for RLS.

The ACB is addressable through a pointer in the associated FCT entry. In addition, a 4-byte field appended (by CICS) to the ACB structure points back to this FCTE.

Note that the ACB is a VSAM control block.

At open time, storage is obtained from a subpool in 31-bit storage. A VSAM GENCB macro is issued to generate the ACB with attributes obtained from the FCT entry. At open time, VSAM fills in more information in the ACB. Some of this is subsequently copied back into the FCTE.

The storage for the ACB is freed when the file is closed.

There is one ACB per VSAM FCT entry.

The layout of the ACB is defined by the VSAM IFGACB structure, and also by a DSECT of the same name.

ACBs are not cataloged and are not restored across WARM or emergency starts. The ACB is rebuilt every time a CICS file is opened.

A special type of ACB, known as a base cluster ACB, is created by DFHFCM to allow for the implicit opening of a base cluster, when required by a non-RLS file access through an alternate index path. In this case, the 4-byte field appended to the ACB structure points to the associated DSNAME block for the base cluster.

A second special type of ACB, known as a *control ACB* is required for VSAM RLS processing. Storage for the control ACB is obtained by DFHFCCA and filled in using the GENCB macro before registering the control ACB. The storage is freed when the control ACB is unregistered by DFHFCCA. The control ACB is passed to VSAM on calls issued by DFHFCCA. It is used for all requests that are not associated with a specific file.

Data control block (DCB)

The DCB identifies to BDAM the file associated with this BDAM request. It is passed to BDAM by DFHFCBD to initiate a BDAM request, and lasts for the lifetime of the CICS run.

The DCB is addressable through a pointer in the associated FCT entry. In addition, a 4-byte field appended (by CICS) to the DCB structure points back to this FCTE.

Note that the DCB is a BDAM control block.

There is one DCB per BDAM FCT entry.

The layout of the DCB is defined by the generalized structure IHADCB. The structure is qualified with a parameter stating that a BDAM DCB is required. There is also a DSECT of the same name.

The DCB is assembled as part of the FCT. (Note that there is no RDO for BDAM files.) DFHFCRP acquires storage for the DCB below the 16MB line and copies the DCB into it (only on cold start). The DCB is cataloged and restored across a warm and emergency start. Thus, unlike an ACB, a DCB is only built once.

Data set name block (DSNB)

The DSNB represents a physical VSAM or BDAM data set that is being accessed through one or more CICS files. It is used by file control to hold information relevant to the data set and not only to the CICS file. Also, it provides a single "anchor block" to control many requests accessing this data set through many different CICS files.

After it has been created, a DSNB survives the lifetime of a CICS run unless the user deletes it by means of an EXEC CICS SET DSNAME REMOVE command or its CEMT equivalent.

The DSNB is addressable through pointers in an FCTE entry, or through DFHTMP using the 44-character name as a key, or using the DSNB number as a key.

A DSNB is created, if it does not exist already, when an FCTE attempts to connect itself to a DSNB. This happens at file open time, or when an EXEC CICS SET FILE DSNAME command (or its CEMT equivalent) is executed.

A DSNB that represents a VSAM base data set has a **base cluster block** embedded in it, which has information specific to the base data set. Note that a BDAM data set has a small amount of information held in the base cluster block.

A DSNB representing a VSAM path has a blank base cluster block embedded in it.

Information about the base data set is obtained from the VSAM catalog when a CICS file (path or base) referencing that data set is opened. The information is stored in the base cluster block.

DSNBs are cataloged in the CICS global catalog and are restored across warm and emergency starts.

DSNBs reside above the 16MB line.

The layout of the DSNB is defined by the DFHDSNPS structure, and by the DFHDSNDS DSECT (using the DFHDSND macro).

The DFHFCDN module handles DSNAME blocks (creation, deletion, FCTE-DSNB connections). DFHFCDN also provides an interface for the EXEC layer to process DSNAME blocks through the use of EXEC CICS INQUIRE or SET DSNAME, and CEMT INQUIRE or SET DSNAME. Modules within the file control component can access the DSNBs directly through pointers in the FCTE.

File browse work area (FBWA)

The FBWA maintains the state of a browse to a data table. It is used for browsing coupling facility data tables, CICS-maintained data tables, and user-maintained data tables.

An FBWA is created when the browse is started (by a STARTBR request), and is addressed by the FRT_FBWA_ADDRESS field in the FRTE. It is stored in a file control IO buffer of the appropriate size to hold the key information.

Some of the fields are specific to CICS-maintained data tables, because the source data set will sometimes be accessed during a browse of a CICS-maintained data table.

There is a variable-length portion at the end of the FBWA. It contains keys, which are pointed to by fields, as shown in the following list, in the fixed part.

- CURRENT_KEY points to the first of the key fields, which is used to hold the key returned by the most recent request.
- REQUEST_KEY points to the second of the key fields, which is used to contain the key specified at the start of a browse segment (STARTBR or RESETBR).
- NEXT_KEY points to the third of the key fields, which is used for CICS-maintained data tables to handle "gaps".

File control static storage (FC static)

File control static storage is used by file control to store information for use throughout the lifetime of a CICS run; for example, SHRCTL vectors and entry points of file control modules.

This storage is used by file control modules and by modules outside the file control component, and lasts for the lifetime of a CICS run. It is addressed by a field in the CSA named CSAFCSBA. It is created by DFHFCIN during CICS initialization before DFHFCRP gets control, and resides in 31-bit storage.

FC static storage is defined by the DFHFCSPS structure and by the DFHFCSDS DSECT.

File control quiesce receive element (FCQRE)

File control uses quiesce receive elements to communicate details of quiesce requests received from SMSVSAM. There is also a permanent error FCQRE used for communicating errors.

The FCQRE contains information about the data set to which the quiesce applies (or the cache for quiesce type QUICA), the type of quiesce, and (for the error FCQRE) the type of error and error data.

Each quiesce request received from SMSVSAM via the quiesce exit results in DFHFCQX, the quiesce exit module, creating an FCQRE which is passed to DFHFCQR, the quiesce receive system task module.

Storage for FCQREs is obtained from storage obtained by using a z/OS GETMAIN request for 31-bit storage.

FCQREs are chained in a one-way linked list anchored from file control static storage. The permanent error FCQRE is also anchored from file control static storage, and is added to the FCQRE chain when an error occurs.

The layout of the FCQRE is defined by the DFHFCQRE structure and the DFHFCQRE DSECT.

File control quiesce send element (FCQSE)

File control uses quiesce send elements to communicate the details of quiesce requests that are to be sent to SMSVSAM.

They contain information about the task initiating the request, the data set to be quiesced, the type of quiesce requested, and the address of an ECB which is posted by SMSVSAM when the request is completed.

Each quiesce request initiated by CICS results in DFHFCQI, the quiesce initiate module, creating an FCQSE which is passed to DFHFCQS, the quiesce send module.

Storage for FCQSEs is obtained from the FC_ABOVE subpool, which resides in 31-bit storage.

FCQSEs are chained in a two-way linked list anchored from fields in file control static storage.

The layout of the FCQSE is defined by the DFHFCQSE structure and the DFHFCQSE DSECT.

File control coupling facility data table pool element (FCPE)

A file control coupling facility data table (CFDT) pool element represents one connection to a Coupling Facility Data Table Pool. For each CFDT pool that can be accessed by a given MVS image, there is a CFDT server running in that image that manages access to the pool.

An FCPE is created and chained to FC static when a file definition that refers to the pool is installed and there is not already a pool element for that CFDT pool. An FCPE can be created in the following situations:

- When files are installed at CICS startup
- When files are installed using CEDA
- When a SET FILE is issued that names a CFDT pool for which a pool element does not exist already.

FCPEs are obtained by using a getmain request from the FCPE subpool that DFHFCRP creates during File Control Initialization, and chained to the FCPE chain in FC static. The head of the FCPE chain is the field FC_FCPE_CHAIN.

FCPEs are catalogued when they are created so that they can be restored at emergency restart.

File control coupling facility data table pool wait element (FCPW)

The file control coupling facility data table (CFDT) pool wait element (FCPW) represents a task that attempted to issue a request to a CFDT that resides in a particular pool, but that must wait because there are no available request slots. Depending on the kind of request, the FCPW represents a locking request slot (LRS) wait or a MaxReqs wait. A flag in the FCPW indicates the kind of wait.

The FCPW is created when a task goes into a MaxReqs or LRS wait. It is obtained by using a getmain request from the pool wait element subpool, and appended to a chain of wait elements for the pool. The wait chains are anchored in the pool

element (FCPE), with one FCPW for each task that is waiting. The FCPE contains head and tail fields for the chains of LRS and MaxReqs FCPWs (FCPE_FIRST_LRS_WAITER, FCPE_LAST_LRS_WAITER, FCPE_FIRST_WAITER, and FCPE_LAST_WAITER). The chains are manipulated using logic that does not require any special case code for the ends of the chains, but that does mean that when the chains are empty, the head and tail fields contain a special initial value, rather than zero.

The FCPW includes the following:
- A pointer to the next FCPW in the chain (if no next FCPW, this contains the special initial value)
- A pointer to previous FCPW in the chain (if no previous FCPW, this contains the special initial value)
- The suspend token for the wait
- The task token of the waiting task
- The suspend start time

File control table entry (FCTE)

Each entry in the file control table defines a CICS file that is defined to be the CICS view of a VSAM or BDAM data set or a data table. The FCTE is used by all modules in the file control component (but never outside), and lasts for the lifetime of a CICS run, or from when it is created by RDO to the end of the CICS run.

The FCTE contains information that can be split into three broad groups:
- CICS information about the file, including statistics
- Information that is used as input to build the VSAM ACB or BDAM DCB
- Information that is returned by VSAM, both from the ACB and direct from the VSAM catalog, when the file is opened.

An FCTE can be created in two ways:
- By defining the file using the DFHFCT TYPE=FILE macro (BDAM only).
- By defining the file online using RDO while CICS is running (VSAM only).

File control table entry (FCPW)

File control coupling facility data tables UOW pool block (FCUP)

The file control coupling facility data table (CFDT) UOW pool block (FCUP) represents recoverable updates made within a unit of work to one or more CFDTs residing in a CFDT pool. An FCUP block is created when a unit of work makes its first recoverable request to a CFDT in a given pool, at the same time as an RMC link is added to represent the recoverable update.

There is one FCUP block per UOW per recoverably-updated CFDT pool. The FCUP is obtained by using a getmain request and released by using a freemain request from the FCUP subpool by using the storage manager quickcell mechanism. The FCUP blocks for a unit of work are chained from the FRAB for that unit of work, addressed by FRAB_FCUP_CHAIN_ADDRESS.

An FCUP block contains the following:
- Forward and back pointers for the chain of FCUP blocks relating to this unit of work
- The name of the CFDT pool
- The CFDT RMC link token.
- A pointer to the pool element for the CFDT pool

- A pointer back to the owning FRAB

File input/output area (FIOA)

The FIOA is analogous to the VSWA for VSAM, in that it represents the request to BDAM. Embedded in the FIOA is what is known as the data event control block (DECB), which is passed to BDAM to initiate the request.

The FIOA is used by DFHFCBD when processing browse requests against BDAM files. It holds position in a browse when browsing a BDAM file.

An FIOA survives as long as the DECB needs to survive to complete the BDAM request; for example, it survives from READ UPDATE to the REWRITE request.

The address of the FIOA is held in the file request thread element (FRTE) in the FRT_WORK_AREA_ADDRESS field.

Storage for the FIOA is acquired from below the 16MB line.

The layout of the FIOA is defined by the DFHFIOA DSECT.

File lasting access block (FLAB)

The FLAB serves as an anchor for the set of file request thread elements (FRTEs) that belong to a specific file in a given transaction and a given environment.

If a transaction accesses several files from the same environment, there is one FLAB for each file. If a transaction accesses the same file from more than one environment, there is one FLAB for each environment.

The FLAB contains pointers to the FCTE for the file, to the owning FRAB, to the chain of FRTEs owned by the FLAB, and to the next FLAB in the chain of FLABs for the unit of work.

The FLAB is used by file control for the following purposes:

- To anchor the FRTEs for the file in the unit of work and environment
- To ensure that a file cannot be closed if there are any FRTEs associated with it, or if there have been recoverable updates made by units of work that have not yet reached syncpoint phase 2
- To ensure that the corresponding file entry cannot be reallocated to a different data set, even if the file is closed and disabled, when there is uncommitted recoverable work associated with the file
- To hold READ SET storage control information across intermediate syncpoints
- To ensure that units of work which have updated the file reach syncpoint before a copy or BWO copy for a file opened in RLS mode is allowed to proceed
- To record the reason for a failure during syncpoint, and keep track of the fact that the file has uncommitted updates in a unit of work as a result of the failure

The file lasting access block is built by DFHFCFR as part of processing of the first file control request for a specific file in a given transaction and environment. FLABs for recoverable files are also rebuilt by DFHFCIR at warm and emergency restart.

The storage for the FLAB is obtained from a FLAB storage subpool in 31-bit storage.

The FLAB is deleted after all the FRTEs have been processed during syncpoint terminate processing, providing that there have been no syncpoint failures for the file in the unit of work. The FLAB storage is not returned to the FLAB storage subpool, but is instead added to a chain of free FLABs, anchored from file control static storage. Subsequent requests to build a FLAB are, if possible, satisfied by a quick cell mechanism from this chain.

If a unit of work is shunted as a result of a syncpoint failure, the FLABs for any files which suffered the syncpoint failure are also shunted.

The chain of FLABs for a unit of work is anchored from field FRAB_FLAB_CHAIN_ADDRESS in the FLAB.

The layout of the FLAB is defined by the DFHFLAB structure and the DFHFLAB DSECT.

File control locks locator blocks (FLLBs)

The file control locks locator block records the fact that a unit of work held locks against a file that were protecting uncommitted changes to the file, and that it is now uncertain whether the locks are valid. This can occur, for example, if the data set against which the locks were held is now in the lost locks state, or if a non-RLS open for update has taken place despite the presence of retained locks and has overridden the locks (in this case the locks are intact, but the data might not be). It is used by file control to keep track of outstanding recovery work, because whilst the data set still has FLLBs associated with it, special processing rules apply (the actual rules vary with the type of lock condition that has occurred).

FLLBs are created by DFHFCRR (for the lost locks condition, or for an OFFSITE=YES CICS restart), or by DFHFCRO (after a file open which has returned the “non-RLS override” reason code).

FLLBs are chained from both the associated DSNB and the associated FRAB. There is one FLLB per file that held locks per unit of work. Since the FLLB records information about a data set and a unit of work, it contains the DSNB address and the local unit of work ID. It also contains an indicator of the type of lock failure condition that it represents.

FLLBs are obtained by using a getmain request from an FLLB subpool above the 16 MB line.

File control locks locator blocks are released by using a freemain request by DFHFCRC at commit time when there are no longer any retained FLABs for the file.

The layout of the FLLB is defined by the DFHFLLB structure and the DFHFLLB DSECT.

File request anchor block (FRAB)

The file request anchor block (FRAB) serves as an anchor for the set of file lasting access blocks (FLABs) that belong to a specific transaction. The file request thread elements (FRTEs) are chained from the FLABs. The FRAB identifies the transaction to which a given file control request belongs.

The FRAB contains pointers to: the next FRAB in the chain from the FC static, the chain of FLABs for this transaction, the chain of FLLBs for the transaction, and any VSAM work area (VSWA) that has suffered exclusive control conflict for the

transaction. The FRAB also contains some indicators related to recovery, such as whether the transaction holds RLS locks, whether the unit of work has been through phase 2 of syncpoint, and whether the unit of work has ever been shunted. There is also some information related to RLS access, including the local unit of work id, a timeout value to be specified on RLS requests, and some problem determination information returned by VSAM record-level sharing (RLS) when deadlocks occur.

The FRAB is built by DFHFCFR as part of processing of the first File Control request in a transaction. The storage for the FRAB is obtained from a FRAB storage subpool in 31-bit storage. The address of the FRAB is then used as the Recovery Manager token associated with the client name FC. At warm or emergency restart, for units of work that had not completed when CICS terminated, FRABs are rebuilt by DFHFCIR. A FRAB is also built if a failure occurs during phase 2 of an intermediate syncpoint. The original FRAB for the transaction is shunted, along with the failed parts of the unit of work, and the newly built FRAB is passed on to the next unit of work in the transaction.

If a unit of work is shunted, the FRAB is shunted with it, unless there was no recoverable file control work in the unit of work.

The FRAB is deleted after all the FLABs have been processed during syncpoint at transaction termination. At the same time, the Recovery Manager token is set to zero. At this point, the FRAB storage is not returned to the FRAB storage subpool, but is instead added to a chain of free FRABs, anchored from file control static storage. Subsequent requests to build a FRAB are, if possible, satisfied by a quick cell mechanism from this chain.

Issuing an INQUIRE_WORK_TOKEN call to the recovery manager with client name FC returns the address of the file request anchor block for a transaction. There is a chain of all the FRABs in a CICS system, anchored from field FC_FRAB_CHAIN in file control static storage.

The layout of the FRAB is defined by the DFHFRAB structure and the DFHFRAB DSECT.

File request thread elements (FRTEs)

File control uses FRTEs to represent active file control requests and to link related requests together as a file thread. File control also uses FRTEs to anchor SET storage used for READ SET UPDATE requests and browse requests with the set option, the lifetime of which is that of the request thread.

Examples of related requests linked together as a file thread are the request sequences STARTBR, READNEXT, ..., ENDBR, and READ UPDATE, REWRITE.

FRTEs are created by the main file control module, DFHFCFR. They are freed either by DFHFCFR at the end of a request or thread of requests, or by the file control recovery control program, DFHFCRC, at syncpoint if this occurs before a thread of requests has completed.

FRTEs for a specific file in a specific task and environment are chained together, and anchored from the FLAB for that file, task and environment.

Storage for FRTEs is acquired from 31-bit storage.

The layout of FRTEs is defined by the DFHFRTE structure and by the DFHFRTE DSECT.

Keypoint list element (KPLE)

The keypoint list forms part of file control's implementation of backup while open (BWO) copy for data sets accessed in non-RLS mode. One KPLE exists for each keypoint and records the start and end times at which tie up records are written.

The KPLE chain is anchored from FC_KPLE_CHAIN in file control static storage.

The keypoint list elements are created, processed, and deleted (when they become redundant) by DFHFCRC following RMKP take keypoint calls from the recovery manager. These calls are made whenever a CICS keypoint is taken. KPLEs are obtained by using a getmain request from 31-bit storage.

The layout of the KPLE is defined by the KPLE structure.

Shared resources control (SHRCTL) block

The SHRCTL block represents the CICS region's requirements of, and the use made of, a local shared resources pool (LSRPOOL). It is used by DFHFCL when calling VSAM to build an LSRPOOL. It is also used by DFHFCL and statistics programs to hold and update file control statistics.

It lasts for the lifetime of a CICS run, and is addressable through a pointer in file control static storage. There are two hundred and fifty five pointers collectively named the SHRCTL vector.

A SHRCTL block holds information such as how many virtual and hyperspace buffers of a particular size are needed, how many strings are needed, the maximum key length allowed. CICS passes this information to VSAM when the pool is built. It also holds statistics about the pool which are sent to the statistics domain when requested or when the pool is deleted.

Each SHRCTL block represents one LSRPOOL, and there can be up to two hundred and fifty five SHRCTL blocks. The layout of each SHRCTL block is defined by the DFHFCTLS structure and by the DFHFCTSR DSECT, and they reside above the 16MB line.

On a CICS cold start, DFHFCRP performs the following:

- Builds a SHRCTL block for each LSRPOOL definition using user-specified information
- Catalogs each created SHRCTL block in the CICS global catalog.

On a CICS warm or emergency start:

- DFHFCRP restores all previously created SHRCTL blocks from the global catalog.
- For LSR pools that do not have an explicit definition, but which are implicitly defined by CICS when the first file using the pool is open, CICS creates the SHRCTL block at this time. It is catalogued and will be recovered by DFHFCRP on a warm or emergency restart.

VSAM work area (VSWA)

The VSAM work area (VSWA) represents a VSAM request to CICS. Embedded in the VSWA is the request parameter list (RPL) which is passed to VSAM to perform the request. In addition to the RPL, the VSWA contains other CICS information related to the request.

The VSWA is used by DFHFCVS and DFHFCRS when processing VSAM files.

A VSWA survives as long as the RPL needs to survive to complete the VSAM request; for example, it survives from READ UPDATE to the REWRITE request.

The address of the VSWA is held in the file request thread element (FRTE) in the FRT_WORK_AREA_ADDRESS field.

Storage for the VSWA is acquired from 31-bit storage.

The layout of the VSWA is defined by the DFHVSWAS structure and by the DFHVSWA DSECT.

Modules

This section describes the file control modules. Unless otherwise stated, addressing mode and residency mode are AMODE 31 and RMODE ANY respectively.

There are also a number of modules which make up the coupling facility data tables server. These all have names of the form DFHCFxx.

Figure 50 on page 1613 shows the main file control modules and their interfaces.

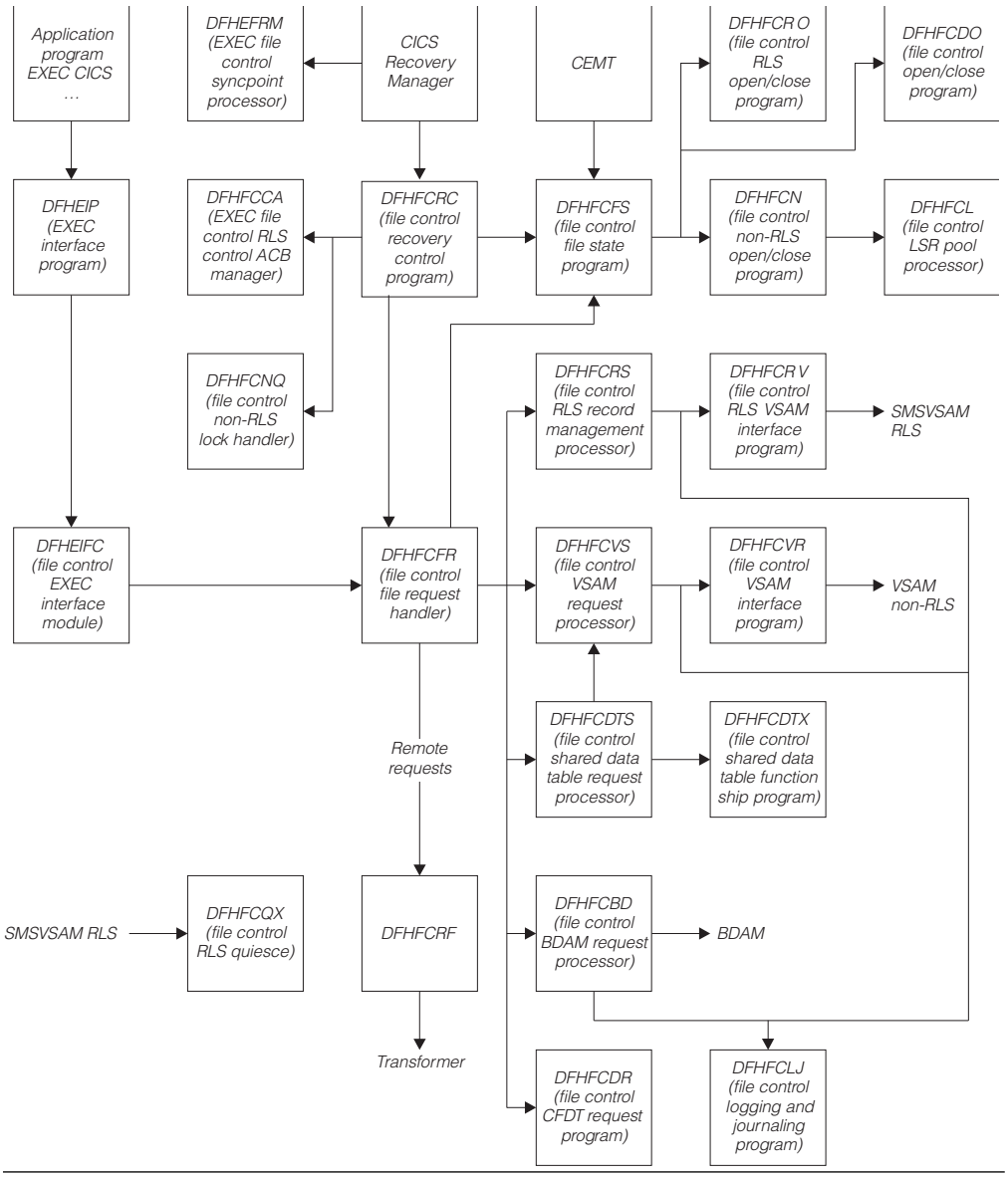


Figure 50. Main file control modules and their interfaces

DFHEIFC (file control EXEC interface module)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHEIFC. Stored in the CSA in a field named CSAEIFC.

Purpose

DFHEIFC is DFHEIP's file control interface. It routes requests to the file control file request handler, DFHFCFR.

Called by

DFHEIP exclusively.

Inputs

The EIEI parameter list, as defined by the DFHEIEIA DSECT.

Outputs

Updated EIEI parameter list, with completed EIB.

Operation

- Call user exit XFCREQ.
- Call file control request handler DFHFCFR.
- Call user exit XFCREQC.

How loaded

At CICS startup, as part of the building of the CICS nucleus. The nucleus is built by DFHSIB1, which uses its nucleus build list to determine the content and characteristics of the CICS nucleus.

DFHFCAT (file control catalog manager)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCAT. The entry point address is held in FC static storage in a field named FC_FCAT_ADDRESS, which is set by DFHFCRP when it loads DFHFCAT.

Purpose

The file control catalog manager is part of the file control component. This program processes inquire and update requests on the state of the backup while open (BWO) attributes in the ICF catalog for VSAM data sets and inquire on the quiesce state in the ICF catalog. The DFSMS Callable Services interface is used for these operations.

Called by

DFHFCDN
: Get the base data set name for a DSNB that has not yet been validated, update the recovery point, or to set the BWO attributes to a 'forward recovered' state

DFHFCN
: Inquire on the current state of, and to update, BWO attributes during file open processing; and to reset these attributes during file close processing.

DFHFCQI
: Inquire on the quiesce state of a data set.

Inputs

The FCAT parameter list, as defined by the DFHFCATA DSECT, is created as part of the subroutine call.

The input parameters are:
Data set name
Recovery point

Outputs

Returned in the FCAT parameter list:
Quiesce state
Base data set name
State (fuzzy, sharp)
Response
Reason

Operation

DFHFCAT provides the following functions:

INQ_BASEDSNAME
Gets the base data set name for a specified data set name from the ICF catalog. This function is used when there is not a validated DSN block for the data set.

INQ_CATALOG_QUIESCESTATE
If the level of DFSMS is 1.3 or higher, issues an IGWARLS call to determine the quiesce state of the data set (quiesced or unquiesced).

INQ_DATASET_STATE
Determines the current state of a VSAM data set's BWO attributes in the ICF catalog. If the BWO attributes indicate that the data set is "back level", that is, a backup copy has been restored but not forward recovered, an exception response is returned; otherwise, a state of 'fuzzy' or 'sharp' is returned, indicating whether or not the data set is defined in the ICF catalog as eligible for BWO.

SET_CATALOG_RECOVERED
Updates a VSAM data set's BWO attributes in the ICF catalog to a 'forward recovered' state to indicate that the data set has been forward recovered.

SET_CATALOG_RECOV_POINT
Updates a VSAM data set's BWO attributes in the ICF catalog with the new recovery point.

SET_BWO_BITS_DISABLED
Updates a VSAM data set's BWO attributes in the ICF catalog to show that the data set is no longer eligible for BWO support, and updates the recovery point.

SET_BWO_BITS_ENABLED
Updates a VSAM data set's BWO attributes in the ICF catalog to show that the data set is eligible for BWO support, and updates the recovery point.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCBD (file control BDAM request processor)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCBD. The entry point address is held in FC static storage in a field named FC_BDAM_ENTRY_ADDRESS.

Addressing mode

AMODE 31.

Residency mode

RMODE 24.

Purpose

The BDAM request processor is part of the file control component. It processes access requests to BDAM files.

Called by

DFHFCFR, after having determined that the request is for a BDAM file.

Inputs

The FCFR parameter list, as defined by the DFHFCFRA DSECT. Also, the file control environment, including FC static storage and the FCT.

Outputs

Updated FCFR parameter list.

Operation

Acquires and releases FIOA storage as necessary. Implements BDAM exclusive control requests. Performs record-length and key-length checking. Calls BDAM to perform the I/O request.

Acquires storage, in the correct key subpool, for requests that specify SET.

How loaded

By DFHFCFS, by means of a loader domain call. DFHFCBD is not loaded unless DFHFCFS is called to open a BDAM file and, in doing so, it discovers that DFHFCBD is not yet in storage.

DFHFCCA (file control RLS control ACB manager)

DFHFCCA is the file control RLS control ACB manager. The RLS control ACB is a special ACB required when a commit protocol application such as CICS uses VSAM RLS. FCCA processes requests to register and unregister the control ACB, and all other file control requests to SMSVSAM that have to be made via the control ACB. These requests are:

- IDAREGP (register)
- IDAUNRP (unregister)
- IDARECOV (clear recovery status)
- IDAINQRC (inquire on recovery)
- IDAQUIES (quiesce)
- IDALKREL (release locks, and retain locks marked for retention)
- IDARETLK (mark locks for retention)

DFHFCCA also includes the code for the RLSWAIT exit used by control ACB requests. Whenever CICS issues such a request, VSAM drives the RLSWAIT exit as soon as it is about to transfer control to the SMSVSAM address space. CICS is then able to drive the dispatcher and schedule other CICS tasks whilst the SMSVSAM address space is busy processing the request.

DFHFCDL (file control CFDT load program)

DFHFCDL is attached by DFHFCDO to load a load-capable coupling facility data tavle with records from a source data set.

DFHFCDN (file control DSN block manager)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCDN. The entry point address is held in FC static storage in a field named FC_FCDN_ADDRESS, which is set by DFHFCRP when it loads DFHFCDN.

Purpose

The DSNAME block manager is part of the file control component. This program is called to perform various operations on data set name blocks. These operations include connecting and disconnecting DSN blocks and FCT entries, setting their attributes, and deleting them when no longer required. The program also allows the caller to inspect a particular DSN block or browse a set of blocks. It can also be called to update the backup while open (BWO) attributes in the ICF catalog for VSAM data sets, and to set the quiesce state to normal in all DSN blocks. Finally it can be called to catalog the information in a DSN block to the CICS global catalog.

Called by

DFHAMFC
: Connect a DSN block to a newly created FCT entry

DFHAMPFI
: Connect the DSN block for the CSD to the associated FCT entry

DFHEIQDN
: Connect, disconnect, delete, set attributes, browse, and inquire against DSN blocks in response to external requests; and to update the BWO attributes in the ICF catalog for a VSAM data set to a 'forward recovered' state

DFHEIQDS
: Connect or disconnect DSN blocks and FCT entries in response to external requests

DFHFCLF
: Set the availability attribute to unavailable after a forward recovery log stream failure

DFHFCMT
: Disconnect the DSN block when deleting an FCT entry

DFHFCN
: Connect or disconnect and to catalog a DSN block

DFHFCRC
: Update the recovery point in the ICF catalog for all VSAM data sets that are open for update in non-RLS mode and defined as eligible for BWO support at keypoint time

DFHFCRD
: To reset all quiesce states to normal after an SMSVSAM server failure

DFHFCRO
Connect or disconnect and to catalog a DSN block
DFHFCRP
Connect or reconnect DSN blocks during file control initialization or restart.

Inputs

The FCDN parameter list, as defined by the DFHFCDNA DSECT, is created as part of the subroutine call.

The input parameters include:
Request identifier
Address of FCTE or FCTE token
Data set name
Browse token
Availability status
Type of pointer
Recovery point

Outputs

Output parameters, as part of the FCDN parameter list. Apart from the response, all these are returned on the inquire or browse requests. The parameters include:

Access method
Base data set name
Availability status
DSNB type
File count
DSNB valid status
Lost locks status
Forward-recovery log stream name
Forward-recovery log ID
Recovery status
Response
Reason

Operation

- Connect:

  The inputs are a data set name and an FCTE pointer or an FCTE token, with an indication of whether the entity to be connected is a base or an object.

  If the FCT entry is already connected, the connection is broken before connecting it to a DSN block representing the new object. The DSN block that is connected can exist already, or DFHFCDN creates a new block before connecting it.

  The request is rejected if it requires an existing connection to be broken, and there are uncommitted updates to the file; that is, there are retained locks.
- Disconnect:

  The connection between the FCT entry and the DSN block is broken. The DSN block remains even if there are no other FCT entries connected to it. The request is rejected if there are uncommitted updates to the file: that is, there are retained locks.
- Delete:

Checks are made to ensure that the DSN block is allowed to be deleted. If the deletion can proceed, the table manager is called to delete the DSN from the DSN index, and the storage domain is called to free the storage.

- Inquire:

  The attributes stored in the DSN block are returned to the caller in the FCDN parameter list.
- Set:

  The availability status is set in the DSN block. The catalog domain is called to catalog the change.
- Start browse, get next, end browse:

  The DSN blocks are browsed in order. For each, the attributes are returned to the caller.
- Catalog:

  The information in a DSN block is cataloged to the CICS global catalog.
- SET_CATALOG_RECOVERED:

  This function is used by DFHEIQDN. DFHFCDN in turn issues a SET_CATALOG_RECOVERED call to DFHFCAT to update the BWO attributes in the ICF catalog for a given VSAM data set to a 'forward recovered' state.
- UPDATE_RECOVERY_POINTS:

  This function is used by DFHFCRC. DFHFCDN in turn issues a SET_CATALOG_RECOV_POINT call to DFHFCAT to update the recovery point in the BWO attributes in the ICF catalog for every data set that is open for update in non-RLS mode and defined as eligible for BWO support.

  The recovery point is the time from which a forward-recovery utility should start applying log records. It is always before the time the last backup was taken. For further information about recovery points and backup while open in general, see the Troubleshooting for recovery processing.
- RESET_ALL_QUIESCE_STATUS:

  This function is used by DFHFCRD. The DSNB table is scanned, and the quiesce status is reset to normal in each DSNB.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCDO (file control CFDT open/close program)

When called using the FCFS parameter list, DFHFCDO performs an equivalent function for coupling facility data table opens and closes as is performed by DFHFCN for non-RLS VSAM files.

When called using the FCDS parameter list, DFHFCDO performs statistics collection for coupling facility data tables, and disconnects from CFDT pools at shutdown.

DFHFCDR (file control CFDT request processor)

DFHFCDR performs an equivalent function for coupling facility data tables as is performed by DFHFCVS for non-RLS VSAM files, and uses the same interface.

DFHFCDTS (file control shared data table request program)

DFHFCDTS performs an equivalent function for CICS-maintained and user-maintained data tables as is performed by DFHFCVS for non-RLS VSAM files and uses the same interface.

DFHFCDTX (file control shared data table function ship program)

DFHFCDTX receives file requests from DFHFCDTS in FCFRR format, converts them into command level interface form and then calls ISP to function ship the request.

The response returned by ISP in the EIB is translated back into an FCFRR response and reason code.

DFHFCDU (file control CFDT UOW calls program)

DFHFCDU encapsulates the processing required to call the coupling facility data tables server for unit of work related operations, such as commit, backout, inquire. It is called via the FCDU parameter list by DFHFCDW and DFHFCDY.

DFHFCDW (file control CFDT RMC program)

DFHFCDW provides a recovery manager connector (RMC) between file control and the coupling facility data tables server, to support 2-phase commit and recovery for recoverable coupling facility data tables. It is called by the CICS Recovery Manager using the RMLK parameter list.

DFHFCDY (file control CFDT resynchronization program)

DFHFCDY performs resynchronization of coupling facility data table pools and links. It is called using the FCDY parameter list by DFHFCDO, DFHFCDR and DFHFCDU.

DFHFCES (file control ENF servicer)

DFHFCES is the file control ENF servicer. It is used to prompt dynamic restart of RLS file control when the SMSVSAM Server becomes available again after an earlier failure. DFHFCES is invoked whenever the MVS Event Notification Facility notifies CICS (via the CICS domain manager ENF support) that SMSVSAM is available.

DFHFCES establishes a transaction environment, and calls DFHFCRR to dynamically restart RLS.

DFHFCFL (file control FRAB and FLAB processor)

DFHFCFL is the File Control FRAB/FLAB processor. It contains a number of functions to process FLAB control blocks belonging to a particular base data set. It processes the functions of the FCFL interface.

The DSNB of the data set is not locked during the processing of the commands. As a FLAB exists, and hence an FCTE, the DSNB cannot be deleted, therefore there is no need to lock the DSNB.

DFHFCFR (file control file request handler)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCFR. Stored in the CSA in a field named CSAFCEP.

Purpose

The central module in the file control component.

Processes file control requests issued by DFHEIFC (requests from application programs), or from other CICS modules (internal CICS file control requests).

Receives and routes file control access-method dependent requests to one of the following:

- DFHFCRS for VSAM RLS files
- DFHFCVS for VSAM non-RLS files
- DFHFCBD for BDAM files
- DFHFCDR for coupling facility data tables
- DFHFCTS for user-maintained data tables
- DFHFCDTS for non-update requests to CICS maintained data table
- DFHFCVS for update requests to CICS-maintained data tables
- DFHFCRF for requests to remote files

Implements TEST_FILE_USER requests.

Routes RESTART_FILE_CONTROL requests to DFHFCVS and DFHFCRS during the file control initialization.

Frees buffers at the request of DFHAPSM when 'short on storage' has been detected.

Performs a CLEAR_ENVIRONMENT when requested by DFHERM, DFHAPLI or DFHUEH. This cleans up file control storage at the completion of a task-related user exit, a user-replaceable program, or a global user exit:

- The FLAB and FRTE chain are scanned to find all FRTEs for the specified environment.
- An ENDBR request is issued to terminate any active browse operation.
- An UNLOCK request is issued for any active READ UPDATE or WRITE MASSINSERT.

Called by

DFHAPLI
AP language interface program
DFHAPSM
AP domain storage notify gate
DFHDMPCA
CSD manager adapter
DFHDTLX
Shared data tables load program
DFHEIFC
File control EXEC interface module
DFHERM
Resource manager interface (RMI) module
DFHFCDL
Coupling facility data tables load program
DFHFCDTS
File control shared data table request processor
DFHFCFR
File control file request handler (a recursive call)
DFHFCRC
File control recovery control program

DFHFCRP
File control restart program
DFHUEH
AP user exit handler.

Inputs

The FCFR parameter list, as defined by the DFHFCFRA DSECT. Also the file control environment, including FC static storage and the FCT.

Outputs

Updated FCFR parameter list.

Operation

Selects on the request type, and passes control to the routine specific to that request.

Performs monitoring.

Obtains a FLAB and FRTE to represent this request, or scans the FLAB and FRTE chains to associate this request with a previous FRTE if required. Some checking for error situations is performed during the scan.

Performs file state checking to determine whether or not a (VSAM or BDAM) request to a file is able to proceed. If file is enabled but closed and is not a request to a remote file, opens it before carrying out the request.

Checks for "privileged" requests.

If the request is not remote, checks the "service request" attributes for the file to determine whether the request can proceed.

Checks the file's access method (VSAM or BDAM as defined in the FCT). If BDAM, calls DFHFCBD to process the request. If VSAM and non-RLS, calls DFHFCVS to process the request. If VSAM and RLS, calls DFHFCRS to process the request. If a data table, calls DFHFCDTS for read requests against a CICS-maintained data table or any request against a user-maintained table, and calls DFHFCVS otherwise (that is, for update and browse requests against a CICS-maintained data table). If the file is remote, calls DFHFCRF to process the request.

On return, performs clean up if required.

How loaded

By DFHSIB1 as part of the CICS nucleus.

DFHFCFS (file control file state program)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCFS. The entry point address is held in FC static storage in a field named FC_FCFS_ADDRESS, which is set by DFHFCRP when it loads DFHFCFS.

Purpose

The file control file state program is part of the file control component.

The program processes requests to enable, disable, open, and close files. Such requests can originate from explicit requests (either CEMT or EXEC CICS SET), from implicit requests (such as implicit open), or from requests made from CICS internal processing.

Close and disable requests are processed in different ways, depending on whether the request has been issued with the WAIT or the NOWAIT option. A request with the WAIT option is treated as a synchronous request, that is, control returns to the requesting program only after all users of the file have completed their use.

A request with the NOWAIT option is treated as an asynchronous request. In this case, the file is marked with the intended state and control is returned immediately.

Called by

DFHAMFC
Enable a newly installed file

DFHDMPCA
Change the state of the CSD

DFHDMRM
Close CSD after an error

DFHDTLX
Close the data set associated with a shared data table

DFHEIQDS
Implement CEMT and EXEC CICS requests

DFHFCDL
Close the data set associated with a coupling facility data table

DFHFCDTS
Close shared data table if remote connection disabled or invalidated

DFHFCFR
Implicit open

DFHFCQU
Close files for quiesce, cancel close for unquiesce, enable files

DFHFCRC
Open files which need backout, and close files at syncpoint

DFHFCRD
Immediate close of RLS files

DFHFCRV
Close files for pending immediate close requests

DFHFCSD
Close files on a normal CICS shutdown

DFHFCU
Open all files with FILSTAT=OPEN coded

DFHFCVS
Open the base, and during empty file or I/O error processing.

Inputs

The FCFS parameter list, as defined by the DFHFCFSA DSECT, is created as part of the subroutine call.

The input parameters are:

Request identifier (open, close, enable, disable, cancel close)
FCTE address
FCTE token
Open options (open base, open for backout)
Close qualifier (close pending, shutdown, immediate close, quiesce, and so on)
Action (wait, do not wait, force)

Outputs

Returned in the FCFS parameter list:

DFHFCN return code
Register 15 return code
VSAM return code

Operation

Before any processing to change the state of a file is carried out, its FCT entry is locked by means of a DFHKC ENQ call. At the conclusion of file state change processing, the FCT entry is unlocked before returning to the caller.

- Enable file.

  DFHFCFS marks the FCT entry as 'enabled', and catalogs the change.

- Disable file.

  If the WAIT option is specified, DFHFCFS tests whether the transaction issuing the request is a current user of the file. If it is, DFHFCFS returns an exception response.

  DFHFCFS next marks the FCT entry entry as 'disabled' and catalogs the change. If the disable request stems from a close request (see later), DFHFCFS also sets the implicit indicator, thereby marking the state as 'unenabled'. However, if this close request originated from DFHFCSD as part of CICS shutdown processing, DFHFCFS does ***not*** mark the state as 'unenabled'.

  Finally, if the WAIT option is specified, the FCT entry is unlocked before waiting for the 'disabled' ECB in the FCT entry to be posted by the transaction that reduces the use count to zero.

- Open file.

  If the file is unenabled (due to a previous close), DFHFCFS enables it and catalogs the new state, unless the open option is open for backout.

  If the file refers to a BDAM data set, DFHFCFS tests whether DFHFCBD is already loaded; if not, it calls loader domain to do so.

  If the file is a data table, DFHFCFS loads and initializes data table services, if this has not been done already on a previous open request.

  DFHFCFS next calls DFHFCN (for non-RLS) or DFHFCRO (for RLS) to perform the physical open. After the file has been successfully opened, its FCT entry is marked accordingly.

For a data table, DFHFCFS issues OPEN and LOAD requests to data table services.

- Close file.

  If there is no close qualifier, the file is first implicitly disabled (as described earlier), taking into account the WAIT or NOWAIT option. The new state is cataloged.

  If the file use count is zero, DFHFCFS calls DFHFCN or DFHFCRO to perform the physical close. After the file has been successfully closed, its FCT entry is marked accordingly.

  An immediate close is issued if the SMSVSAM RLS server fails. The close must wait until there are no requests active in the RLS record management processor. The enablement state of the file is not changed. A close with close qualifier of quiesce is issued to process an RLS quiesce request. The file is unenabled, and the state catalogued.

  For a data table, DFHFCFS issues a CLOSE request to data table services, except in the case of a special type of CLOSE request issued by DFHFCVS for a user-maintained data table, when loading is complete and the source data set is to be closed, but not the table itself.

  For a remote data table, DFHFCFS issues a DISCONNECT request to data table services.

  If the file use count is nonzero, DFHFCFS sets the 'close requested' indicator in the FCT and returns to the caller. Any subsequent transaction that reduces the use count to zero tests the 'close requested' indicator and, if set, performs the actual close.

  When called by DFHFCSD during CICS shutdown, DFHFCFS ensures that files are closed, marks the file as 'closed unenabled' in the FCT, but does ***not*** record this change in the global catalog. This allows implicit file opens on a subsequent restart.

- Cancel close.

  An in-progress close is cancelled if a data set is unquiesced. The close_in_progress flag is reset, any tasks waiting for the file to close are resumed, and the file is re-enabled.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCIN1 (file control initialization program 1)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCIN1. Stored in the CSA in a field named CSAFCXAD.

Purpose

The file control initialization program is part of the file control component. This program initializes file control and starts the file control restart task. It also waits for the restart task to complete, and returns the status of the completion to the caller.

Called by

DFHSII1, as part of CICS initialization.

Inputs

The FCIN parameter list, as defined by the DFHFCINA DSECT.

Outputs

Updated FCIN parameter list.

Operation

Initialize:

- Calls storage manager domain to add a subpool for file control static storage.
- Calls storage manager domain to create the storage for file control static storage.
- Initializes file control static storage.
- Attaches the file control restart task by means of a DFHKC request, with entry point address DFHFCIN2.

WAITINIT:

- Issues a dispatcher domain call to wait on the CICS ECB which indicates that the file control restart task has finished (FC_RECOV_ALLOWED_ACB) in file control static storage.
- On completion of the wait, tests the response and returns to DFHSII1.

How loaded

Link-edited with DFHFCIN2 to form the DFHFCIN module, which is loaded by DFHSIB1 as part of the CICS nucleus.

DFHFCIN2 (file control initialization program 2)

Call mechanism

Attached by DFHFCIN1 as a separate CICS task. Given control by means of the DFHKC TYPE=ATTACH mechanism.

Entry address

DFHFCIN2. Because DFHFCIN2 is link-edited with DFHFCIN1, the entry address is known to DFHFCIN1 at the time the DFHKC TYPE=ATTACH is issued.

Purpose

The file control initialization program is part of the file control component. This program loads and calls the file control restart program (DFHFCRP), to perform file control restart as a separate task.

Called by

CICS task control, after being attached by DFHFCIN1.

Inputs

None.

Outputs

The initialized file control component. Addresses and indicators completed in file control static storage.

Operation

Calls loader domain to acquire (that is, to load) the DFHFCRP program. Stores the entry point address of the loaded module (which is also the load point) in DFHFCIN2's automatic storage in a field named FCRP_ENTRY_ADDRESS.

If the ACQUIRE request failed, calls loader domain to define program and then retries the ACQUIRE request.

Calls DFHFCRP by means of a subroutine call via the kernel.

On successful completion, calls loader domain to release DFHFCRP. On both successful and unsuccessful completion, posts the ECBs FC_NON_RECOV_ALLOWED_ECB and FC_RECOV_ALLOWED_ECB. The success or otherwise of File Control restart is indicated by the flag FCSCMPLT in file control static storage.

On unsuccessful completion, posts the Restart Task ECB complete and returns.

How loaded

By DFHSIB1 as part of the CICS nucleus.

DFHFCIR (file control initialize recovery)

DFHFCIR is the File Control Initialize Recovery Module. It initializes the File Control environment in which recovery after a CICS failure is carried out.

DFHFCIR handles the delivery of recovery data by the CICS Recovery Manager during its scan of the system log at warm or emergency restart, and rebuilds the file control structures that represent units of work that were in-flight or shunted when CICS terminated.

During its log scan, Recovery Manager calls File Control's recovery gate, which invokes the module DFHFCRC. DFHFCRC passes the calls through to DFHFCIR via a kernel subroutine call. The calls are the RMDE functions START_DELIVERY, DELIVER_RECOVERY, DELIVER_FORGET and END_DELIVERY.

DFHFCL (file control shared resources pool processor)

Call mechanism

BALR, obtaining LIFO storage on entry.

Entry address

DFHFCLNA. DFHFCL is, together with DFHFCN and DFHFCM, link-edited with DFHFCFS. All calls to DFHFCL are made from DFHFCN; the entry point address is known to DFHFCN from the link edit.

Purpose

The shared resources pool processor is part of the file control component.

This program is called at file open time to create a specific local shared resources pool if it does not exist. It is also called to delete a specific pool when the last file to use the pool is being closed.

The size and characteristics of the pool being built are obtained either from information in the resource definition or, if that information has not been provided, from the best information available to DFHFCL at the time of the open.

Called by

DFHFCL is called exclusively by DFHFCN.

Inputs

The FCLPARAM parameter list, created in DFHFCN's automatic storage and addressed by register 1 on the call.

The input parameters are:

Request identifier (build, delete)
LSR pool number

Outputs

Returned in the FCLPARAM parameter list:

DFHFCL return code
BLDVRP/DLVRP return code
VSAM return code

Operation

If the request is for LSR pool creation, DFHFCL first checks whether the SHRCTL block includes specifications for the number of strings, maximum key length, and the number of virtual and hyperspace buffers of each of the eleven sizes in the pool. If these values are known, DFHFCL sets up the BLDVRP parameter list and creates the pool by issuing the BLDVRP macro.

If some or all of the pool characteristics are not specified in the SHRCTL definition, DFHFCL calculates the pool requirements from the information in the FCT and the VSAM catalog.

Each FCT entry is inspected to find whether it is to be included in the pool being built. If so, its DSNAME is determined and this is used to obtain data set characteristics from the VSAM catalog. The information required for the BLDVRP macro is accumulated in the SHRCTL block and the pool is built from these values.

If the request is for LSR pool deletion, DFHFCL first obtains the VSAM statistics for the pool and saves them in the SHRCTL block. These statistics are unobtainable after the pool has been deleted.

DFHFCL next deletes the specified pool by issuing a DLVRP macro.

Finally, DFHFCL sends pool statistics to the statistics domain as unsolicited data.

How loaded

As a constituent part of DFHFCFS, which is loaded by DFHFCRP as part of file control initialization.

DFHFCLF (file control log failures handler)

DFHFCLF provides control of long term logger failures for File Control. It is called in the event of a failure of a general log stream, which will be either the forward recovery log for a data set or the autojournal for a file.

The CICS Log Manager invokes DFHFCLF when an MVS log stream being used for forward recovery or file autojournalling suffers a long term failure. The call is made using the LGGL ERROR function.

When file control opens a forward recovery log stream or an autojournal, it will register this call back gate to the Log Manager by specifying FCLF as the file control error gate.

When called, DFHFCLF takes action to ensure that the log stream failure causes minimum damage. For a forward recovery log failure it closes all files open against the data set using that forward recovery log (across the sysplex for a data set accessed in RLS mode) and issues a message advising that a new backup copy should be taken. For an autojournal it closes the file using that autojournal and issues a warning message.

DFHFCLJ (file control logging and journaling program)

DFHFCLJ is the file control logging and journaling program. It is called to perform logging for transaction backout and forward recovery, to write to journals for autojournal requests and to write to the log of logs.

Records are written to the system log using the RMRE APPEND function, and optionally forced using the RMRE FORCE function. Records are written to forward recovery logs and autojournals using the LGGL WRITE function, and to the log of logs using the LGGL WRITE_JNL function.

Replication logging is a form of logging in CICS where updates to VSAM and RLS files in one location are replicated on another site. In this way, multi-site operations can reduce their recovery time objective (RTO) to near zero. For more information, see the FCLJ gate functions in Application Manager Domain's specific gates.

DFHFCMT (file control table manager)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCMT. The entry point address is held in FC static storage in a field named FC_FCMT_ADDRESS, which is set by DFHFCRP when it loads DFHFCMT.

Purpose

The file control table manager is part of the file control component. This program is called to add, delete, and set FCT entries, and to return attributes of an FCT entry (inquire).

Called by

DFHAMFC
Inquire on, add, or delete a newly created FCT entry to the system

DFHAMPFI
Add the entry in the FCT for the CSD to the system

DFHDMPCA
Inquire on and set the attributes of the FCT entry for the CSD

DFHEDFX
Inquire on the attributes of an FCT entry

DFHEIQDS
Inquire on or set the attributes of FCT entries, or delete an FCT entry.

Inputs

The FCMT parameter list, as defined by the DFHFCMTA assembler DSECT, is created as part of the subroutine call.

The input parameters are:

Common parameters:
- File name
- String number
- Journal ID
- Recovery characteristics
- Journaling characteristics
- Enablement status
- Open time
- Data set disposition
- Service request attributes
- Record format
- Number of data buffers
- Number of index buffers
- Whether to catalog the FCT entry

VSAM-specific parameters:
- VSAM password
- Empty status
- Data set name sharing
- LSR pool ID
- Base name
- Forward recovery log ID
- BWO eligibility
- RLS access mode
- Read integrity

BDAM-specific parameters:
- Exclusive control

Outputs

Output parameters, as part of the FCMT parameter list. Apart from the response, all these are returned on the inquire or browse requests. The output parameters are:

Common parameters:
- File type
- String number
- Record size
- Key length
- Key position
- Recovery characteristics
- Journaling characteristics
- Enablement status
- Open status
- Open time
- Data set type
- Data set disposition
- Data set name
- Base data set name
- Service request attributes
- Record format
- Block format
- Access method
- Remote name
- Remote system

VSAM-specific parameters:
- VSAM password
- Empty status
- Object type
- Data set name sharing
- Number of data buffers
- Number of index buffers
- Number of active strings
- LSR pool ID
- Whether using shared resources
- Forward-recovery log ID
- RLS access mode
- Read integrity

BDAM-specific parameters:
- Block size
- Block key length
- Relative address form
- Exclusive control
- Response
- Reason

Data Table specific parameters:
- Table type
- Table size

Operation

- Add:

Storage for the new FCT entry is obtained out of the VSAM FCT storage subpool.

The new FCT entry is completed by filling in the information from the caller's parameter list.

The name of the new FCT entry is added to the TMP index.

Finally the information in the new entry is written to the CICS global catalog if required.

- Delete:

  The request is rejected if there are uncommitted updates for the file; that is, there are retained locks. DFHTMP is called to locate and quiesce the FCT entry.

  Any DSN block that is connected to the FCT entry is disconnected.

  The FCT entry name is deleted from the TMP index.

  The storage for the FCT entry is freed. In the case of a BDAM FCT entry, its DCB storage is also freed.

  Any catalog entries for the FCT entry are deleted.

- Set:

  DFHTMP is called to locate the FCT entry.

  The request is rejected if there are uncommitted updates for the file; that is, there are retained locks.

  If the FCT entry is not marked ‘closed' and ‘disabled' (or ‘unenabled'), the request is rejected.

  Changes are made to the information in the FCT according to the caller's parameter list.

  Finally the changes are recorded by writing them to the CICS global catalog.

- Inquire:

  DFHTMP is called to locate the FCT entry.

  The attributes are returned in the FCMT parameter list.

- Connect:

  DFHTMP is called to locate the FCT entry.

  The connect count is incremented. The FCT token is returned to the caller.

- Disconnect:

  DFHTMP is called to quiesce the FCT entry.

  A check is made to ensure that the file is closed and disabled (or unenabled). If the check fails, an error is returned to the caller.

  The connect count in the FCT is cleared and a call is again made to DFHTMP to release the quiesce.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCN (file control open/close program)

Call mechanism

BALR, obtaining LIFO storage on entry.

Entry address

DFHFCNNA. DFHFCN is link-edited with DFHFCFS. All calls to DFHFCN are made from DFHFCFS; the entry point address is known to DFHFCFS from the link-edit.

Purpose

The file control open/close program is part of the file control component.

This program performs the physical opening and closing of files by making the corresponding requests to VSAM or BDAM. Associated with these operations are a number of further activities that must be completed before control is returned to DFHFCFS.

These activities include:

- Dynamic allocation of the file
- Empty file checking
- Dynamically setting up ACB fields in advance of the VSAM open
- Copying into file-control control blocks VSAM information about the file that is available after the open
- Inquiring on, and updating, the VSAM data set's backup while open (BWO) attributes in the ICF catalog for a file that is defined as eligible for BWO support if the appropriate prerequisite software levels have been installed
- On close, deallocating the file if necessary and clearing the file control information related to the file
- Resetting a VSAM data set's BWO attributes in the ICF catalog during close processing.

Called by

DFHFCFS, exclusively.

Inputs

The FCSPARMS parameter list, created in DFHFCFS's automatic storage and addressed by register 1 on the call.

The input parameters are:

FCTE address
Request identifier

Outputs

Returned in the FCSPARMS parameter list:

DFHFCN return code
Register 15 return code
VSAM return code
Base data set name
Recovery attributes of base

Operation

Execution of the DFHFCN code is serialized. This is done by DFHFCFS issuing a DFHKC ENQ before calling DFHFCN, and a DFHKC DEQ after calling DFHFCN. As a consequence, only a single open or close request to any file can be in progress at any time, and multiple concurrent requests are single-threaded.

- The main actions when processing an open request are as follows:
  1. If the file is being opened for update and any type of autojournalling is specified on the file definition, then the autojournal log stream is opened, via a call to DFHLGGL.
  2. The file is tested to determine if it is allocated to the job by means of a JCL statement or is to be allocated dynamically.

     If the file is already allocated, any existing DSN block to which it may be connected is disconnected and a new block with the actual DSNAME is connected. Connecting and disconnecting of DSNAME blocks is always performed by calling DFHFCDN.

     If the file is not already allocated, it is at this point dynamically allocated to the DSNAME in the DSNAME block to which it is connected.

     In the case of a VSAM file, the file's data set name is used to issue appropriate SHOWCAT and LOCATE instructions to determine relevant information from the VSAM catalog about the data set that the file represents. In particular, the following are obtained:

     Base/path indicator
     Base data set name
     Attributes of the data set
     Key length of the base
     Relative key position of base key
     Maximum record length
     Control interval size
     Share options
     High RBA
  3. The data set is checked to determine if it is empty (high RBA is zero) or is to be emptied.

     The ‘load' mode indicator is set on.
  4. DFHFCDN is now called to connect the FCT entry to a DSNAME block for the base cluster (which may be the existing allocation DSNAME block, or may need to be newly created, or may already exist and need only be pointed to from the FCT). The base cluster's attributes, as obtained from the VSAM catalog, are stored in the base cluster block.

     The file's recovery characteristics are checked against any that may already have been stored in the base cluster block and, if they have not yet been set up, are saved there. Any conflict with the stored values is handled. In some cases the new value overrides the old one, in others an error is returned.

     During this processing, if this is the first open for update for a file associated with this particular data set:

     a. a call is made to the VSAM callable interface IGWARLS, in order to get any recovery attributes that may be defined in the VSAM catalog. If they are present, then they override any values in the FCT entry.
     b. if forward recovery logging is specified, the forward recovery log stream is opened, using either the log stream name from the VSAM catalog, or a log stream name derived from the id specified in the file definition.

In the case of an entry sequenced data set or a path to an ESDS, the next available RBA in the data set is determined and stored in the base cluster block.

5. If the file uses a shared resources (LSR) pool, and if the pool is not currently in existence, DFHFCL is called to determine the pool's characteristics and to build it.
6. Before opening a VSAM file, any STRNO, BUFND, or BUFNI parameters that may have been specified in the JCL DD statement are copied to the FCT entry (for LSR opens, these are ignored). The ACB is now created and its various options and parameters filled in from information in the FCT entry. The OPEN is finally completed by a call to VSAM.
7. If the file refers to a BDAM data set, the assembled DCB is used for the open request and no dynamic setting of DCB options is carried out.
8. After the VSAM file has been successfully opened, certain file attributes are obtained from VSAM and are stored in the FCT entry. These include:

   Key length
   Relative key position
   Base/path/AIX indicator
   KSDS/ESDS/RRDS/VRRDS indicator
   Number of strings required for an update operation.
9. For a file opened for update against a VSAM base data set when the update use count in the DSNB for this data set is zero, the BWO attributes in the ICF catalog are validated to find their current state. This is done by making an INQ_DATASET_STATE call to DFHFCAT, regardless of whether the file is defined to CICS as eligible for BWO support.

   The file open request is rejected if one of the following is true:

   a. The BWO attributes in the ICF catalog show ***either*** that the data set is "back level", that is, a backup copy has been restored but not forward recovered, ***or*** that either the catalog or the data set has been corrupted.
   b. The BWO attributes in the FCT entry conflict with those defined in the DSNB, that is, the file has already been opened with different attributes since the DSNB was created.

   If the file is defined to CICS as eligible for BWO support, the BWO attributes in the ICF catalog are updated by making a SET_BWO_BITS_ENABLED call to DFHFCAT.

   However, if the file is not defined to CICS as eligible for BWO support, but the BWO attributes in the ICF catalog currently show that the VSAM base data set is eligible for BWO support, the BWO attributes in the ICF catalog are disabled by making a SET_BWO_BITS_DISABLED call to DFHFCAT, and CICS issues a warning message.

   Note: The ICF BWO attributes are a property of a VSAM sphere; therefore, the VSAM base data set and alternate index path definitions should be consistent. For a general description of the CICS backup while open (BWO) facility, see Troubleshooting for recovery processing.
10. The base DSNB, and path DSNB if this is a path, are marked as validated and catalogued.

- The main actions when processing a close request are as follows:
  1. If the close request is for the last file that was opened for update against a VSAM base data set and the file is defined to CICS as eligible for BWO support, the BWO attributes in the ICF catalog are reset so that BWO

support is no longer enabled. This is done by making a SET_BWO_BITS_DISABLED call to DFHFCAT.

2. Before performing the access method close for a VSAM file, the number of accumulated EXCPs is obtained by making a call to VSAM and is saved in the FCT entry ready to be sent to the statistics domain as part of the file statistics.
3. A CLOSE request is then made by issuing the appropriate (VSAM or BDAM) macro.
4. The ACB storage is freed, and certain fields in the FCT entry which are no longer valid are cleared.
5. File statistics and data table statistics, if any, are sent to the statistics domain as unsolicited data.
6. If the file being closed uses shared resources, and if it is the last to have been closed in its LSR pool, DFHFCL is called to delete the pool.
7. If the file was dynamically allocated at open time, it is deallocated, leaving a pointer to the DSNAME block in the FCT entry.
8. If the file had an autojournal, then the autojournal log stream is closed.
9. If the base data set was forward recoverable, and its use count is non-zero, then the forward recovery log stream is closed.

How loaded

As a constituent part of DFHFCFS, which is loaded by DFHFCRP as part of file control initialization.

DFHFCNQ (file control non-RLS lock handler)

DFHFCNQ is the file control non-RLS lock handler. It is called using the FCCA RETAIN_DATASET_LOCKS interface to retain locks in cases of backout failure. It is called using the NQNQ INTERPRET_ENQUEUE interface to interpret File Control locks for presentation purposes.

Lock retention

When DFHFCRC encounters a failure during an attempt to backout a unit of work it must retain all record locks held by that UOW for the failing data set. It issues an FCCA RETAIN_DATASET_LOCKS request to DFHFCCA for RLS access data sets and to this DFHFCNQ for non-RLS access data sets.

Lock name interpretation

Non-RLS locks include record locks for all file types, and for VSAM files, mass-insert range locks, load mode locks and ESDS WRITE locks. Each lock belongs to one of some half dozen or so pools created by DFHFCRP during CICS initialization. DFHFCNQ is called using the NQNQ INTERPRET_ENQUEUE interface and is passed the enqueue pool name and the lock identifier. The name of pool to which a lock belongs is sufficient information to allow the identifier to be parsed and its constituents returned to the caller.

The pool names and lock constituents are:

- FCDSRECD - Data set name and record identifier - for VSAM and CICS-maintained data tables
- FCFLRECD - File name and record identifier - for BDAM and user-maintained data tables

- FCDSRNGE - Data set name and record identifier - VSAM range locks
- FCDSLDMD - Data set name - VSAM load mode locks
- FCDSESWR - Data set name - VSAM ESDS WRITE locks
- FCFLUMTL - File name - UMT load locks

DFHFCOR (file control offsite recovery completion)

DFHFCOR is the file control RLS offsite recovery completion transaction.

Transaction CFOR is attached when CICS detects that is has completed its RLS offsite recovery processing. RLS offsite recovery is only performed when OFFSITE=YES is specified as a system initialization override. CFOR may be attached either during RLS warm or emergency restart (if there is no RLS offsite recovery work to be performed) or during file control commit processing (if the commit was for the last remaining item of RLS offsite recovery work).

DFHFCOR issues message DFHFC0575 and awaits an operator reply. When the reply is received, it enables RLS access for new transactions.

DFHFCQI (file control RLS quiesce initiation)

DFHFCQI is the RLS Quiesce Initiation module. It provides code to initiate a quiesce request against a base data set. It also provides code to inquire on the quiesce state of a base data set, and to complete a quiesce request against a base data set. Quiesce initiations are issued by the CICS API, or by CICS internally, or by CICS internally cancelling certain in-progress quiesce operations. Quiesce inquiries are issued via the CICS API. Quiesce completions are issued by CICS internally.

DFHFCQR (file control quiesce receive transaction)

DFHFCQR is the VSAM RLS Quiesce Receive module, running under a dedicated CFQR system transaction. It provides code to take quiesce requests from the CICS VSAM RLS quiesce exit and pass them to DFHFCQU for processing. As DFHFCQR runs under a system transaction, it has full transaction environment which enables it to invoke API-capable global user exits, or to call parts of file control that reference the TCA.

DFHFCQS (file control RLS quiesce send transaction)

DFHFCQS is the VSAM RLS Quiesce Send module, running under a dedicated CFQS system transaction. It provides code to take quiesce requests from another task and pass them to SMSVSAM. As DFHFCQS runs under a system transaction, it has full transaction environment which enables it to invoke API-capable global user exits, or to call parts of file control that reference the TCA. DFHFCQS is called from DFHFCQT, the quiesce system transaction module, if the transaction id under which DFHFCQT was started is 'CFQS'.

DFHFCQT (file control RLS quiesce common system transaction)

DFHFCQT is the file control RLS quiesce common system transaction.

There are two file control system transactions dedicated to RLS quiesce processing: CFQS and CFQR. CFQS sends quiesce requests to SMSVSAM in order to initiate the quiesce or unquiesce of a data set throughout the sysplex. CFQR receives quiesce requests from VSAM RLS and performs the quiesce processing required for the CICS region concerned. These transactions share a common top-level program, DFHFCQT.

There is no DFHFCQT parameter list. The action DFHFCQT takes depends on the transid of the transaction it is running under. If it is CFQS then DFHFCQS

SEND_QUIESCES is called. If it is CFQR then DFHFCQR RECEIVE_QUIESCES is called. If DFHFCQS or DFHFCQR subsequently fail with a disastrous error, control is returned to DFHFCQT and a transaction abend is issued, having first re-attached the transaction concerned to ensure that RLS Quiesce support is not lost for ever.

DFHFCQU (file control RLS quiesce processor)

DFHFCQU is the RLS Quiesce Process module. It processes quiesce requests received from SMSVSAM via the quiesce exit mechanism.

DFHFCQX (file control RLS quiesce exit)

DFHFCQX is the RLS Quiesce Exit module. It is called by SMSVSAM whenever the CICS region concerned is required to perform processing for a quiesce request.

The quiesce exit is specified on the RLS control ACB EXLST. The exit initiates processing and returns to VSAM. It must not issue any VSAM requests. It is scheduled as an IRB on the TCB that registered the RLS control ACB. Because of the environment DFHFCQX cannot issue CICS requests. GTF tracing is used to trace entry, exit and any errors.In addition, timestamps are made on entry to and exit from DFHFCQX, and are stored in fields FC_DFHFCQX_ENTRY_STCK and FC_DFHFCQX_EXIT_STCK respectively of the File Control Static area.

On entry to DFHFCQX, register 1 contains the address of a VSAM structure mapped by IFGQUIES which defines the quiesce request. The processing of the quiesce request is performed by the CFQR long-running system transaction (DFHFCQR). To communicate the quiesce to CFQR, DFHFCQX creates an FC Quiesce Receive Element (FCQRE) to describe the request, and adds it to a chain in file control static storage, posting an ECB associated with the chain also in FC static.

DFHFCRC (file control recovery control program)

DFHFCRC provides recovery control for file control. All calls from the Recovery Manager domain to file control come through DFHFCRC.

DFHFCRC is called by the Recovery Manager domain to participate in syncpoint and in warm and emergency restart.

Early on during startup File Control registers as a client of the CICS Recovery Manager. During File Control initialization, File Control will add its recovery gate to the kernel, specifying DFHFCRC as the entry point, and then declares the recovery gate to the CICS Recovery Manager via an RMCD SET_GATE call.

At syncpoint, a resource owner such as File Control may be called either

1. to prepare, optionally followed by shunt-unshunt pairs, followed either by calls to backout (as in 2) or a call to commit.
2. to backout, which involves start_backout, optional delivery of backout data, and end_backout, followed by prepare and commit, optionally followed by backout retries (which consist of shunt-unshunt pairs followed by the start_backout - delivery of backout data - end_backout - prepare - commit sequence).

At warm or emergency restart, a resource owner such as File Control will be called with start_delivery, optional deliver_recovery and deliver_forget calls, followed by end_deliver.

The Recovery Manager functions processed by DFHFCRC are:

- RMRO PERFORM_PREPARE

- RMRO PERFORM_COMMIT
- RMRO START_BACKOUT
- RMRO DELIVER_BACKOUT_DATA
- RMRO END_BACKOUT
- RMRO PERFORM_SHUNT
- RMRO PERFORM_UNSHUNT
- RMKP TAKE_KEYPOINT
- RMDE START_DELIVERY
- RMDE DELIVER_RECOVERY
- RMDE DELIVER_FORGET
- RMDE END_DELIVERY

DFHFCRC performs different processing depending on the function with which it has been called:

PERFORM_PREPARE

Any active VSAM requests are terminated, and a vote of READ_ONLY is returned if the unit of work did not make any recoverable file control updates, a vote of YES if the prepare was successful, or a vote of NO otherwise.

PERFORM_COMMIT

For a forwards syncpoint, any changes made by the unit of work to recoverable user-maintained data tables are committed. For a backwards syncpoint, locks for any backout-failed data sets are retained. All other locks are released.

On transaction termination, the FLABs and FRAB are freed unless there are FLABs marked for retention. On an intermediate syncpoint, various flags in the FLABs and FRAB are reset to indicate that a commit has been performed.

START_BACKOUT

Any active VSAM requests are terminated, and any changes made by the unit of work to recoverable user-maintained data tables are backed out.

DELIVER_BACKOUT_DATA

The recoverable file control change represented by the log record delivered to DFHFCRC is backed out via calls to DFHFCFR which reverse the update. The change is not backed out if the unit of work has already suffered a backout failure for the data set, or if the data set is in a 'non-RLS update permitted' state, or if this call is being made as part of a CEMT or EXEC CICS SET DSNAME RESETLOCKS request.

If a failure occurs during the backout, then backout failure processing is carried out.

END_BACKOUT

Under normal conditions there should be no processing required at END_BACKOUT, but it is conceivable that there might be outstanding active VSAM requests to be terminated.

PERFORM_SHUNT

The failed parts of the unit of work's file control structures are put into a condition to survive without an executable transaction environment. This involves retaining any FLABs that are marked for retention, which will allow files to be closed, but not to be reallocated to a different data set.

If this is an intermediate syncpoint, and the shunt is due to a failure in phase 2 of syncpoint, the transactional parts of the unit of work are copied into a new control structure to be passed to the follow-on unit of work. A new FRAB is acquired to anchor this control structure. If this is transaction termination, or the shunt is due to a failure in phase 1 of syncpoint, the transactional parts are cleaned up.

PERFORM_UNSHUNT

The file control structures are converted back into a condition suitable for a unit of work that is in an executable state. Retained FLABs for the unit of work are restored.

TAKE_KEYPOINT

DFHFCRC is called when CICS takes a keypoint, to perform processing required by BWO backup on non-RLS data sets. This involves the writing of a set of 'tie up records' and the calculation of a new BWO recovery time.

START_DELIVERY

DFHFCIR is called to process the call.

DELIVER_RECOVERY

DFHFCIR is called to process the call.

DELIVER_FORGET

DFHFCIR is called to process the call.

END_DELIVERY

DFHFCIR is called to process the call.

DFHFCRD (file control RLS clean up transaction)

As soon as CICS detects an SMSVSAM server failure, it runs program DFHFCRD under transaction CSFR to perform clean up.

Following the server failure all current RLS ACBs become unusable. DFHFCRD scans a chain of files open in RLS mode, which is anchored from file control static storage and call DFHFCFS to perform an IMMEDIATE_CLOSE for each open file.

DFHFCRD then waits:
1. for the last file to close,
2. once the last file has closed, for SMSVSAM to complete any residual requests against the RLS control ACB.

When both these events have occurred, DFHFCRD calls DFHFCCA to perform UNREGISTER_CONTROL_ACB processing in order to clean up the CICS and VSAM state with respect to the control ACB.

DFHFCRD finally posts an ECB which allows dynamic RLS restart to go ahead. Dynamic RLS restart cannot start until DFHFCRD has completed clean up and posted this ECB.

DFHFCRF (file control function shipping interface module)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

FC_FCRF_ADDRESS stored in FC Static Storage.

Purpose

DFHFCRF is the function shipping interface module. It is called by the access method independent module DFHFCFR for record management requests (e.g. reads, writes, rewrites, etc.) that are to be directed to files that are defined as remote.

DFHFCRF is called with the FCFR parameter list. From this it constructs an FCRF parameter list, which is subsequently passed to DFHISP and, in turn, either to DFHXFX (the MRO transformer) or to DFHXFFP (the ISC transformer).

DFHFCRF executes the following requests from the DFHFCFRR parameter list:

- Simple read requests
  - READ_INTO and READ_SET
- The read update family
  - READ_UPDATE_INTO and READ_UPDATE_SET
  - REWRITE
  - REWRITE_DELETE
  - UNLOCK
- The browse family
  - START_BROWSE
  - RESET_BROWSE
  - READ_NEXT_SET, READ_NEXT_INTO, READ_PREVIOUS_SET, READ_NEXT_UPDATE_SET, READ_NEXT_UPDATE_INTO, READ_PREVIOUS_UPDATE_SET, and READ_PREVIOUS_UPDATE_INTO
  - END_BROWSE
- Write requests
  - WRITE
- Delete requests
  - DELETE

Called by

DFHFCFR, the File Control file request handler.

Inputs

The FCFR parameter list, as defined by the DFHFCFRA DSECT.

Outputs

The FCRF parameter list, as defined by the DFHFCRFA DSECT.

Operation

Traces module entry.

Checks for an explicit SYSID specified on the request and sets the remote system and remote file name in the DFHFCRF parameter list ready for function shipping.

Increments statistics for the type of request.

Checks request specific parameters

Ships the request.

Handles return codes.

Finally, traces the module exit.

How loaded

By FCRP at file control initialization.

DFHFCRL (file control share control block manager)
Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCRL. The entry point address is held in FC static storage in a field named FC_FCRL_ADDRESS, which is set by DFHFCRP when it loads DFHFCRL.

Purpose

The file control share control block manager is part of the file control component.

This program modifies the CICS specification of a shared resources pool. The changes are allowed to be made only when the actual pool is deleted.

Called by

DFHAMFC, when installing an LSR pool defined by RDO.

Inputs

The FCRL parameter list, as defined by the DFHFCRLA DSECT, is created as part of the subroutine call.

The input parameters are:

Request identifier
Pool identifier
Number of strings
Maximum key length
Share limit
Buffer characteristics

Outputs

The response and reason codes only. These are returned in the FCRL parameter list.

Operation

The SHRCTL block for the specified pool is addressed. A test is made to determine whether or not the pool is currently built; if it is built, the request is rejected with an error response.

The pool characteristics specified in the input parameter list are included in the SHRCTL block.

Finally the information in the SHRCTL block is written to the CICS global catalog.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCRO (file control RLS open/close program)

DFHFCRO performs an equivalent function for RLS opens and closes as is performed by DFHFCN for non-RLS access mode.

DFHFCRP (file control restart program)

Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCRP. This address is needed only by DFHFCIN2 during initialization; it is therefore not saved in FC static storage.

Purpose

The file control restart program is part of the file control component. This program creates a file control component on a cold or initial start of CICS, or re-creates it after a warm or emergency start. For a warm or emergency start, the intention is to reconstruct the identical file control environment which was in effect at the time of the previous CICS termination.

Called by

DFHFCIN2, during file control initialization.

Inputs

None.

Outputs

The restarted file control component. File control static addresses and indicators are set up. DFHFCRP's response and reason codes are set in the parameter list defined by DFHFCRPA DSECT.

Operation

Calls loader domain to define (if necessary) and acquire (load) the following file control programs: DFHDTINS, DFHFCAT, DFHFCCA, DFHFCDN, DFHFCD2, DFHFCES, DFHFCFL, DFHFCFS, DFHFCIR, DFHFCLF, DFHFCLJ, DFHFCMT, DFHFCNQ, DFHFCQI, DFHFCQU, DFHFCQX, DFHFCRC, DFHFCRL, DFHFCRO, DFHFCRR, DFHFCRS, DFHFCRV, DFHFCSD, DFHFCST, and DFHFCVS.

Adds gates to the kernel for recovery control, ENF services, and log stream failure notification.

Calls storage manager domain to add (create) the following storage subpools: file control general below 16MB, VSAM FCTE, BDAM FCTE, ACB, DCB, SHRCTL, DSN, FFLE, FRAB, FRTE, FLLB, FLAB, RPL, IFGLUWID, file control fixed-length buffer storage. Calls the NQ domain to add (create) enqueue subpools for: data set record NQs, file record NQs, range NQs, load mode NQs, ESDS write NQs, and UMT loading NQs.

Calls DFHTMP to create TMP primary indexes for the FCT, AFCT, and DSN tables, and a TMP secondary index for the DSN table.

If RLS is supported (correct level of DFSMS, and RLS=YES SIT parameter) initializes the CSFR, CFQS, CFQR and CFOR tasks, registers file control's interest in the SMSVSAM ENF signal by a LISTEN call to DFHDMEN, and calls DFHFCRR to restart RLS.

On a warm or emergency start:

- Determines installation levels of the MVS/Data Facility Product (MVS/DFP) (or DFSMS), the Data Facility Hierarchical Storage Manager (DFHSM), and the Data Facility Data Set Services (DFDSS) for VSAM backup while open (BWO) support.
- Restores DSNAME blocks from the CICS global catalog, recreating a DSN control block in the DSN subpool storage. For each block, adds its DS name to the TMP primary index, and adds its DS number to the TMP secondary index.
- Restores VSAM file entries from the CICS global catalog. For each entry, adds its file name to the TMP FCT index.
- Restores BDAM file entries from the CICS global catalog. For each entry, adds its file name to the TMP FCT index. Further, for each entry, restores the BDAM DCB from the catalog and copies it to an entry in the DCB storage subpool.
- Restores DSNAME references from the CICS global catalog. For each entry, locates its FCTE and invokes DFHFCDN to connect the FCTE to its DSN block.
- Restores SHRCTL blocks from the CICS global catalog.

On a cold start:

- As for a warm or emergency start, determines installation levels of MVS/DFP, DFHSM, and DFDSS for VSAM backup while open (BWO) support.
- Purges the CICS global catalog of all FCTEs, SHRCTL blocks, DSNAME references, AFCTEs, and BDAM DCBs.

- Calls the loader domain to load the FCT specified by the FCT system initialization parameter.
- Builds all eight SHRCTL blocks, using any information that may have been specified in the loaded FCT. Writes the blocks to the CICS global catalog.
- For each file control table entry in the loaded FCT, creates an FCT entry in the FCT storage subpool, copies the information to it, adds the file name to the TMP index, and writes the table entry to the CICS global catalog.
- Calls the loader domain to delete the previously loaded FCT.

Indicates file control restart complete for non-recoverable business by setting FC_NON_REV_ALLOWED_ECB on.

Sends message to inform that file control restart is complete.

If all was successful, turns on the FCSCMPLT flag in FC static.

Finally, posts the FC_RECOV_ALLOWED_ECB in FC static.

How loaded

By the file control initialization module 2, DFHFCIN2, and deleted after it has completed.

DFHFCRR (file control RLS restart)

DFHFCRR is used to restart the RLS component of File Control. It is called whenever CICS is restarted and after any total RLS failure. DFHFCRR is also called whenever a resource can be made available again after earlier failures have been rectified, and after recovery from Lost Locks.

DFHFCRR is invoked whenever CICS is restarted (COLD, WARM or EMERGENCY) by DFHFCRP, and following any total RLS failure (DYNAMIC restart) by DFHFCES.

DFHFCRR is also called to retry work which has been shunted because a resource (a data set, and RLS cache, or the VSAM RLS server) was not available. For this purpose, it is called by DFHFCQU when CICS is notified that a data set has been unquiesced, has completed a non-BWO copy or has completed forward recovery, and when CICS is notified that a previously failed cache is now available; by DFHFCFL when the API interface is used to retry all shunted work for a given data set; and by DFHFCRO when an override condition is detected, in order to drive any shunted work. DFHFCRR is also called by DFHFCQU when CICS is notified that all systems have completed lost locks recovery for a data set.

DFHFCRS (file control RLS record management processor)

DFHFCRS performs an equivalent function for RLS access mode record management requests as is performed by DFHFCVS for non-RLS access mode requests.

DFHFCRV (file control RLS VSAM interface processor)

DFHFCRV performs an equivalent function for RLS access mode record management requests as is performed by DFHFCVR for non-RLS access mode requests.

DFHFCSD (file control shutdown program)
Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCSD. The entry point address is held in FC static storage in a field named FC_FCSD_ADDRESS, which is set by DFHFCRP when it loads DFHFCSD.

Purpose

The file control shutdown program is part of the file control component. Its purpose is to close all CICS files that are still open during phase 2 of a normal controlled CICS termination. This processing is bypassed for immediate termination.

Called by

DFHSTP, to close all open files managed by CICS file control.

Inputs

The FCSD parameter list, as defined by the DFHFCSDA DSECT, is created as part of the subroutine call.

The input parameters are:

Type of shutdown (immediate, warm)

Outputs

The response and reason codes only, which are returned in the FCSD parameter list.

Operation

DFHFCSD has only one function: TERMINATE.

On a 'warm' shutdown (that is, a not-immediate shutdown), DFHFCSD calls DFHTMP to scan all FCT entries. For each file, it calls DFHFCFS to close the file. A special CLOSE qualifier (shutdown) is specified on the call to DFHFCFS so as not to catalog the FCT entry as in an 'unenabled' state. DFHFCSD also calls DFHFCDO to disconnect coupling facility data table pools.

If RLS is supported, the quiesce system tasks CFQS and CFQR are terminated.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCST (file control statistics program)
Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCST. The entry point address is held in FC static storage in a field named FC_FCST_ADDRESS, which is set by DFHFCRP when it loads DFHFCST.

Purpose

The file control statistics program is part of the file control component.

This program is called to collect statistics for a single file, together with any data table statistics, or to collect statistics for the activity in a shared resources pool.

It is also called to return file statistics associated with a file's use of a shared resources pool.

Called by

DFHSTFC
Collect file statistics

DFHSTLS
Collect pool statistics and also file-in-pool statistics.

Inputs

The FCST parameter list, as defined by the DFHFCSTA DSECT, is created as part of the subroutine call.

The input parameters are:

Request identifier
File name
FCTE token
Statistics record
Pool identifier
Browse token
Reset indicator

Outputs

Returned in the FCST parameter list:

Browse token
Response
Reason

Operation

- Collect file statistics:

  The FCT entry token is validated if supplied; otherwise, the file name is used to locate the FCT entry.

  The file statistics, and any data table statistics, are collected from the FCTE and copied into the statistics record. The statistics in the FCTE are optionally reset according to the reset indicator.

  For data tables, a STATISTICS data table service request is issued to retrieve and reset those statistics that are maintained by data table services. These statistics are appended to the file statistics record.

  The FCT entry is unlocked and the statistics record returned to the caller.

- Collect pool statistics:

  The SHRCTL block for the specified pool is addressed. The pool statistics are copied into the statistics record and are returned to the caller.
- Start browse of files in pool:

  Storage is obtained from the general file control pool for the browse cursor. The browse token is returned to the caller.
- Get statistics for next file in pool:

  DFHTMP is invoked to locate the FCT entry identified by the browse cursor. If the file uses the specified pool, the shared pool statistics for this file are retrieved and returned in the statistics record.

  The statistics contain the data and index buffer sizes, and the number of times buffer waits occurred.

  The browse cursor is updated before returning to the caller.
- End browse of files in pool:

  The browse cursor storage is freed before returning to the caller.

How loaded

By DFHFCRP as part of file control initialization.

DFHFCVR (file control VSAM interface program)

Call mechanism

BALR, obtaining LIFO storage on entry.

Entry address

DFHFCVR. DFHFCVR is link-edited with DFHFCVS. For calls to DFHFCVR from DFHFCVS, the entry point address is known to DFHFCVS from the link-edit. This address is also stored in FC static storage in a field named FC_FCVR_ENTRY. In addition, there is a further “entry address”, UPADEXIT, which is the entry code for the UPAD exit code.

Purpose

The VSAM request interface program is part of the file control component.

This module contains code that issues the VSAM requests, and performs UPAD exit processing in the case of synchronous requests to LSR files, or performs the IOEVENT wait (‘FCIOWAIT’) in the case of asynchronous requests to NSR files.

The module also contains a number of further routines that implement functions required by DFHFCVS.

Called by

DFHFCBD
To issue a message

DFHFCFR
To wait on a CICS ECB

DFHFCVR
Recursively, to issue an ENDREQ request to free a deadlock

DFHFCVS
When issuing VSAM requests

DFHFCVS
To execute one of the constituent functions
VSAM
To invoke the UPAD exit.

Inputs

The FCWSV parameter list, as defined by the DFHFCWS macro, is created in the caller's automatic storage and addressed by register 1 on the call. The input parameters are:

Request identifier
FCTE address
VSWA address
ECB address
Wait resource type
Message number
Dump code

In addition, DFHFCVR requires access to the TCA for certain of its operations.

Outputs

FCVR_RESPONSE parameter (only), defined as part of the FCWSV parameter list.

Operation

Initialize: Copies the VSAM exit list to FC static storage. This action is performed as part of file control initialization.

VSAM_Request: Issues the request to VSAM. Performs the IOEVENT wait. Handles LSR 'no buffers' logical error. Issues change mode request to perform the request under the concurrent TCB if possible.

Get_Strings and Free_Strings: Acquires and frees the required number of shared strings from the LSR pool.

Get_TRANID and Free_TRANID: Allocates and releases a VSAM tranid required during sequential update operations to an LSR file.

Wait_CICSECB: Issues a function request to wait for a CICS ECB to be posted.

Wait_String: Issues a function request to wait for a private string to become available.

Send_Message: Issues a function request to send a message.

How loaded

Link-edited with DFHFCVS to form the DFHFCVS load module, which is loaded by DFHFCRP as part of file control initialization.

DFHFCVS (file control VSAM request processor)
Call mechanism

Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address

DFHFCVS. The entry point address is held in FC static storage in a field named FC_FCVS_ADDRESS, which is set by DFHFCRP when it loads DFHFCVS.

Purpose

Processes file control requests to VSAM files.

Also initializes certain FC static storage fields during file control initialization.

Called by

DFHFCDTS
To access the VSAM source data set to satisfy requests that cannot be satisfied by the table itself

DFHFCFR
After having determined that the request is for a VSAM file.

Inputs

The FCFR parameter list, as defined by the DFHFCFRA DSECT. Also the file control environment, including FC static storage and the FCT.

Outputs

Updated FCFR parameter list.

Operation

Selects on the request type, and passes control to the routine specific to that request.

Acquires and releases the VSWA as necessary.

Logs and journals the request if required.

Performs record-length and key-length checking.

Acquires storage, in the correct key subpool, for requests that specify SET.

Calls DFHFCVR to perform the VSAM request.

Resolves conflicts of exclusive control.

Performs record locking and resolves locking conflicts, including the detection of deadlocks caused either by single tasks that deadlock themselves or by multiple tasks that deadlock each other.

Performs initialization of FC static storage during file control initialization.

For CICS-maintained data tables, calls data table services to update the table to keep it in step with the VSAM source data set.

How loaded

By DFHFCRP as part of file control initialization.

Parameter lists

File control provides the following functions in OCO modules:

FCCR POINT function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The POINT function locates a record in a coupling facility data table.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY
is the 16-byte key of the record to be accessed. For approximate key operations, this specifies the start key and is updated on successful completion to contain the key of the record accessed.

KEY_COMPARISON
is the comparison condition, and can take the values
`LT|LTEQ|EQ|GTEQ|GT`

KEY_MATCH_LENGTH
is the key match length for generic key operations.

UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

KEY
returns the 16-byte key of the located record.

RESPONSE
is DFHFCCR's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND TABLE_LOADING TABLE_TOKEN_INVALID TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR HIGHEST function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The HIGHEST function returns the highest key in a coupling facility data table, if any.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

KEY
returns the 16-byte key of the highest record.

RESPONSE
is DFHFCCR's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND TABLE_LOADING TABLE_TOKEN_INVALID TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR READ function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The READ function reads a record from a coupling facility data table, optionally for update.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY_COMPARISON
is the comparison condition, and can take the values
`LT|LTEQ|EQ|GTEQ|GT`

KEY_MATCH_LENGTH
is the key match length for generic key operations.

KEY
is the 16-byte key of the record to be accessed. For approximate key operations, this specifies the start key and is updated on successful completion to contain the key of the record accessed.

BUFFER
is the input buffer for read requests.

UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

SUSPEND
specifies whether to wait if the requested record is locked by an active lock, and can take the values

`YES|NO`

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

UPDATE_TOKEN
returns a token on a read for update.

KEY
returns the 16-byte key of the highest record.

LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE
is DFHFCCR's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND RECORD_BUSY RECORD_LOCKED TABLE_LOADING INVALID_REQUEST INCOMPLETE_UPDATE TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT UOW_TOO_LARGE POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR READ_DELETE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The READ_DELETE function reads and deletes a record from a coupling facility data table. It is not used by CICS.

FCCR UNLOCK function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The UNLOCK function unlocks a record previously read for update in a coupling facility data table.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY
is the 16-byte key of the record to be unlocked.

BUFFER
is the input buffer for read requests.

UPDATE_TOKEN
is the token returned on the preceding read for update.

UOW_ID
is the unit of work identification, which is required for the locking model (non-recoverable or recoverable).

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

RESPONSE
is DFHFCCR's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND RECORD_CHANGED TABLE_LOADING INVALID_REQUEST UPDATE_TOKEN_INVALID TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR LOAD function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The LOAD function adds a record to a coupling facility data table during loading.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY
is the 16-byte key of the record to be loaded.

DATA
is the address and length of the record data to be loaded.

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

RESPONSE
is DFHFCCR's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED DUPLICATE_RECORD MAXIMUM_RECORDS_REACHED NO_SPACE_IN_POOL INVALID_REQUEST INVALID_LENGTH TABLE_TOKEN_INVALID TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR WRITE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The WRITE function writes a new record to a coupling facility data table.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY
is the 16-byte key of the record to be added.

DATA
is the address and length of the record data to be added.

UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

SUSPEND
specifies whether to wait if the requested record is locked by an active lock, and can take the values

YES|NO

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE
is DFHFCCR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED DUPLICATE_RECORD RECORD_BUSY RECORD_LOCKED MAXIMUM_RECORDS_REACHED NO_SPACE_IN_POOL TABLE_LOADING INVALID_REQUEST INVALID_LENGTH UPDATE_TOKEN_INVALID INCOMPLETE_UPDATE TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT UOW_TOO_LARGE POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR REWRITE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The REWRITE function rewrites an existing record in a coupling facility data table, following a read for update.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY
is the 16-byte key of the record to be rewritten.
DATA
is the address and length of the record data to be rewritten.
UPDATE_TOKEN
is the token returned on the preceding read for update.
UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).
SUSPEND
specifies whether to wait if the requested record is locked by an active lock, and can take the values

`YES|NO`
TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:
LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.
LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.
LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.
RESPONSE
is DFHFCCR's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`
[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND RECORD_CHANGED RECORD_BUSY RECORD_LOCKED MAXIMUM_RECORDS_REACHED NO_SPACE_IN_POOL TABLE_LOADING INVALID_REQUEST INVALID_LENGTH UPDATE_TOKEN_INVALID INCOMPLETE_UPDATE TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT UOW_TOO_LARGE POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR DELETE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The DELETE function deletes a record from a coupling facility data table, following a read for update.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY_COMPARISON
is the comparison condition, and can take the values

`LT|LTEQ|EQ|GTEQ|GT`

KEY_MATCH_LENGTH
is the key match length for generic key operations.

KEY
is the 16-byte key of the record to be deleted.

UPDATE_TOKEN
is the token returned on the preceding read for update.

UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

SUSPEND
specifies whether to wait if the requested record is locked by an active lock, and can take the values

`YES|NO`

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

KEY
is the 16-byte key of the record deleted.

LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE
is DFHFCCR's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND RECORD_CHANGED RECORD_BUSY RECORD_LOCKED TABLE_LOADING INVALID_REQUEST UPDATE_TOKEN_INVALID INCOMPLETE_UPDATE TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT UOW_TOO_LARGE POOL_STATE_ERROR CF_ACCESS_ERROR

FCCR DELETE_MULTIPLE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The DELETE_MULTIPLE function deletes records from a coupling facility data table, subject to key match conditions, until no more records match or an exception occurs.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY_COMPARISON
is the comparison condition, and can take the values

`LT|LTEQ|EQ|GTEQ|GT`

KEY_MATCH_LENGTH
is the key match length for generic key operations.

KEY
is the 16-byte key of the record(s) to be deleted.

UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

SUSPEND
specifies whether to wait if the requested record is locked by an active lock, and can take the values

`YES|NO`

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

DELETED_RECORD_COUNT
is the number of records successfully deleted by the delete_multiple request.

KEY
is the 16-byte key of the last record deleted.

LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE
is DFHFCCR's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECORD_NOT_FOUND RECORD_CHANGED RECORD_BUSY RECORD_LOCKED TABLE_LOADING INVALID_REQUEST UPDATE_TOKEN_INVALID INCOMPLETE_UPDATE TABLE_TOKEN_INVALID TABLE_DESTROYED UOW_FAILED UOW_NOT_IN_FLIGHT UOW_TOO_LARGE POOL_STATE_ERROR CF_ACCESS_ERROR

FCCT OPEN function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The OPEN function defines a coupling facility data table table and establishes a connection between it and a CICS file. A security check is performed for access to the table name. If the table does not exist, it is implicitly created. If the table requires loading, it can only be opened if the access mode specifies exclusive access (or prefer_shared, allowing exclusive access if necessary).

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

RECORD_LENGTH
specifies the maximum record length, in the range 1 to 32767.

KEY_LENGTH
specifies the key length, in the range 1 to 16.

MAXIMUM_RECORDS
specifies the maximum number of records which can be stored in the table.

UPDATE_MODEL
specifies the method to be used for updating. It can take any of the values:
CONTENTION|LOCKING|RECOVERABLE

Contention means version compare and swap. Locking means normal update locking. Recoverable includes backout support in addition to the basic locking model.

INITIAL_LOAD
specifies whether initial load is required. It can take the values:
YES|NO

OPEN_MODE
specifies a read_only or read_write open. It can take the values
READ_ONLY|READ_WRITE

ACCESS_MODE
specifies whether the table is being opened for exclusive or shared use. It can take the values:
EXCLUSIVE|SHARED|PREFER_SHARED

Only one user at a time can have an exclusive open active. If the table requires loading and is not yet being loaded, it can only be opened in exclusive mode. If PREFER_SHARED is specified, the table will be opened in exclusive mode if loading is required, otherwise it will be open in shared mode.

SHARED_ACCESS
specifies for an exclusive mode open whether other users will be allowed shared access to the file at the same time. It can take the values:
NONE|READ_ONLY|READ_WRITE

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

TABLE_TOKEN
is a unique token representing the connection to this table. It must be passed on all subsequent requests against that open table, including close and set.

RECORD_LENGTH
returns the maximum record length of the table.

KEY_LENGTH
returns the key length of the table.

MAXIMUM_RECORDS
returns the maximum number of records limit for the table.

UPDATE_MODEL
returns the update model for the data table. It can take any of the values:
CONTENTION|LOCKING|RECOVERABLE

Contention means version compare and swap. Locking means normal update locking. Recoverable includes backout support in addition to the basic locking model.

INITIAL_LOAD
returns whether or not the data table requires initial loading. It can take the values:
YES|NO

ACCESS_MODE
returns whether the table was opened for exclusive or shared use. It can take the values:
EXCLUSIVE|SHARED

LOADED
returns an indication of whether the table has been loaded. If the table was created as empty this is set to yes as if loading were already done. It can take the values:

`YES|NO`

CURRENT_USERS
returns the number of explicit opens which are currently active against the table (not including internal recoverable opens issued by the server).

CURRENT_RECORDS
returns the number of records in the data table.

CURRENT_HIGH_KEY
returns the key of the last record in the table at the time of the request, or low values if the table does not contain any records.

RESPONSE
is DFHFCCT's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED ACCESS_NOT_ALLOWED TABLE_NOT AVAILABLE NOT_YET_LOADED SHARED_ACCESS_CONFLICT EXCLUSIVE_ACCESS_CONFLICT INCOMPATIBLE_ATTRIBUTES INCOMPLETE_ATTRIBUTES INCORRECT_STATE RECOVERY_NOT_ENABLED OPTION_NOT_SUPPORTED NO_SPACE_IN_POOL MAXIMUM TABLES_REACHED TOO_MANY_USERS TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCT CLOSE function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The CLOSE function terminates the connection to the specified table.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token which was returned by the open.

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

RESPONSE
is DFHFCCT's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED TABLE_TOKEN_INVALID TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCT DELETE function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The DELETE function deletes a coupling facility data table, provided that it is not currently open. A security check for table access is performed.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

RESPONSE
is DFHFCCT's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED ACCESS_NOT_ALLOWED TABLE_NOT_FOUND EXCLUSIVE_ACCESS_CONFLICT TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCT SET function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The SET function is used to change the attributes of a table. The maximum number of records can be changed, the open mode can be changed to indicate no longer loading, and the access mode can be changed from exclusive to shared.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

MAXIMUM_RECORDS
specifies the maximum number of records which can be stored in the table.

AVAILABLE
indicates whether new open requests are to be allowed for this table. It can take the values:

YES|NO

LOADED
indicates whether the table is to be marked as loaded. It can take the values:

YES|NO

ACCESS_MODE
specifies the access mode which is to be set for the table. It can take the values:

EXCLUSIVE|SHARED

The access mode is normally set to shared when a data table load has completed.

SHARED_ACCESS
specifies the shared access which is to be allowed by other users when the access mode is shared.

NONE|READ_ONLY|READ_WRITE

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

RESPONSE
is DFHFCCT's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED ACCESS_NOT_ALLOWED TABLE_NOT_FOUND SHARED_ACCESS_CONFLICT EXCLUSIVE_ACCESS_CONFLICT ALREADY_SET INCORRECT_STATE OPTION_NOT_SUPPORTED TABLE_TOKEN_INVALID TABLE_DESTROYED POOL_STATE_ERROR CF_ACCESS_ERROR

FCCT EXTRACT_STATISTICS function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The EXTRACT_STATISTICS function returns information about a table which is currently open, with optional reset.

Input parameters:

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token which was returned by the open.

RESET_STATISTICS
is an optional parameter which specifies whether or not statistics are to be reset. It can take the values

YES|NO

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters:

CURRENT_USERS
is the number of explicit opens which are currently active against the table (not including internal recoverable opens issued by the server).

CURRENT_RECORDS
is the number of records currently in the data table.

HIGHEST_RECORDS
is the highest number of records in the table as seen by the current server at any time since the last statistics reset.

CONTENTION_COUNT
is the number of times a rewrite or delete failed because of a mismatched version (for the contention model) or the number of times that a lock was found to be unavailable (for the locking or recoverable models) since the last statistics reset.

RESPONSE
is DFHFCCT's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED TABLE_TOKEN_INVALID

FCCU PREPARE function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The PREPARE function prepares to commit a unit of work.

Input parameters:

UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters:

RESPONSE
is DFHFCCU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR

FCCU RETAIN function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The RETAIN function marks a unit of work as retained.

Input parameters:
UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.
TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters:
RESPONSE
is DFHFCCU's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`
[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR

FCCU COMMIT function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The COMMIT function commits a unit of work.

Input parameters:
UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters:
RESPONSE
is DFHFCCU's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`
[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR

FCCU BACKOUT function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The BACKOUT function backs out a unit of work.

Input parameters:
UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.
TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters:
RESPONSE
is DFHFCCU's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`
[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES POOL_STATE_ERROR CF_ACCESS_ERROR

FCCU INQUIRE function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The INQUIRE function inquires about the status of a unit of work.

Input parameters:

UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

UOW_RESTARTED
is an optional parameter which indicates whether the inquire should select only units of work which have been through restart processing, and can take the values:

```
NO|YES
```

TRANSACTION_NUMBER
is used for debug trace purposes.

BROWSE
specifies whether the inquire is for a single unit of work or for the first or next UOW in a browse. If omitted, a single UOW inquire is performed. If specified, it can take the values

```
FIRST|NEXT
```

FIRST indicates a search for a UOWID greater than or equal to the specified UOWID, and NEXT indicates a search for a UOWID greater than the specified UOWID.

Output parameters:

UOW_STATE
indicates the state of an active unit of work, and can have any of the values:

```
IN_FLIGHT|IN_DOUBT|IN_COMMIT|IN_BACKOUT
```

In_flight means that the unit of work has made some changes but has not yet reached the stage of prepare to commit. In_doubt means that it has been prepared but not committed or backed out. In_commit means that commit processing has been started. In_backout means that backout processing has been started. (When commit or backout processing completes, the unit of work is deleted).

UOW_ID
is the CICS unit of work id of the UOW for which inquire data is being returned.

UOW_RESTARTED
indicates whether the unit of work has been through restart processing, and can take the values:

```
NO|YES
```

UOW_RETAINED
indicates whether the locks for the unit of work have been marked as retained, either explicitly within the current connection or implicitly by a restart. It can take the values:

```
NO|YES
```

RESPONSE
is DFHFCCU's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND

FCCU RESTART function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The RESTART function establishes recovery status on connecting to a CFDT server.

Input parameters:

UOW_SUBSYSTEM_NAME
is not specified by CICS (the CICS applid is used by default).

TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters:

RESPONSE
is DFHFCCU's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED SUBSYSTEM_ALREADY_ACTIVE RESTART_ALREADY_ACTIVE TABLE_OPEN_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR

FCDS EXTRACT_CFDT_STATS function

This function causes statistics relating to coupling facility data table usage to be extracted from the coupling facility data tables server.

Input parameters:

FCTE_POINTER
is the address of the FCTE entry of the file for which CFDT statistics are to be extracted.

RESET_STATISTICS
indicates whether the statistics fields are to be reset to zero or not. It takes the values

```
YES|NO
```

TRANSACTION_NUMBER
is an optional parameter which allows the transaction number to be passed to the CFDT server for inclusion in trace messages.

Output parameters:

CURRENT_USERS
is an optional fullword parameter which returns the current number of users of the coupling facility data table (that is, the number of opens issued against it).

MAXIMUM_RECORDS
is an optional fullword parameter which returns the current value of the MAXNUMRECS limit for the data table.

CURRENT_RECORDS
is an optional fullword parameter which returns the current number of records in the coupling facility data table.

HIGHEST_RECORDS
is an optional fullword parameter which returns the highest number of records which have ever been in this coupling facility data table since it was last created.

CONTENTION_COUNT
is an optional fullword parameter which returns the number of contentions which have been detected, for a coupling facility data table which uses the contention update model.

RESPONSE
is DFHFCDS's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR CFDT_REOPEN_ERROR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_STATS_ERROR CFDT_SYSIDERR CFDT_TABLE_GONE
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	POOL_ELEMENT_NOT_FOUND ABEND DISASTER_PERCOLATION

FCDS DISCONNECT_CFDT_POOLS function

This function causes CICS to disconnect from any coupling facility data table pools to which it is connected.

Input parameters: None

Output parameters:

RESPONSE
is DFHFCDS's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	CFDT_DISCONNECT_ERROR
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU PREPARE function

This function causes the coupling facility data table server to be called to prepare a unit of work which has made recoverable updates to one or more coupling facility data tables.

Input parameters:

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the prepare is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The prepare call will be issued to the CFDT server for this pool.

POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

UOW_ID
is the identifier for the unit of work which is to be prepared.

Output parameters:

RESPONSE
is DFHFCDU's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR RESYNC_RETRY_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU RETAIN function

This function causes the coupling facility data table server to be called to convert locks held by the unit of work against recoverable coupling facility data tables into retained locks.

Input parameters:

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the retain is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The retain call will be issued to the CFDT server for this pool.

POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.
UOW_ID
is the identifier for the unit of work for which locks are to be retained.

Output parameters:
RESPONSE
is DFHFCDU's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`
[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR RESYNC_RETRY_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU COMMIT function

This function causes the coupling facility data table server to be called to commit a unit of work which has made recoverable updates to one or more coupling facility data tables.

Input parameters:
POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the commit is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The commit call will be issued to the CFDT server for this pool.
POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.
UOW_ID
is the identifier for the unit of work which is to be committed.

Output parameters:
RESPONSE
is DFHFCDU's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES UOW_FAILED NO_SPACE_IN_POOL POOL_STATE_ERROR CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR RESYNC_RETRY_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU BACKOUT function

This function causes the coupling facility data table server to be called to backout a unit of work which has made recoverable updates to one or more coupling facility data tables.

Input parameters:

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the backout is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The backout call will be issued to the CFDT server for this pool.

POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

UOW_ID
is the identifier for the unit of work which is to be backed out.

Output parameters:

RESPONSE
is DFHFCDU's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND UOW_MADE_NO_CHANGES POOL_STATE_ERROR CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR RESYNC_RETRY_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU INQUIRE function

This function causes an INQUIRE to be issued to the coupling facility data table in order to obtain information about the status of an active unit of work. If the BROWSE parameter is specified, then the function will return the status of the next unit of work in the browse.

Input parameters:

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the INQUIRE is to be issued. The inquire call will be issued to the CFDT server for this pool.

POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

UOW_ID
identifies the unit of work for which status information is to be returned, or gives the previous unit of work in the browse.

UOW_RESTARTED
is an optional input parameter which indicates whether or not the inquire should select only units of work which have been through restart processing. It can take the values

```
YES|NO
```

BROWSE
is an optional parameter which specified whether the inquire is for a single unit of work or for the first or next UOW in a browse, and which can take the values

```
FIRST|NEXT
```

If the BROWSE parameter is omitted, the request is a single UOW inquire. The FIRST option indicates a search for a UOW id greater than or equal to the specified UOW_ID, and next indicates a search for a UOW id greater than the specified UOW_ID.

Output parameters:

RETURNED_UOW_ID
Is the unit of work for which the browse is returning status information.

UOW_STATE
indicates the state of the unit of work, and can have the values:
`IN_FLIGHT|IN_DOUBT|IN_COMMIT|IN_BACKOUT`
UOW_RESTART_STATE
indicates whether the unit of work has been through restart processing.
UOW_RETAINED
indicates whether the locks for the unit of work have been retained.
RESPONSE
is DFHFCDU's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`
[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED RECOVERY_NOT_ENABLED UOW_NOT_FOUND CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR RESYNC_RETRY_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDU RESTART function

This function establishes recovery status for a coupling facility data table pool when a CICS region has successfully connected to it.

Input parameters:
POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for recovery status is to be established. The RESTART call will be issued to the CFDT server for this pool.
POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

Output parameters:
RETURNED_UOW_ID
Is the unit of work for which the browse is returning status information.
UOW_STATE
indicates the state of the unit of work, and can have the values:
`IN_FLIGHT|IN_DOUBT|IN_COMMIT|IN_BACKOUT`
UOW_RESTART_STATE
indicates whether the unit of work has been through restart processing.
UOW_RETAINED
indicates whether the locks for the unit of work have been retained.
RESPONSE
is DFHFCDU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	SERVER_CONNECTION_FAILED SUBSYSTEM_ALREADY_ACTIVE RESTART_ALREADY_ACTIVE TABLE_OPEN_FAILED NO_SPACE_IN_POOL CF_ACCESS_ERROR CFDT_SYSIDERR CFDT_SERVER_NOT_AVAILABLE CFDT_SERVER_NOT_FOUND CFDT_CONNECT_ERROR CFDT_DISCONNECT_ERROR
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDY RESYNC_CFDT_POOL function

This function causes a coupling facility data table pool to be resynchronized.

Input parameters:

POOL_NAME
is the name of the coupling facility data table pool which is to be resynchronized.

Output parameters:

RESPONSE
is DFHFCDY's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INITIATE_RECOVERY_FAILED TERMINATE_RECOVERY_FAILED CFDT_SERVER_CALL_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDY RESYNC_CFDT_LINK function

This function causes a link between a unit of work and a coupling facility data table pool to be resynchronized.

Input parameters:

POOL_NAME
is the name of the coupling facility data table pool for which the link is to be resynchronized.

UOW_ID
is the unit of work ID which identifies the link to be resynchronized.

Output parameters:

RESPONSE
is DFHFCDY's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	INITIATE_RECOVERY_FAILED TERMINATE_RECOVERY_FAILED CFDT_SERVER_CALL_FAILED
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCDY RETURN_CFDT_ENTRY_POINTS function

This function causes module DFHFCDY to return the entry point addresses of the other modules with which it is link-edited.

Input parameters: None

Output parameters:

CFDT_EP_DFHFCDW
is the entry point address of module DFHFCDW.

CFDT_EP_DFHFCDU
is the entry point address of module DFHFCDU.

RESPONSE
is DFHFCDY's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FORMAT INVALID_FUNCTION
DISASTER	ABEND DISASTER_PERCOLATION

FCFL END_UOWDSN_BROWSE function

After a browse of all the data set failures within a unit of work, the END_UOWDSN_BROWSE function releases the storage that was used for a snapshot of the failures.

Input parameters:

BROWSE_TOKEN
is the token which was used for the browse.

Output parameters:

RESPONSE
is DFHFCFL's response to the call. It can have any of these values:

`OK|INVALID|DISASTER|PURGED`

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_BROWSE_TOKEN
DISASTER	DISASTER_PERCOLATION ABEND

FCFL FIND_RETAINED function

This function looks for any FLAB associated with the specified data set which is flagged as retained, indicating that there are retained locks associated with the data set.

Input parameters:

DSNAME
is the 44-character name of the data set for which associated retained locks are to be found.

Output parameters:

RETLOCKS
indicates whether or not there are retained locks associated with the data set, and can have either of these values:

`RETAINED|NORETAINED`

RESPONSE
is DFHFCFL's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND

FCFL FORCE_INDOUBTS function

This function is used by the CEMT or EXEC CICS SET DSNAME() UOWACTION(COMMIT | BACKOUT | FORCE) command. Shunted indoubt units of work are forced to complete in the specified direction. FORCE means that the direction is obtained from the ACTION specified on the transaction definition.

Input parameters:

DSNAME
is the 44-character name of the data set for which shunted indoubt units of work are to be forced to complete.

DIRECTION
is the direction in which the units of work are to complete: forwards (commit),

backwards (backout), or heuristic (from the action specified on the transaction definition). It can have any of these values:

```
FORWARD|BACKWARD|HEURISTIC
```

Output parameters:

RESPONSE

is DFHFCFL's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]

is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND

FCFL GET_NEXT_UOWDSN function

This function returns the failure information for the next data set that has a failure within the unit of work being browsed.

Input parameters:

BROWSE_TOKEN

is the token for the browse, which was returned by a START_UOWDSN_BROWSE call.

Output parameters:

DSNAME

is the 44-character name of the data set for which failure information is returned.

[RLSACCESS]

indicates whether the data set was last open in RLS or non-RLS access mode, and can have either of these values:

```
RLS|NOTRLS
```

[CAUSE]

indicates the cause of the failure, and can have any of these values:

```
CACHE|RLSSERVER|CONNECTION|DATASET|UNDEFINED
```

[RETAIN_REASON]

indicates the reason for the failure, and can have any of these values:

```
RLSGONE|COMMITFAIL|IOERROR|DATASETFULL|INDEXRECFULL|
OPENERROR|DELEXITERROR|DEADLOCK|BACKUPNONBWO|
LOCKSTRUCFULL|FAILEDBKOUT|NOTAPPLIC|RR_COMMITFAIL|
RR_INDOUBT
```

RESPONSE

is DFHFCFL's response to the call. It can have any of these values:

```
OK|INVALID|EXCEPTION|DISASTER
```

[REASON]

is returned when RESPONSE is EXCEPTION, INVALID, or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	END_OF_LIST
INVALID	INVALID_BROWSE_TOKEN
DISASTER	DISASTER_PERCOLATION ABEND

FCFL RESET_BFAILS function

This function is used by the CEMT and EXEC CICS SET DSNAME() ACTION(RESETLOCKS) command. It purges shunted unit of work log records which hold backout-failure or commit-failure locks on the specified data set, and releases the locks.

Input parameters:

DSNAME

is the 44-character name of the data set for which backout and commit failures are to be reset.

Output parameters:

RESPONSE

is DFHFCFL's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]

is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND REMOVE_FAILURE

FCFL RETRY function

This function is used by the CEMT and EXEC CICS SET DSNAME() UOWACTION(RETRY) command. It drives retry of any failed backouts and commits for the specified data set, by informing DFHFCRR that the failed resource (that is, the data set) is now available.

Input parameters:

DSNAME

is the 44-character name of the data set for which backouts or commits are to be retried.

Output parameters:

RESPONSE

is DFHFCFL's response to the call. It can have any of these values:

`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]

is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND RESOURCE_NOT_FOUND

FCFL START_UOWDSN_BROWSE function

This function starts a browse of the data set failures within a unit of work. A snapshot of the failed data sets for the unit of work and the reasons for the failures are collected in an in-storage table to be browsed by the GET_NEXT_UOWDSN function.

Input parameters:

UOW
is the 8-byte local unit of work identifier.

Output parameters:

BROWSE_TOKEN
is a token which is used during the browse.

RESPONSE
is DFHFCFL's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|PURGED
```

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
EXCEPTION	UOW_NOT_FOUND NO_FLABS_FOUND
DISASTER	DISASTER_PERCOLATION ABEND

FCFL TEST_USER function

This function is used to test if the task has updated a record, and therefore established itself as a file user, either for any data set or for a specified data set. It can be used either as a domain subroutine call or as an inline macro.

Input parameters:

[ENVIRONMENT]
is an optional parameter which is a fullword environment identifier. If specified, then the function will test whether the task is a user of any files within that environment.

[DSNAME]
is an optional parameter which specifies that a particular data set is to be tested.

Output parameters:

FLAB_PTR
is the address of a FLAB which was found by the test. If a non-zero value is returned, then this means that the user is a task.

RESPONSE
is DFHFCFL's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	DISASTER_PERCOLATION ABEND

FCLJ FILE_OPEN function

This function is called when a file is opened, and causes a 'tie up record' record to be written to the log of logs if either the file (or associated data set) is forward recoverable or if autojournalling is specified for the file, to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if autojournalling is specified for the file.

Input parameters:

FCTE_ADDRESS
is the address of the file control table entry for the file being opened.

Output parameters:

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR

FCLJ FILE_CLOSE Function

This function is called when a file is closed, and causes a file close log record to be written to the log of logs if either the file (or associated data set) is forward recoverable or if autojournalling is specified for the file, to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if autojournalling is specified for the file.

Input parameters:

FCTE_ADDRESS
is the address of the file control table entry for the file being closed.

Output parameters:

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR

FCLJ READ_ONLY Function

This function causes a read_only log record to be written to an autojournal, if read-only autojournalling is specified on the file definition. The log record is built using the input parameters.

Input parameters:

BASE_ESDS_RBA
is the RBA of the record being read, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file being read.

KEY_ADDRESS
is the address of the key of the record being read.

KEY_LENGTH
is the key length of the record being read.

RECORD_ADDRESS
is the address of the record being read.

RECORD_LENGTH
is the length of the record being read.
SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

```
YES|NO
```

Output parameters:
RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

```
OK|INVALID|PURGED|DISASTER
```

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ READ_UPDATE Function

This function causes a read_update log record to be written to the system log, if the file is recoverable, and if the destination parameter specifies either LOG or BOTH. It causes a read_update log record to be written to the autojournal if journaling of read updates is specified on the file definition, and if the destination parameter specifies either JOURNAL or BOTH. The log record is built using the input parameters.

Input parameters:
BASE_ESDS_RBA
is the RBA of the record being read for update, if the file is an ESDS.
FCTE_ADDRESS
is the address of the file control table entry for the file being read for update.
KEY_ADDRESS
is the address of the key of the record being read for update.
KEY_LENGTH
is the key length of the record being read for update.
RECORD_ADDRESS
is the address of the record being read for update.
RECORD_LENGTH
is the length of the record being read for update.
DESTINATION
specifies whether the log record is to be written to the autojournal, the system log, or both. It is used to suppress writing records that would otherwise be requested by the file definition. It can have any of these values:

```
JOURNAL|LOG|BOTH
```

SYNCHRONIZE_LOG
indicates whether or not the system log is to be synchronized (forced) when the log record is written. It can have either of these values:

```
YES|NO
```

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

```
YES|NO
```

Output parameters:

[LOG_TOKEN]
is an optional parameter which is returned if SYNCHRONIZE(NO) was specified, and which contains a token to be used when subsequently synchronizing (forcing) the system log.

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

```
OK|INVALID|PURGED|DISASTER
```

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ WRITE_UPDATE Function

This function causes a write_update log record to be written to the forward recovery log, if the file (or associated data set) is forward recoverable, and to the autojournal, if journaling of write updates is specified on the file definition. A write_update log record represents the completion of a file REWRITE request. The log record is built using the input parameters.

Input parameters:

BACKOUT
indicates if the call is made as part of transaction backout processing. It can have either of these values:

```
YES|NO
```

BASE_ESDS_RBA
is the RBA of the record being rewritten, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file being rewritten to.

KEY_ADDRESS
is the address of the key of the record being rewritten.

KEY_LENGTH
is the key length of the record being rewritten to.

RECORD_ADDRESS
is the address of the record being rewritten.

RECORD_LENGTH
is the length of the record being rewritten.

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

```
YES|NO
```

Output parameters:

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

```
OK|INVALID|PURGED|DISASTER
```

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ WRITE_ADD Function

This function causes a write_add log record to be written to the system log if the file is recoverable, and if the destination parameter specifies BOTH. It causes a write_add log record to be written to the autojournal if journaling of write adds was specified on the file definition. The log record is built using the input parameters.

Input parameters:

BACKOUT
indicates if the call is made as part of transaction backout processing. It can have either of these values:

`YES|NO`

BASE_ESDS_RBA
is the RBA of the record being added, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file being written to.

KEY_ADDRESS
is the address of the key of the record being added.

KEY_LENGTH
is the key length of the record being written to.

MASSINSERT
indicates whether or not the record is being added as part of a mass insert. It can have either of these values:

`YES|NO`

DESTINATION
specifies whether the log record is to be written to the autojournal only, or to both the autojournal and the system log. It is used to suppress writing records that would otherwise be requested by the file definition. It can have either of these values:

`JOURNAL|BOTH`

RECORD_ADDRESS
is the address of the record being added.

RECORD_LENGTH
is the length of the record being added.

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

`YES|NO`

Output parameters:

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

`OK|INVALID|PURGED|DISASTER`

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ WRITE_ADD_COMPLETE Function

This function causes a write_add_complete log record to be written to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if write_add_complete journaling is specified on the file definition. It causes a truncated write_add_complete log record to be written to the system log if the file is a recoverable ESDS accessed in non-RLS mode. If MASSINSERT(YES) and MASSINSERT_STAGE(LAST) are specified, then only the system log record is written, and not the forward recovery log or autojournal record. The log record is built using the input parameters.

Input parameters:

BACKOUT
indicates if the call is made as part of transaction backout processing. It can have either of these values:

YES|NO

BASE_ESDS_RBA
is the RBA of the record that has been added, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file that has been written to.

KEY_ADDRESS
is the address of the key of the record which has been added.

KEY_LENGTH
is the key length for the file which has been written to.

MASSINSERT
indicates whether or not the record was added as part of a mass insert. It can have either of these values:

YES|NO

[MASSINSERT_STAGE]
is an optional parameter which indicates whether the record is either the first or last record added during a massinsert sequence. It can have either of these values:

FIRST|LAST

RECORD_ADDRESS
is the address of the record which has been added.

RECORD_LENGTH
is the length of the record which has been added.

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

YES|NO

Output parameters:

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ WRITE_DELETE Function

This function causes a write_delete log record to be written to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if journaling of write_deletes is specified on the file definition. The log record is built using the input parameters.

Input parameters:

BACKOUT
indicates if the call is made as part of transaction backout processing. It can have either of these values:

`YES|NO`

BASE_ESDS_RBA
is the RBA of the record being deleted, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file.

KEY_ADDRESS
is the address of the key of the record being deleted.

KEY_LENGTH
is the key length for the file.

BASE_KEY_ADDRESS
is the address of the base key of the record being deleted, which is used if the data set is being accessed via a path.

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

`YES|NO`

Output parameters:

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

`OK|INVALID|PURGED|DISASTER`

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR RM_RETURNED_ERROR

FCLJ SYNCHRONIZE_READ_UPDATE Function

This function causes any log records previously written to the system log for this file to be synchronized (forced). The log token returned on a previous call to write a log record for this file is supplied as input.

Input parameters:

FCTE_ADDRESS
is the address of the file control table entry for the file being read for update.

LOG_TOKEN
is the token returned on a previous call. The system log record written by the previous call, plus any log records written before that, are hardened.

Output parameters:
RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:
`OK|INVALID|PURGED|DISASTER`
[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND RM_RETURNED_ERROR

FCLJ TAKE_KEYPOINT Function

If BWO copy is supported by this CICS (indicated by a flag in file control static storage), then this function performs a scan of the file control table and, unless it has been called within the last half hour, writes a tie up record for each file open for update in non-RLS mode that is BWO-eligible and forward recoverable to the forward recovery log.

A tie up record specifies which CICS system within the sysplex opened the file, and the data set which the file was opened against. Tie up records are used by forward recovery utilities, for example CICS VSAM Recovery for z/OS.

Input parameters: None

Output parameters:
KEYPOINT_TAKEN
indicates whether or not the set of tie up records was successfully written. It can have either of these values:
`YES|NO`
RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:
`OK|INVALID|PURGED|DISASTER`
[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR TM_GETNEXT_FCTE_FAILED

FCLJ DATASET_COPY Function

This function is called when DFSMSdss initiates a copy of an RLS data set via the VSAM RLS quiesce mechanism. The function causes a 'tie up record' to be written to the log of logs if either the data set is forward recoverable, or some flavor of autojournalling has been specified in the file definition. In addition, if applicable, a record is written to the forward recovery log.

A tie up record specifies which CICS system within the sysplex opened the file, and the data set which the file was opened against. Tie up records are used by forward recovery utilities, for example CICS VSAM Recovery for z/OS.

Input parameters:

FCTE_ADDRESS
is the address of the file control table entry for the file associated with a data set being copied.

Output parameters:

RESPONSE
is DFHFCLJ's response to the call. It can have any of these values:

```
OK|INVALID|PURGED|DISASTER
```

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND LG_RETURNED_ERROR

FCQR RECEIVE_QUIESCES function

This function consists of a forever loop around a dispatcher wait on an ECB. It receives work from the CICS RLS quiesce exit DFHFCQX whenever SMSVSAM requires CICS to perform processing for a quiesce request. DFHFCQX queues the request to DFHFCQR by adding an FC Quiesce Receive Element (FCQRE) to a chain anchored in file control static storage, and posting the ECB associated with the chain, also in FC static.

The posting of the ECB wakes the CFQR transaction, which executes the code in DFHFCQR. The FCQREs on the chain are processed, and DFHFCQU is called with function PROCESS_QUIESCE to perform the actual work. The ECB might also be posted to inform DFHFCQR that CICS is terminating. When DFHFCQU has finished processing, DFHFCQR unchains and frees the FCQRE.

Input parameters: None.

Output parameters:

RESPONSE
is DFHFCQR's response to the call. It can have any of these values:

```
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
```

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND PROCESS_QUIESCE_ERROR DISASTER_PERCOLATION

FCQS SEND_QUIESCES Function

This function consists of a forever loop around a dispatcher wait on a list of ECBs. Work is received from tasks that want to send a quiesce request to SMSVSAM. Such tasks call DFHFCQI with function INITIATE_QUIESCE, which queues the request to DFHFCQS by adding an FC Quiesce Send Element (FCQSE) to the chain anchored in file control static storage, and posting an ECB associated with the chain, also in FC static.

When the ECB is posted, it wakes the CFQS transaction, which executes the code in DFHFCQS. The FCQSEs on the chain are processed, and DFHFCCA is called

with function QUIESCE_REQUEST to issue the appropriate flavor of IDAQUIES macro to SMSVSAM. This is an asynchronous operation, and SMSVSAM returns the address of an ECB that will be posted when the IDAQUIES completes. This is saved in the FCQSE.

DFHFCQS then goes back into its dispatcher wait. It is waiting on a list of ECBs, the ECB for the chain plus an ECB for **each** IDAQUIES request. It wakes and processes the chain whenever one of these ECBs is posted. The wait also specifies a timeout interval, so that IDAQUIES requests that hang can be detected. When DFHFCQS wakes up, this can mean that: there is new work on the chain, or a quiesce request has completed, or a quiesce request timed out, or CICS is terminating. When a quiesce request has completed or timed out, DFHFCQS will resume the initiating task if it is waiting, after issuing appropriate messages and invoking global user exit XFCQUIS if active.

Input parameters: None.

Output parameters:

RESPONSE
is DFHFCQS's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	ABEND TIMEOUT_CANCEL_ERROR DISASTER_PERCOLATION

FCQU PROCESS_QUIESCE function

DFHFCQU PROCESS_QUIESCE is called whenever a quiesce request is received from VSAM RLS. The quiesce exit DFHFCQX queues requests to the CFQR system transaction (DFHFCQR), which calls DFHFCQU to process each one in turn. The PROCESS_QUIESCE function is also called to implement a non-RLS variant of QUIESCE called NON_RLS_CLOSE. This is for non-RLS files, is only used internally by CICS, and does not run under the CFQR system transaction. Each quiesce request type is processed in a different way by DFHFCQU.

QUIESCE
corresponds to an SMSVSAM QUICLOSE. All files open against the data set are closed, the file state of each file is set to unenabled but with a flag that says re-enable on QUIOPEN, and a QUICMP is issued for the QUICLOSE back to VSAM RLS to indicate our QUICLOSE processing is complete. The immediate option on the DFHFCQU call governs how file closes are to be performed. If NO or omitted then closes will occur when all UOWs using the data set have completed normally. If YES then all such UOWs will be force purged to speed things up.

UNQUIESCE
corresponds to an SMSVSAM QUIOPEN. All files associated with the data set are checked to see if the file state requires resetting back to enabled, because it had been set unenabled by a QUICLOSE.

NONBWO_START
corresponds to an SMSVSAM QUICOPY. CICS prepares for a non-BWO backup of the data set by preventing new units of work from updating the data set, allowing existing UOWs to finish updating the data set, and then

issuing a QUICMP for the QUICOPY back to SMSVSAM to indicate that QUICOPY processing is complete. The files involved are not closed.

NONBWO_END
corresponds to an SMSVSAM QUICEND. All files associated with the data set are checked to see if the file state requires resetting to enabled because it had been set unenabled by an OPEN failure, and a set of 'tie up records' are written for the data set.

BWO
corresponds to an SMSVSAM QUIBWO. CICS prepares for a BWO backup of the data set by writing a set of 'tie up records' allowing existing units of work to finish updating the data set, and then issuing a QUICMP for the QUIBWO back to SMSVSAM to indicate that QUIBWO processing is complete. The files involved are not closed, nor are updates prevented.

BWO_END
corresponds to an SMSVSAM QUIBEND. The only processing involved is to stop an existing BWO quiesce if one is in progress.

LOST_LOCKS_RECOVERED
corresponds to an SMSVSAM QUILLRC. It notifies CICS that lost locks recovery has been completed for the data set throughout the sysplex. DFHFCRR is called with function LOST_LOCKS_RECOVERED to process the availability of the data set.

FORWARD_RECOVERY_COMPLETE
corresponds to an SMSVSAM QUIFRC. It notifies CICS that forward recovery has been completed for the data set. DFHFCRR is called with function RESOURCE_AVAILABLE to process the availability of the data set.

CACHE_AVAILABLE
corresponds to an SMSVSAM QUICA. It notifies CICS that a previously failed cache structure is now available. DFHFCRR is called with function RESOURCE_AVAILABLE to process the availability of the cache.

NON_RLS_CLOSE
processes a non-RLS variant of type CLOSE called NON_RLS_CLOSE. All ACBs open against the specified non-RLS data set are closed.

Some of the requests cause global user exit XFCVSDS to be invoked if active and a DSNB exists for the data set, and XFCVSDS can suppress certain of the requests. Suppression causes the quiesce request to be cancelled throughout the sysplex (by issuing the inverse quiesce request).

The types of quiesce that DFHFCQU can receive fall into two 'completion' categories.

1. Those for which VSAM does not require completion notification. For these no IDAQUIES QUICMP is issued. The successful return of the quiesce exit DFHFCQX to VSAM is sufficient. The requests in this category are:

   ```
   UNQUIESCE, NONBWO_END, BWO_END, CACHE_AVAILABLE,
   LOCKS_RECOVERY_COMPLETE, FORWARD_RECOVERY_COMPLETE.
   ```

2. Those for which VSAM requires completion notification because CICS must complete some critical processing. For these an IDAQUIES QUICMP must be issued when CICS processing is complete. The requests in this category are:

   ```
   QUIESCE, NONBWO_START, BWO_START.
   ```

Input parameters:

QUIESCE_TYPE
indicates the type of quiesce being requested. It can have any of these values:

QUIESCE|UNQUIESCE|NONBWO_START|NONBWO_END|BWO_START|
BWO_END|LOCKS_RECOVERY_COMPLETE|
FORWARD_RECOVERY_COMPLETE|CACHE_AVAILABLE|
NON_RLS_CLOSE

DSNAME|CACHE_NAME
either specifies the 44-character name of the data set to which the quiesce request applies, or (when the quiesce_type is CACHE_AVAILABLE) the 16-character name of the cache structure which has become available.

[IMMEDIATE]
applies when the quiesce_type is QUIESCE or NON_RLS_CLOSE, and indicates whether units of work which have updated the data set will be forced to complete immediately, or whether the request will wait for such units of work to complete naturally. It can have either of these values:

YES|NO

[CONCURRENT]
applies when the quiesce_type is NONBWO_START or BWO_START, and indicates whether the concurrent copy technique is being used. It is purely informational, and has no effect on the processing. It can have either of these values:

YES|NO

[QUIESCE_TOKEN]
is a token which is supplied by SMSVSAM when certain quiesce requests are initiated, and must be passed back when the quiesce complete is issued.

Output parameters:

RESPONSE
is DFHFCQU's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_QUIESCE_TYPE
EXCEPTION	DSNB_NOT_FOUND
DISASTER	ABEND DISASTER_PERCOLATION DFHFCRR_ERROR DFHFCQI_ERROR DFHFCFS_ERROR DFHTM_FAILURE

FCRR RESTART_RLS Function

This function performs a restart of the RLS component of file control. The exact processing depends on the type of restart being performed.

COLD and INITIAL: The RLS control ACB is registered, and RLS is cold started, both via calls to DFHFCCA.

WARM and EMERGENCY: The RLS control ACB is registered, and recovery information is inquired upon from SMSVSAM, both via calls to DFHFCCA. If the recovery information indicates that there are data sets in lost locks status, then the corresponding DSNBs are marked as being lost locks, and preparation for lost locks recovery is carried out. Any orphan locks are eliminated.

DYNAMIC: This type of restart occurs when a new instance of the SMSVSAM server becomes available following a previous server failure.

Having waited for file control restart to complete if it was still in progress, and for any in-progress dynamic RLS restart to complete, RLS access is drained if this has not already been done, the control ACB is registered, and recovery information is inquired upon from SMSVSAM, all three via calls to DFHFCCA. If the recovery information indicates that there are data sets in lost locks status, then the corresponding DSNBs are marked as being lost locks, and preparation for lost locks recovery is carried out. Any orphan locks are eliminated. The CICS recovery manager is called to unshunt any units of work that are backout-failed due to the SMSVSAM server failure or a general file backout failure, and any units of work that are commit-failed due to the SMSVSAM server failure.

Input parameters:

TYPE_OF_RESTART
indicates the type of RLS restart being performed, and can have any of these values:
`COLD|WARM|EMERGENCY|DYNAMIC`

Output parameters:

RESPONSE
is DFHFCRR's response to the call. It can have any of these values:
`OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]
is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION INVALID_RESTART_TYPE
EXCEPTION	REGISTER_CTL_ACB_FAILED COLD_START_RLS_FAILED DRAIN_RLS_FAILED LOST_LOCKS_INFO_LOST INQUIRE_RECOVERY_FAILED LOST_LOCKS_COMPLETE_FAILED ORPHAN_RELEASE_FAILED
DISASTER	DSSR_FAILED TM_LOCATE_FAILED TM_UNLOCK_FAILED ABEND DISASTER_PERCOLATION

FCRR RESOURCE_AVAILABLE function

This function causes the CICS recovery manager to be notified of the availability of the specified resource. When the resource_type is DSET, an RMRE AVAIL call is issued for the specified data set. When the resource_type is CACHE, an RMRE avail call is issued for every data set that has outstanding work shunted due either to a cache failure or to a general file backout failure. When the resource_type is OTHER, an RMRE AVAIL call is issued for the specified resource.

Input parameters:

RESOURCE_TYPE
is the type of resource which has become available, and can have any of these values:

DSET|CACHE|OTHER

RESOURCE_NAME
is the 44-character field containing the name of the resource which has become available.

RESOURCE_NAME_LENGTH
is a halfword containing the actual length of the resource name.

Output parameters:

RESPONSE
is DFHFCRR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION INVALID_RESOURCE_TYPE
DISASTER	ABEND DISASTER_PERCOLATION

FCRR LOST_LOCKS_RECOVERED function

This function is called when lost locks recovery for a data set has been completed by all CICS regions that were sharing it, and causes the flag in the DSNB which indicates that the data set is in lost locks state to be cleared.

Input parameters:

RESOURCE_NAME
is the 44-character field containing the name of the resource (data set) for which lost locks recovery has been completed.

Output parameters:

RESPONSE
is DFHFCRR's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:

RESPONSE	Possible REASON values
INVALID	INVALID_FUNCTION
EXCEPTION	SPHERE_UNKNOWN
DISASTER	TM_LOCATE_FAILED TM_UNLOCK_FAILED ABEND DISASTER_PERCOLATION

File Control's call back gates

Table 72 summarizes file control's call back gates. It shows the FC level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the format for calls to the gate.

Table 72. File control's call back gates

Gate	Trace	Function	Format
RMRO			RMRO
	FC 0BE0	PERFORM_PREPARE	
	FC 0BE1	PERFORM_COMMIT	
		START_BACKOUT	
		DELIVER_BACKOUT_DATA	
		END_BACKOUT	
		PERFORM_SHUNT	
		PERFORM_UNSHUNT	
RMKP			RMKP
	FC 0BE0	TAKE_KEYPOINT	
	FC 0BE1		
RMLK			RMLK
	FC 24A0	PREPARE	
	FC 24A1	COMMIT	
		SEND_DO_COMMIT	
		SHUNT	
		UNSHUNT	
RMDE			RMDE
	FC 0BE0	START_DELIVERY	
	FC 0BE1	DELIVER_RECOVERY	
		DELIVER_FORGET	
		END_DELIVERY	
LGGL			LGGL
	FC 2350	ERROR	
	FC 2351		
DMEN			DMEN
	FC 0BD0	NOTIFY_SMSVSAM_AVAILABLE	
	FC 0BD1		

You can find descriptions of these functions and their input and output parameters, in the following topics:

- "Recovery manager domain's callback formats" on page 996
- "Log manager domain's call-back formats" on page 680
- "Domain manager domain's callback formats" on page 418

The functions of the RMRO gate are processed by DFHFCRC. For PERFORM_PREPARE and PERFORM_COMMIT, DFHFCRC performs prepare and commit processing respectively for any file resources involved in the unit of work. For START_BACKOUT, DELIVER_BACKOUT_DATA and END_BACKOUT, DFHFCRC backs out changes made to file resources by the unit of work. For PERFORM_SHUNT and PERFORM_UNSHUNT, DFHFCRC respectively shunts and unshunts the file control structures representing recoverable parts of the unit of work.

The functions of the RMKP gate are processed by DFHFCRC. For TAKE_KEYPOINT, DFHFCRC performs processing required for forward recovery of BWO-eligible non-RLS files.

The functions of the RMLK gate are processed by DFHFCDW, which performs syncpoint and recovery functions for recoverable coupling facility data tables.

The functions of the RMDE gate are passed through by DFHFCRC to DFHFCIR. For START_DELIVERY, DFHFCIR takes no action. For DELIVER_RECOVERY and DELIVER_FORGET, DFHFCIR uses the log records that are delivered to it to rebuild file control structures representing the recoverable parts of each unit of work, and also rebuilds locks for non-RLS files. For END_DELIVERY, DFHFCIR notifies file control that the rebuilding of recovery information at CICS restart is now complete.

The functions of the LGGL gate are processed by DFHFCLF. For ERROR, DFHFCLF takes actions to handle a log stream failure for a general log used by file control.

The functions of the DMEN gate are processed by DFHFCES. For NOTIFY_SMSVSAM_AVAILABLE, DFHFCES calls DFHFCRR with a function of RESTART_RLS and TYPE_OF_RESTART as DYNAMIC.

Exits

A number of global user exit points are provided for file control.

In DFHEIFC
XFCREQ and XFCREQC

In DFHFCFS
XFCSREQ and XFCSREQC

In DFHFCN
XFCNREC and XFCRLSCO

In DFHFCRC
XFCBFAIL, XFCBOUT, XFCBOVER and XFCLDEL

In DFHFCRO
XFCRLSCO

The following global user exit points are provided specifically for data table services: XDTAD, XDTLC, and XDTRD.

Trace

The following point IDs are provided for file control:

- AP 04xx, for which the trace levels are FC 1, FC 2, and Exc
- AP 0Bxx, for which the trace levels are FC 1, FC 2, and Exc.
- AP 23xx, for which the trace levels are FC 1, FC 2, and Exc.
- AP 24xx, for which the trace levels are FC 1, FC 2, and Exc.

Note: Trace entries for shared data table services have point IDs at the lower end of the AP 0Bxx range, and a corresponding trace level of FC 2. Trace entries for coupling facility data tables are from AP 2440 upwards.

Front end programming interface (FEPI)

The front end programming interface (FEPI) is an integral part of CICS Transaction Server. The function is called a front end programming interface because it enables you to write CICS application programs that access other CICS or IMS programs. In other words, it provides a front end to those programs.

Design overview

This section describes how FEPI works at a high level. It discusses how the FEPI functions are provided within CICS.

FEPI as a CICS transaction

The main functions of FEPI are provided through the **CSZI** transaction, which is defined in group DFHFEPI. CSZI runs the FEPI Resource Manager, which is responsible for most of the functions of FEPI.

The FEPI Resource Manager transaction is attached during a late stage of CICS initialization. CSZI runs as a high-priority CICS system task, and cannot be canceled by an operator; it is terminated during CICS shutdown processing.

The FEPI commands communicate with the Resource Manager through the FEPI adapter program, which is loaded when CICS initializes, and is part of the CICS nucleus.

The FEPI adapter receives information from FEPI commands through two EXEC stubs, **DFHESZ** and **DFHEIQSZ**. DFHESZ handles the FEPI application programming commands, while DFHEIQSZ handles the system programming commands.

These two EXEC stubs call the adapter to do FEPI work. The adapter communicates with the Resource Manager through work queues. See "Application flows" for details of these flows.

Application flows

"FEPI as a CICS transaction" outlined the main components of FEPI. This section shows the pathways followed by a FEPI command.

Application programming command flows: The FEPI application programming commands flow through the normal EXEC CICS route into DFHEIP, from where they are routed to DFHESZ. DFHESZ passes the command parameter list to the FEPI adapter. After checking and other processing, the adapter generates another parameter list in internal format, and places it on a queue for the FEPI Resource Manager to process.

While the adapter is waiting for the Resource Manager to process the command, it issues a wait. The event control block (ECB) for this wait is contained in the parameter list queued to the Resource Manager. Consequently, the application that issued the FEPI command is in a wait state while the Resource Manager is processing the FEPI command. For information about wait processing, see Dealing with waits.

When the Resource Manager has retrieved the command from its queue, and processed it, the ECB is posted, thus ending the wait.

Control returns from the adapter to DFHEIP, and the application program in the normal fashion.

Figure 51 shows this processing. Note that the details are for illustration only.

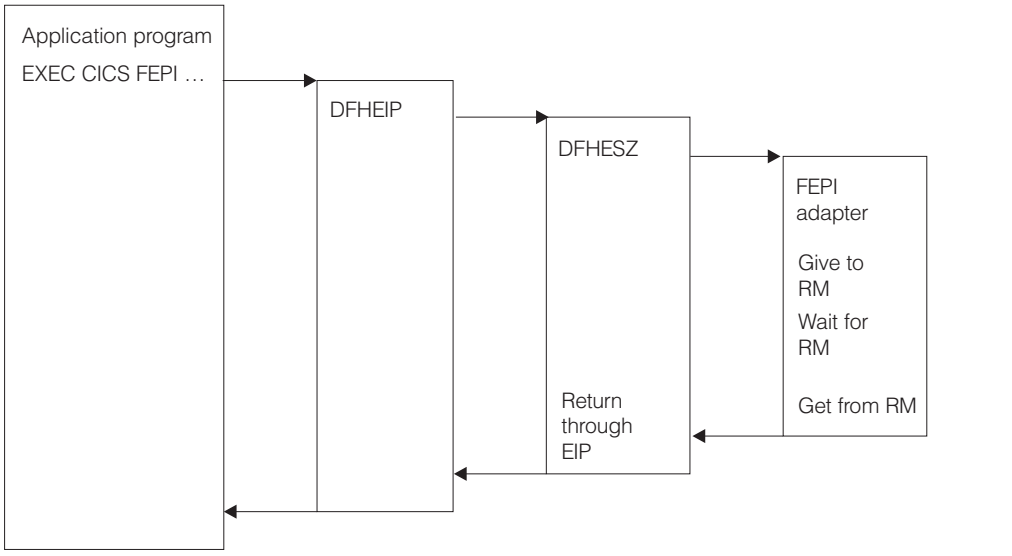


Figure 51. FEPI application programming command flows

System programming command flows: The FEPI system programming commands flow through DFHEIQSZ rather than DFHESZ, but the overall picture is the same as for FEPI application programming requests.

However, some system commands can flow directly to the FEPI Resource Manager, bypassing the EXEC stub. These commands are mainly concerned with FEPI processing to be done at "special" events, such as task termination and CICS shutdown.

Figure 52 shows this processing. The details are for illustration only.

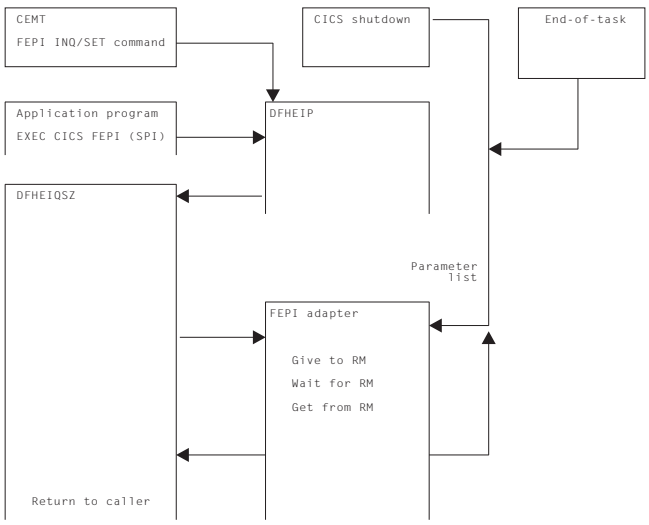


Figure 52. FEPI system programming command flows

Logic flow within the FEPI adapter: Figure 53 shows the logic flow within the FEPI adapter in more detail. In particular, it shows the points at which the FEPI global user exits, XSZBRQ and XSZARQ, and the FEPI journaling function, are invoked.

Journaling of data occurs after the Resource Manager has processed the request, but before XSZARQ is called (if active). Data is not journaled if your XSZBRQ exit program rejects the request.

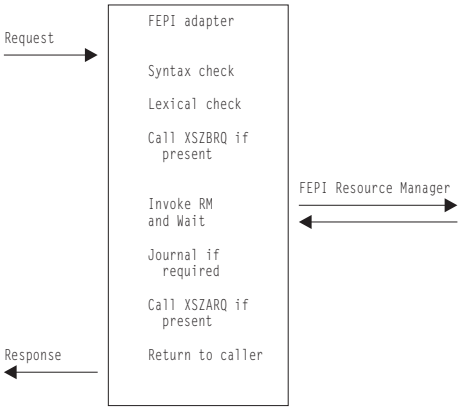


Figure 53. Logic flow within the FEPI adapter

The FEPI adapter and Resource Manager: The FEPI adapter runs as part of the invoking CICS task, and so runs under the **QR** task control block (TCB). The FEPI Resource Manager, running as CSZI, runs under the **SZ** TCB (reserved for use by the Resource Manager).

Consequently, the interface between the adapter and the Resource Manager uses waits and queues to synchronize access. The control block used to pass information between the adapter and the Resource Manager is called the **DQE**.

Figure 54 shows this interaction. The details are for illustration only.

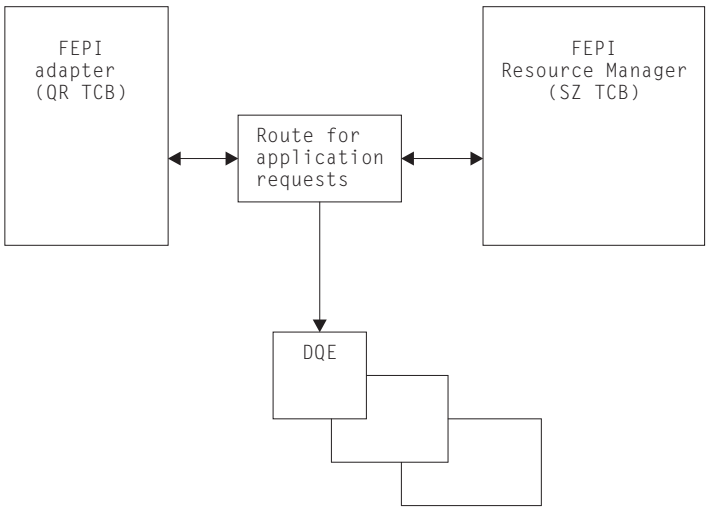


Figure 54. Interaction of the FEPI adapter and Resource Manager

The FEPI Resource Manager work queues

When organizing its work, the FEPI Resource Manager uses a mechanism that is optimized for the FEPI environment. Each DQE is chained to a queue representing the work to be done next.

The most common mechanism used for this movement between queues is the connection on which the original FEPI command is operating.

Summary of Resource Manager work queues: In addition to the application queue, there are other queues used only by the Resource Manager. They are:

API/Norm
: Used for FEPI application requests

API/Expd
: Used for FEPI high-priority application requests

PRB Used for Resource Manager internal work

PRB/Time
: Used for Resource Manager internal time-dependent work

IRB Used to control work done in z/OS Communications Server exits

IRB/Time
: Used to control time-dependent work done in z/OS Communications Server exits

TPEND8
: Used to process z/OS Communications Server TPEND8 conditions

Timer Used to control timer-related work

Free Used to hold z/OS Communications Server RBs that have to be freed

Discard
: Used to control requests initiated by FEPI DISCARD commands.

CICS work
: Used to schedule work that has to run under the CICS QR TCB.

Control blocks

This section lists *some* of the FEPI control blocks and their resident storage subpools, where applicable. For details of the subpools, see "Storage manager domain (SM)" on page 1063.

DFHSZSDS (Static area)
: Used to anchor all FEPI storage

DFHSZDCM (Common area)
: Used to anchor all FEPI Resource Manager storage (*SZSPFCCM*)

DFHSZDND (Node)
: Represents a node (*SZSPFCND*)

DFHSZDPD (Pool)
: Represents a pool (*SZSPFCPD*)

DFHSZDTD (Target)
: Represents a target (*SZSPFCTD*)

DFHSZDPS (Propertyset)
: Represents a property set (*SZSPFCPS*)

DFHSZDCD (Connection)
: Represents a connection (a node-target pair) (*SZSPFCCD*)

DFHSZDCV (Conversation)
: Represents a FEPI conversation (*SZSPFCCV*)

DFHSZDSR (Surrogate)
: Used to associate nodes, pools, and targets with other control blocks—*not* to be confused with a CICS surrogate terminal (*SZSPFCSR*)

DFHSZDQE (Queue element)
Used to schedule Resource Manager work (*SZSPFCWE*).

Some of the relations between FEPI control blocks are shown in Figure 55.

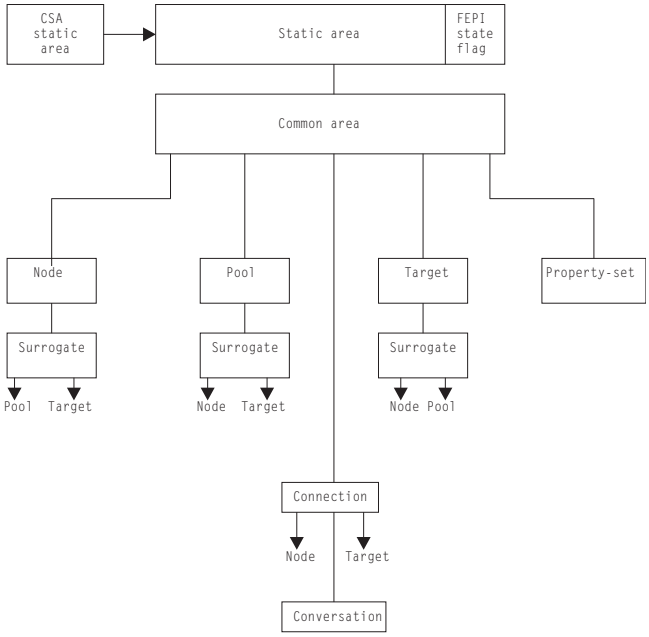


Figure 55. FEPI control block relationships

Dump

This section documents the areas that can be listed by the FEPI dump routines. For information about how to use these facilities for problem determination, see Formatting system dumps.

Here is a list all the FEPI areas that can be interpreted. If an area does not exist in your system, it does not appear in the dump—no error message is produced.

- The static area
- The common area:
  - The temporary ACB.
- Property sets
- Pools:
  - Connections within the pool
  - Node surrogates chained to the pool
  - Target surrogates chained to the pool
  - Queued allocate DQEs waiting within the pool
- Nodes:
  - Connections used by the node
  - Pool surrogates chained to the node
  - Node's ACB
  - Node's RPL

  - Unsolicited BINDs queued to the node
- Targets:
  - Connections used by the target
  - Connections queueing on the target
  - Pool surrogates chained to the target
- Connections:
  - Current API request
  - Connection's RPL
  - Connection's RESP data
  - Formatted data extension:
    - Graphics plane
    - Attributes
    - Highlights
    - Color
    - Selection
    - Validation
- Active conversations
- Browse conversations
- Inactive conversations
- CICS work queues
- PRB DQEs
- PRB time DQEs
- IRB DQEs
- IRB time DQEs
- TPend8 DQEs
- Discard DQEs
- API normal DQEs
- API expd DQEs
- Timer DQEs
- Free RBs
- The stacks (level 2 only).

A DQE is interpreted further, as follows:
- The DRP representing the DQE
- The DQE associated storage
- Any horizontal DQE extension (chained) DQEs.

FEPI and the z/OS Communications Server

This section outlines how FEPI interacts with the z/OS Communications Server, and discusses the Communications Server control blocks and exits.

You should refer to *z/OS Communications Server: SNA Programming* for all information relating to the z/OS Communications Server programming.

z/OS Communications Server control blocks

FEPI uses standard z/OS Communications Server programming facilities for its communication. The way in which z/OS Communications Server control blocks interact with FEPI control blocks is as follows:

ACBs Each FEPI node represents a terminal connected to the partner system. Consequently, each node has an **access control block** (ACB). This ACB is opened when the node is acquired, and closed when the node is released.

NIBs Each FEPI target contains the applid of the back-end system. This is used to build a **node initialization block** (NIB), when a connection is acquired by issuing a z/OS Communications Server REQSESS request. In common with CICS data communication, the "confidential" flag is set off.

RPLs There are two types of **request parameter list** (RPL) used by FEPI:

- Each FEPI outbound request causes the generation of an RPL. This RPL lasts only for the duration of the FEPI request.
- Each FEPI node has a "Receive-Any" RPL. When an inbound flow occurs, this RPL is attached to the FEPI connection, and turned into a "Receive-Specific" RPL. When the flow has been received, a new "Receive-Any" RPL is generated and attached to the node.

z/OS Communications Server exits

FEPI communicates with the z/OS Communications Server as asynchronously as possible. Therefore, Communications Server exits are extensively used for FEPI communication.

The following Communications Server exits receive control at specific stages of the communication process:

DFASY
Processes the receipt of expedited-data-flow control indicators.

LOGON
Processes the receipt of a CINIT in which FEPI is acting as the primary logical unit (PLU).

LOSTERM
Processes the loss of a session.

NSEXIT
Processes:

- The failure of a process that was responded to positively
- A session outage
- The receipt of network service RUs.

SCIP Processes the receipt of session-control requests.

TPEND
Processes the termination of the Communications Server.

Modules

Module	Function
DFHSZATC	adapter command tables
DFHSZATR	adapter program
DFHSZBCL	clean up API requests at error routine
DFHSZBCS	RM collect statistics
DFHSZBFT	FREE transaction requests scheduler
DFHSZBLO	lost session reporter
DFHSZBRS	RM collect resource ID statistics
DFHSZBSI	signon exit scheduler

Module	Function
DFHSZBST	STSN transaction scheduler
DFHSZBUN	unsolicited data transaction scheduler
DFHSZBUS	RM unsolicited statistics recording
DFHSZDUF	dump formatting routine
DFHSZFRD	formatted 3270 RECEIVE support
DFHSZFSD	formatted 3270 SEND support
DFHSZIDX	SLU P queue install/discard exit
DFHSZPCP	SLU P flow controller
DFHSZPDX	SLU P drain completion exit
DFHSZPID	SLU P send data processor
DFHSZPIX	SLU P send completion exit
DFHSZPOA	SLU P send response processor
DFHSZPOD	SLU P receive data processor
DFHSZPOR	SLU P response processor
DFHSZPOX	SLU P receive specific response exit
DFHSZPOY	SLU P receive specific response processor
DFHSZPQS	SLU P REQSESS (request session) issuer
DFHSZPQX	SLU P REQSESS exit
DFHSZPSB	SLU P bind processor
DFHSZPSC	SLU P session controller
DFHSZPSD	SLU P SDT processor
DFHSZPSH	SLU P SHUTC processor
DFHSZPSQ	SLU P quiesce complete (QC) processor
DFHSZPSR	RESETSR processor CSECT
DFHSZPSS	SLU P STSN processor
DFHSZPSX	SLU P OPNSEC completion exit
DFHSZPTE	SLU P TERMSESS processor
DFHSZRCA	node control processor
DFHSZRCT	issue processor
DFHSZRDC	delete connection processor
DFHSZRDG	discard node processor
DFHSZRDN	delete node processor
DFHSZRDP	dispatcher
DFHSZRDS	discard property set processor
DFHSZRDT	discard target procsssor
DFHSZREQ	request passticket module
DFHSZRFC	FREE completion processor
DFHSZRGR	Dispatcher work queue processor
DFHSZRIA	allocate processor
DFHSZRIC	define connection processor
DFHSZRID	discard processor

Module	Function
DFHSZRIF	install free processor
DFHSZRII	install processor
DFHSZRIN	install node processor
DFHSZRIO	ACB open processor
DFHSZRIP	install pool processor
DFHSZRIQ	inquire processor
DFHSZRIS	install processor
DFHSZRIT	install target processor
DFHSZRIW	SET processor
DFHSZRNC	NODE processor
DFHSZRNO	NOOP processor
DFHSZRPM	timer services
DFHSZRPW	request preparation
DFHSZRQR	queue for REQSESS processing
DFHSZRQW	request queue processor
DFHSZRRD	RECEIVE request processor
DFHSZRRT	request release processor
DFHSZRSC	connection processor
DFHSZRSE	SEND request processor
DFHSZRST	START request processor
DFHSZRTM	recovery services
DFHSZRXD	EXTRACT processor
DFHSZRZZ	TERMINATE processor
DFHSZSIP	initialization processor
DFHSZVBN	copy NIB mask to real NIB
DFHSZVGF	get queue element FIFO
DFHSZVQS	REQSESS dispatcher
DFHSZVRA	z/OS Communications Server receive_any processor
DFHSZVRI	z/OS Communications Server receive_any issuer
DFHSZVSC	delayed bind processor
DFHSZVSL	SETLOGON request issuer
DFHSZVSQ	z/OS Communications Server feedback interpreter
DFHSZVSR	z/OS Communications Server feedback interpreter
DFHSZVSY	z/OS Communications Server feedback interpreter
DFHSZWSL	RPL exit after SETLOGON
DFHSZXDA	z/OS Communications Server DFASY exit
DFHSZXFR	RPL exit to free request block
DFHSZXLG	z/OS Communications Server logon exit
DFHSZXLT	z/OS Communications Server LOSTERM (lost terminal) exit
DFHSZXNS	z/OS Communications Server NSEXIT (network services) exit
DFHSZXPM	STIMER IRB exit routine

Module	Function
DFHSZXRA	z/OS Communications Server RECEIVE_ANY exit
DFHSZXSC	z/OS Communications Server SCIP (session control) exit
DFHSZXTP	z/OS Communications Server TPEND exit
DFHSZYLG	RPL exit following logon reject
DFHSZYQR	post for REQSESS processing
DFHSZYRI	z/OS Communications Server RECEIVE_ANY issuer
DFHSZYSC	z/OS Communications Server SCIP exit extension
DFHSZYSR	z/OS Communications Server feedback interpreter
DFHSZYSY	z/OS Communications Server feedback interpreter
DFHSZZAG	get RECEIVE_ANY request block
DFHSZZFR	free RECEIVE_ANY request block
DFHSZZNG	get session control request block
DFHSZZRG	get RPL request block
DFHSZ2CP	SLU2 flow controller
DFHSZ2DX	SLU2 drain completion exit
DFHSZ2ID	SLU2 send data processor
DFHSZ2IX	SLU2 send completion exit
DFHSZ2OA	SLU2 send response processor
DFHSZ2OD	SLU2 receive data processor
DFHSZ2OR	SLU2 response processor
DFHSZ2OX	SLU2 receive specific completion exit
DFHSZ2OY	SLU2 receive specific action module
DFHSZ2QS	SLU2 REQSESS issuer
DFHSZ2QX	SLU2 REQSESS exit
DFHSZ2SB	SLU2 bind processor
DFHSZ2SC	SLU2 session controller
DFHSZ2SD	SLU2 SDT processor
DFHSZ2SH	SLU2 SHUTC processor
DFHSZ2SQ	SLU2 QC processor
DFHSZ2SR	SLU2 RESETSR processor
DFHSZ2SX	SLU2 OPNSEC processor
DFHSZ2TE	SLU2 TERMSESS processor

Function shipping

Function shipping allows a transaction from one CICS system to access a resource owned by another CICS system. This topic does not contain information about function shipping using IP interconnectivity (IPIC).

For information about IPIC support for function shipping see Function shipping.

The CICS function shipping facility enables separate CICS systems to be connected so that a transaction in one system is able to retrieve data from, send data to, or

initiate a transaction in, another CICS system. The facility is available to application programs that use the command-level interface of CICS.

Design overview

Figure 56 gives an overview of the function shipping component of CICS.

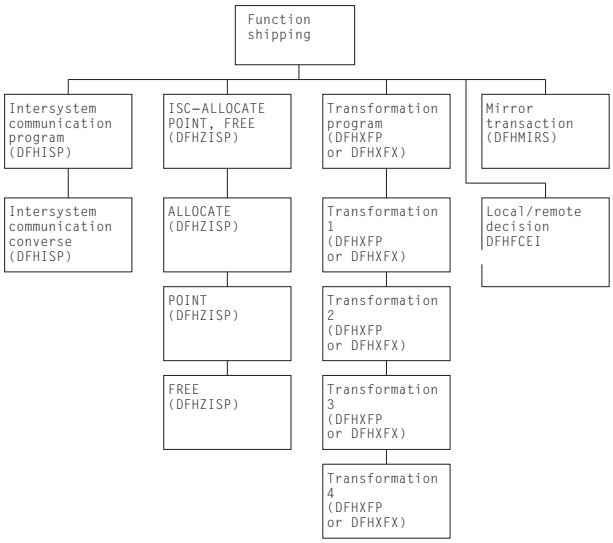


Figure 56. CICS function shipping

This section provides an overview of the operation of CICS when it is being used to communicate with other connected CICS systems for CICS function shipping.

Note: CICS function shipping gives a full description of the reasons for CICS function shipping and how the user can take advantage of the facility.

Application programming functions with CICS function shipping

The functions that CICS provides are extended for CICS function shipping so that an application program can issue the following types of command and have them executed on another system:

- Temporary storage commands
- Transient data commands
- Interval control commands
- File control commands
- DL/I calls
- Program link commands (DPL).

Application programs can use these extended functions without needing to know where the resources are located; information about the location of resources is in the appropriate definitions that the system programmer creates. Alternatively, an application program can name a remote system explicitly for a particular request.

Support for syncpoints, whether explicit (through EXEC CICS SYNCPOINT commands) or implicit (through DL/I TERM calls), allows updates to be made in several systems as part of a single logical unit of work.

Error handling routines might need to be extended to handle additional error codes that might be returned from a remote system.

Local and remote names

For a transaction to access a resource (such as a file or transient data destination) in a remote system, it is usually necessary for the local resource table to contain an entry for the remote resource. The name of this entry (that is, the name by which the resource is known in the local system) must be unique within the local system. The entry also contains the identity (SYSIDNT) of the remote system and, optionally, a name by which the resource is known in the remote system. (If this latter value is omitted, it is assumed that the name of the resource in the remote system is the same as the name by which it is known in the local system.)

Mirror transactions

When a transaction issues a command for a function to run on a remote system, the local CICS system encodes the request and sends it to the remote system. The remote system attaches one of the CICS-supplied mirror transactions, which all use the mirror program, DFHMIRS.

The CVMI and CPMI mirror transactions service requests sent as part of an LU6.2 synclevel 1 conversation. The CPMI, CSMI, CSM1, CSM2, CSM3, and CSM5 mirror transactions service requests that are sent as part of an LU6.2 synclevel 2 conversation, an LU6.1 conversation, an MRO conversation, or an IPIC conversation.

For distributed program link (DPL) requests shipped from a CICS application region to a CICS resource region, you can specify the name of the mirror transaction to be attached. If you specify your own mirror transaction, you must define the transaction in the resource region and associate it with the CICS-supplied mirror program, DFHMIRS.

A mirror transaction runs the request from the initiating transaction and returns to the local system the response code and any control fields and data that are associated with the request. If running the request causes the mirror transaction to abend, this information is also returned to the initiating transaction.

If a resource has browse place holders or is recoverable, or the lock has been acquired, the mirror transaction becomes a *long-running mirror* and does not end until the issuing transaction ends the logical unit of work; that is, a SYCNPOINT or RETURN. Any resources that the mirror has acquired are freed when the initiating transaction issues the appropriate command to free those resources.

The CICS-supplied mirror program DFHMIRS is defined as a threadsafe program. For IPIC connections only, CICS runs DFHMIRS on an L8 open TCB whenever possible. For other connection types, CICS does not run DFHMIRS on an open TCB. For threadsafe applications that issue commands for functions on remote CICS systems over IPIC connections, the reduction in TCB switching improves application performance compared to other intercommunication methods. For file control commands, to gain the performance improvement, you must specify the system initialization parameter **FCQRONLY=NO** in the file-owning region.

Initialization of CICS for CICS function shipping

If CICS has been generated with the appropriate options for intercommunication, the initialization of CICS with the ISC=YES system initialization parameter specified causes the following modules to be loaded:

- DFHISP (intersystem communication program)

- DFHXFP (data transformation program)
- DFHXFX (optimized data transformation program).

The entry point addresses of these modules are contained in the optional features list, which is addressed by CSAOPFLA in the CSA.

The mirror program, DFHMIRS, is not loaded until a request is received from a remote system. DFHMIRS can only be loaded if its PROGRAM resource definition is installed, and TRANSACTION resource definitions are installed for the mirror transactions CSMI, CSM1, CSM2, CSM3, and CSM5 or the transactions CVMI and CPMI. All these resource definitions are in the group DFHISC, which is included in DFHLIST.

Note: The ISC=YES system initialization parameter causes other modules besides those specified earlier to be loaded; the ones mentioned here are those specifically required for CICS function shipping.

Communication with a remote system

For multiregion operation, communication between CICS systems can be implemented:

- Through support in CICS terminal control management modules and by use of a CICS-supplied interregion program (DFHIRP) loaded in the link pack area (LPA) of MVS. DFHIRP is invoked by a type 3 supervisory call (SVC). The SVC moves the data to an intermediate area in key 0 MVS CSA storage, and schedules an SRB to move the data from the intermediate area to the target.
- By the cross-system coupling facility (XCF) of MVS. XCF is required for MRO links between CICS regions in different MVS images of an MVS sysplex. It is selected dynamically by CICS for such links, if available.

For ISC, communication between CICS systems takes place via ACF/Communications Server links. CICS and the CICS application programmer are independent of, and unaware of, the type of physical connection used by ACF/Communications Server to connect the two systems.

Protocols

Requests and replies exchanged between systems for CICS interval control, CICS transient data, CICS temporary storage, and DL/I functions are shipped using the standard protocol as defined for SNA logical unit type 6.1.

Requests and replies for CICS file control functions are shipped using a private protocol (with function management headers of type 43).

Symmetrical bracket protocol: Logical unit type 6.1 (LU6.1) sessions between two CICS systems require most protocols to be symmetrical; therefore, CICS receives (as well as sends) end bracket.

Shutdown protocol: The LU6.1 shutdown protocol does not use the SHUTDOWN command; it uses the data flow control commands SBI (stop bracket initiation) and BIS (bracket initial stopped). Shutdown is executed as part of session termination (by DFHZCLS) and ensures that, when a session is terminated normally (as a result of a master terminal release command or a normal CICS shutdown), there are no unfinished syncpoint requests on the session. This means that when the session is initiated, no resynchronization sequence is required.

Sender error recovery protocol (ERP): CICS support for LU6.1 uses a symmetrical SNA protocol called **Sender ERP**. In addition, when CICS wants to send a negative

response to a remote system, it sends a special negative response (0846), which indicates that an ERP message is to follow. This ERP message contains the real system and user sense values, together with a text message. The negative response and ERP message are built by DFHZEMW, and are received and processed by DFHZRAC, DFHZRVX, and DFHZNAC.

Resynchronization protocol: CICS support for LU6.1 sessions that use the syncpoint protocol has associated resynchronization logic, which is used during the initiation of a session after a previous session has terminated abnormally. This logic is used to generate messages concerning the outcome of any logical units of work that were **in doubt** when the previous session failed. The modules involved are DFHZRSY, DFHZSCX, and DFHZNAC.

CICS function shipping environment

This section describes the system entries for function shipping in the terminal control table, and how function shipping requests or replies are transformed between the format suitable for transmission and the internal parameter list format.

System entries in the terminal control table:

All remote systems with which a given system is able to communicate are identified and described in terminal control table system entries (TCTSEs). The name of the system entry is the name specified in the SYSIDNT field of the CICS table entry describing a remote resource.

CICS uses the TCTSE as an anchor point to queue requests made by CICS transactions for connection to the remote system.

Figure 57 on page 1711 shows three TCTTEs. If a transaction fails and you get a transaction dump, this figure shows you how to find the relevant TCTTEs from the TCA.

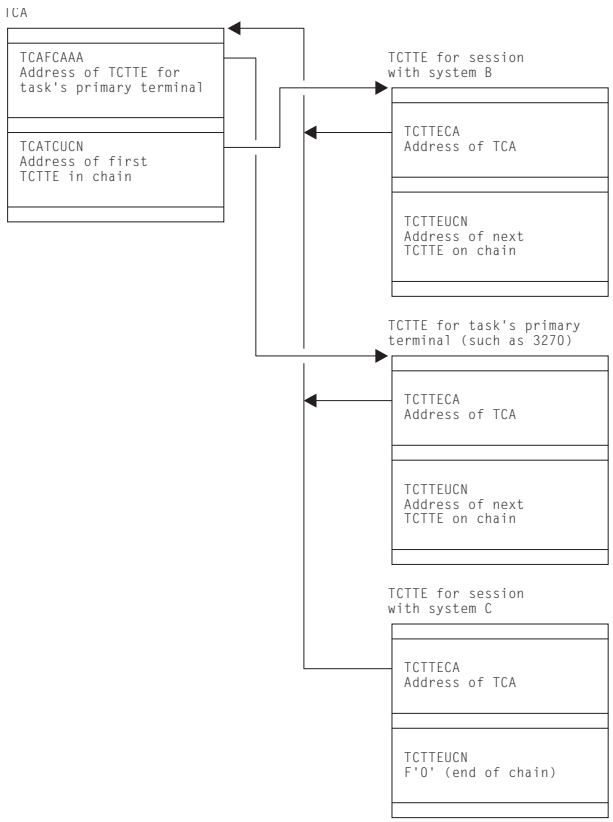


Figure 57. Task's view of CICS function shipping TCTTEs

Transformation of requests and replies for transmission between systems:

Before a request or reply can be transmitted, it must be transformed from its internal, parameter list (EXEC interface) format to a format suitable for transmission; when received after transmission, the request must be transformed back into a parameter list format.

There are four such transformations (numbered 1 through 4), which are performed by DFHXFP, or by DFHXFX if optimized data transformations are possible. The latter applies only to data transformations for function shipping in an MRO environment, excluding those relating to DL/I requests.

Transformation 1
: For a request to be sent by the originating system; transforms from parameter list format to transmission format.

Transformation 2
: For a request received by the mirror transaction; transforms from transmission format to parameter list format.

Transformation 3
: For a reply to be sent by the mirror transaction; transforms from parameter list format to transmission format.

Transformation 4
: For a reply received by the originating system; transforms from transmission format to parameter list format.

The parameter list format refers to the parameter list that is normally passed to DFHEIP (for CICS requests) or to DFHDLI (for DL/I requests).

The transmission formats of these requests and replies (excluding those for sync point protocol) are described in the DFHFMHDS DSECT.

Information that DFHXFP and DFHXFX must retain between transformations 1 and 4 (in the originating system) or between transformations 2 and 3 (in the mirror system) is stored in a transformer storage area called XFRDS. See XFR - Function shipping request control block for a detailed description of this control block.

CICS function shipping: handling of EXEC CICS commands

This topic describes the sending and receiving of requests and replies (other than DL/I or syncpoint requests) between two connected systems at the **application-layer** level; see Figure 58 on page 1713. (The **function management** and **data flow control** layers, implemented by CICS terminal control, work in the same way, regardless of the type of request being transmitted.)

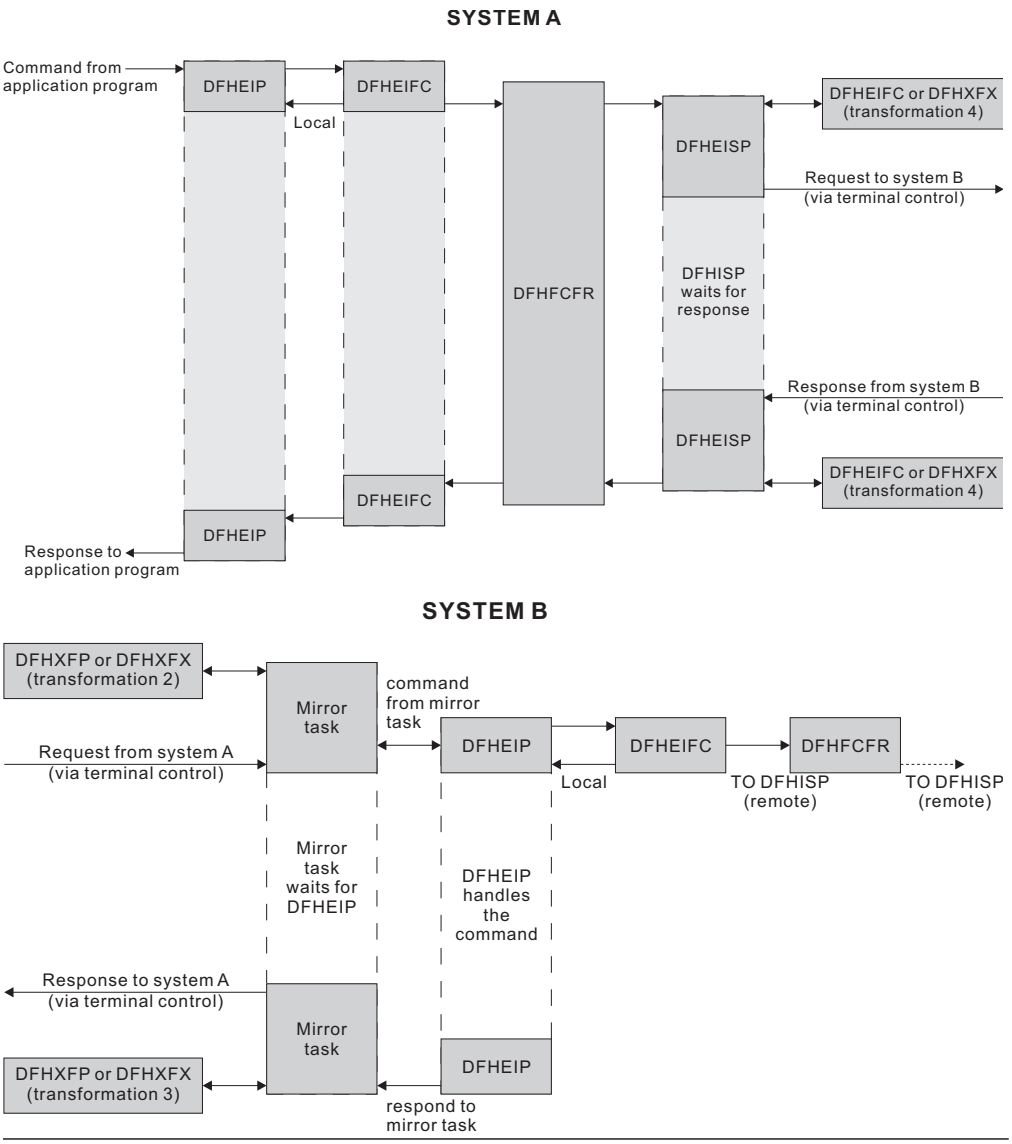


Figure 58. Overview of CICS function shipping

Sending a request to a remote system:

A CICS command is handled for an application program by the EXEC interface program, DFHEIP.

DFHEIP analyzes the arguments of each statement to determine the requested function and to assign values into the appropriate CICS control blocks; DFHEIP also performs storage control and error checking on behalf of the application programmer.

If the system has been initialized with the ISC=YES system initialization parameter, and if the request is for one of the functions that could be executed on a remote system (see "Application programming functions with CICS function shipping" on page 1707), DFHEIP invokes a local/remote decision routine, which inspects the appropriate CICS table to determine whether the request is for a local or a remote resource (unless a remote system has specifically been requested). For all requests except file control, this local/remote decision is taken in DFHEIP. For file control requests, the decision is taken in the file control function shipping interface module, DFHFCRF (see "File control" on page 1588).

If the resource is local:

- DFHEIP invokes the appropriate EXEC interface processor module to process the request locally.
- DFHEIFC calls the file control file request handler, DFHFCFR, to process the request locally, and finally returns control to DFHEIP.

Note: A SYSID value that names the local system also causes the request to be processed locally.

If the resource is remote, DFHEIP or DFHFCRF:

1. Allocates a transformer storage area (XFRDS) chained off the EXEC interface storage EIS. XFRDS provides a central area in which all information about processing of the request can be accessed.
2. Places the following data in XFRDS:
   - Name of remote system, for subsequent use by DFHISP (in XFRDS field XFRSYSNM)
   - Address of the application's list of parameters (EXEC parameter list) associated with the command being executed (in XFRDS field XFRPLIST)
   - Address of the table containing the definition for the requested resource (in XFRDS field XFRATABN).
3. Issues a DFHIS TYPE=CONVERSE macro, which passes control to the CICS function shipping program DFHISP.

DFHISP obtains the address of the TCTSE for the remote system and places it in XFRDS field XFRATCSE. DFHISP obtains the address of the TCTTE that controls the session with the remote system and places it in XFRDS field XFRATCTE. (DFHISP obtains the address by issuing a DFHTC TYPE=POINT macro. If no session is established, there is no TCTTE; in this case DFHISP issues a DFHTC TYPE=ALLOCATE macro to establish the session TCTTE.)

If no session can be allocated because, for example, all sessions are out of service, DFHISP determines whether or not the function request can be queued for shipping at a later time. If it the request can be queued, then XFRATCTE is set to zero.

Optionally (if a TIOA already exists from an earlier CICS function shipping request from the same application), DFHISP also places the address of the TIOA in XFRDS field XFRATIOA.

DFHISP then invokes DFHXFP, or DFHXFX for optimized transformations, to transform the requested command and parameter list into a form suitable for transmission. This is known as **transformation 1**, which:

1. Transforms the original **command** into an appropriate type of request for transmission.
2. Converts the EXEC parameter list into a **data unit** having a standardized character-string format (together with a function control header) suitable for transmission. The data unit is built in the TIOA and contains a copy of each of the parameters that are addressed by the EXEC parameter list. (For economy of transmission, certain types of data are compressed before being placed in the TIOA.)
3. Returns control to DFHISP.

Note: If local queuing is in effect, the data unit is built in user storage.

DFHISP then invokes terminal control to transmit the contents of the TIOA to the remote system and waits for the reply from the remote system, if necessary.

If local queuing is in effect, DFHISP issues a DFHIC TYPE=PUT macro specifying transaction CMPX, which sends the data unit at a later time.

Receiving a request at a remote system: Terminal control receives the request transmission and attaches one of the mirror transactions.

The mirror program allocates space for XFRDS in its LIFO storage area. As in the requesting system, XFRDS is a central area in which all information about the processing of the received request can be accessed. The mirror program places the following data in XFRDS:

- Address of the session TCTTE (in XFRDS field XFRATCTE)
- Address of the TIOA (in XFRDS field XFRATIOA).

The mirror program also allocates scratch pad storage in the LIFO storage area for use by DFHXFP (or DFHXFX) in building argument lists. The address of this storage is placed in XFRPLIST.

The mirror program then invokes DFHXFP, or DFHXFX for optimized transformations, to transform the received request into a form suitable for execution by DFHEIP. This is known as **transformation 2**, which:

1. Transforms the received request (as coded in the function management header of the data unit) into an appropriate CICS command.
2. Decodes the TIOA and builds (in the ***first*** part of the STORAGE area) an EXEC parameter list that basically consists of addresses that point to fields in the TIOA. (Those fields that were compressed for transmission are expanded and placed in the ***second*** part of the STORAGE area; for these fields, the EXEC parameter list points to the expanded versions, not the compressed versions in the TIOA.)

   Note: The NOHANDLE option is specified on each EXEC CICS command that is created; this has the effect of suppressing DFHEIP's branching to an error routine.

3. Returns control to the mirror program.

The mirror program then invokes DFHEIP (in the same way as for an application program), passing to it (in register 1) the address of the EXEC parameter list just built.

DFHEIP or DFHFCRF determines whether the request is for a remote resource on yet another system or for a local resource. If the resource is remote, DFHEIP or DFHFCRF allocates a new and separate transfer storage area XFRDS and invokes DFHISP (as described under "Sending a request to a remote system" on page 1714).

If the resource is local, the reply is processed for the mirror program in the usual way.

Sending a reply at a remote system: The process of sending a reply in response to a request from another system is similar to that for sending a request; see "Sending a request to a remote system" on page 1714.

When DFHEIP has successfully completed execution of the command, control is returned to the mirror program with the results of the execution in the EXEC interface block (EIB). The mirror program then invokes DFHXFP, or DFHXFX for optimized transformations, to transform the command response into a suitable form for the transmission of the reply. This is known as **transformation 3**, which:

1. Checks whether the existing TIOA is long enough to take the reply; if not, DFHXFP (or DFHXFX) frees the existing TIOA and creates a new one.
2. Converts the EXEC parameter list (kept in the scratch pad area STORAGE) into a **data unit** having the standardized character-string format suitable for transmission. The data unit is built in the TIOA. If the request is received by the mirror program without CD (that is, the requesting system did not expect a reply), the mirror program issues a DFHTC TYPE=READ or TYPE=FREE macro. If an error is detected, the mirror program is forced to abend, so that at least a record of the request failure is written.
3. Returns control to the mirror program.

The mirror program then invokes terminal control to transmit the TIOA. (The mirror program does this by issuing a DFHTC TYPE=(WRITE,WAIT,READ) macro if the mirror program holds any state information that must be held for a further request or until a syncpoint. Otherwise, a DFHTC TYPE=(WRITE,LAST) macro is issued.

Receiving a reply from a remote system: Terminal control receives the reply and returns control to the initiating task; in particular, control is passed to DFHISP, which has been waiting for the reply.

DFHISP invokes DFHXFP, or DFHXFX for optimized transformations, (passing to it the address of the XFRDS area) in order to transform the reply into the form expected by the application program. This is known as **transformation 4**, which:

1. Obtains the addresses of the TIOA and of the original EXEC parameter list from XFRATIOA and XFRPLIST in the XFRDS area.
2. Uses data in the reply to complete the execution of the original command. For example:
   - Sets return codes in the EIB from status bits in the reply

- Stores other received data (if any) in locations specified in the original EXEC parameter list.

3. Frees the TIOA.
4. Returns control to DFHISP.

DFHISP returns control to DFHEIP (if appropriate through DFHEIFC), which raises any error conditions associated with return codes set in the EIB. DFHEIP then returns control to the application program.

CICS function shipping—handling of DL/I requests

DL/I requests are handled in a similar manner to that for CICS commands; see Figure 59 on page 1718.

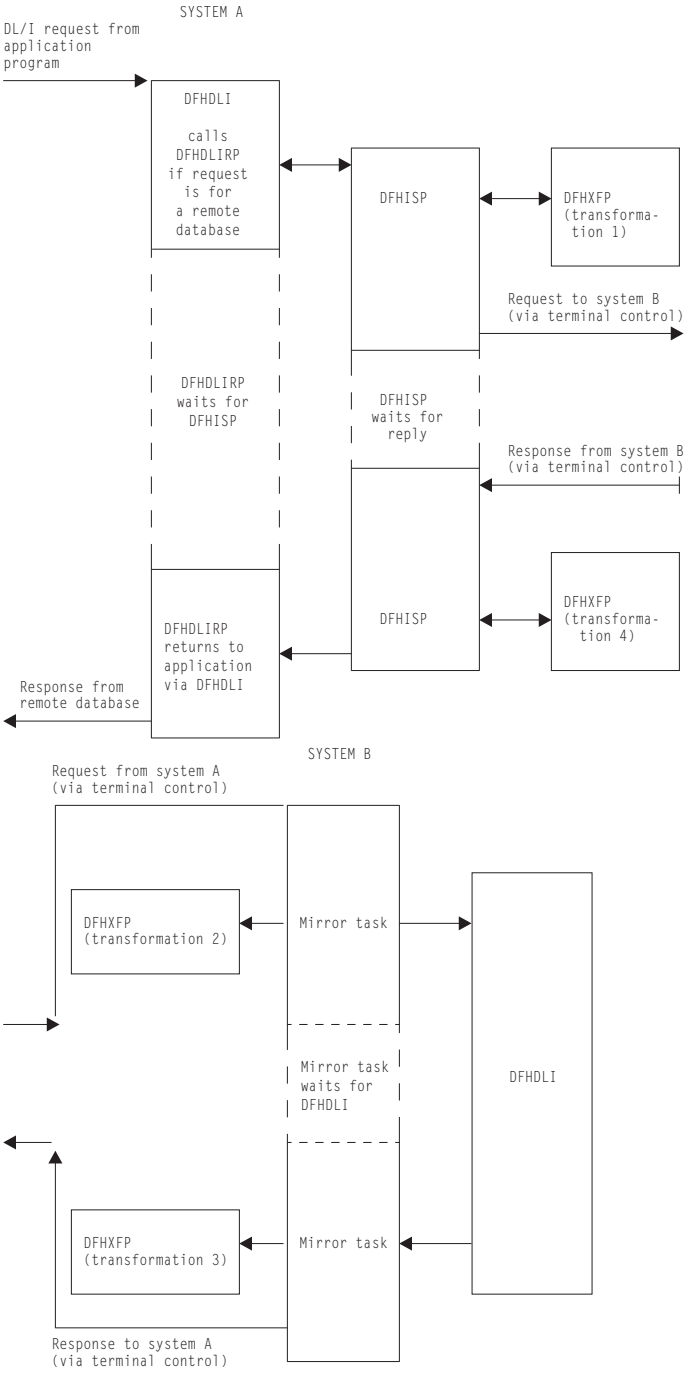


Figure 59. Overview of CICS function shipping of DL/I requests

Sending a DL/I request to a remote system: All DL/I requests are handled by DFHDLI.

DFHDLI determines whether the request is for a remote, or DBCTL database, and routes the request to the appropriate DL/I call processor. If the request is for a remote database, DFHDLI invokes DFHDLIRP, which passes control to DFHISP by issuing a DFHIS TYPE=CONVERSE macro.

DFHISP then:

1. Invokes DFHXFP to transform the request into a form suitable for transmission
2. Invokes terminal control to transmit the request.

Receiving a DL/I request at a remote system: As for a CICS request, the appropriate mirror transaction (in this case, CSM5) is attached.

The mirror program invokes DFHXFP to transform the received request into a form suitable for execution by DFHDLI.

The mirror program then passes the request to DFHDLI in the same way as any other application program would. DFHDLI determines what type of DL/I request is being made and then routes the request to the appropriate DL/I call processor: DFHDLIRP (remote, that is, daisy-chained to yet another system), or DFHDLIDP (DBCTL).

Sending a DL/I reply at a remote system: When DFHDLI has successfully completed the request, control is returned to the mirror program with the results in the user interface block (UIB). The mirror program then:

1. Invokes DFHXFP to transform the results into a form suitable for transmission
2. Invokes terminal control to transmit the reply.

Receiving a DL/I reply from a remote system: On receipt of the reply, terminal control returns control to DFHISP, which has been waiting for the reply; DFHISP then invokes DFHXFP to transform the reply into a form that can be used by DFHDLI. DFHXFP sets the return codes in an intermediate control block, DFHDRX, so that they can ultimately be copied to the UIB or the TCA for the application program. Control is then returned from DFHISP through DFHDLIRP to DFHDLI, and finally back to the application program.

Terminal control support for CICS function shipping

Terminal control support for CICS function shipping falls into three main areas.

1. TCTTE allocation functions, ALLOCATE, POINT, and FREE. These functions are used mainly by DFHISP to allow a CICS transaction to own additional TCTTEs. These are session TCTTEs to remote systems; these functions are supported by DFHZISP.
2. Syncpoint functions, SPR, COMMIT, ABORT, and PREPARE. These functions are used by the recovery manager connectors to implement the syncpoint protocol; these functions are supported by DFHZIS1.
3. LU6.1 functions. These functions are used by users of terminal control to support the data flow control protocols used in a LU6.1 session.

The functions described in areas 1 and 2 in the previous list are extensions to the DFHTC macro that are intended for internal use by CICS control programs only; they are not documented in the user information.

TCTTE allocation functions: Terminal control provides the following TCTTE-related functions:

`ALLOCATE function`
This allocates to the requesting transaction a session TCTTE for communication with a remote system. The name of the remote system is passed as a parameter. The address of the allocated TCTTE or a return code is returned to the requester. DFHZISP uses the DFHZCP automatic transaction initiation (ATI) mechanism to allocate the session.

If the allocation request cannot be satisfied immediately, an automatic initiate descriptor (AID) is created, and is chained off the system entry; the AID is used to remember, and subsequently to process, the outstanding allocation request.

Parallel sessions can be allocated explicitly, or implicitly by reference to a remote resource; sessions are automatically initiated at allocation time, if necessary. They can also be initiated by a master terminal ACQUIRE command, or automatically during CICS initialization if CONNECT=AUTO is specified in the TCTTE.

POINT function
This causes terminal control to supply the requesting task with the address of a session TCTTE for a named remote system. The TCTTE must have been previously allocated to the requesting task.

FREE function
This detaches a TCTTE from the owning task and makes it available for allocation to another transaction. (The FREE function is the opposite of the ALLOCATE function.)

TERM=YES operand
This operand enables the issuer of a terminal control macro to select explicitly the TCTTE to which the requested function is to be applied. The address of the TCTTE to be processed is passed as a parameter of the request; the TCTTE must have been previously allocated to the requesting task.

FREE TCTTE indicator
This indicator is set as a result of the remote system issuing a (WRITE,LAST) or FREE request to show that the current conversation has finished and that the session should be freed by the current owner of the TCTTE. The receiver of the FREE indicator (usually DFHISP) must issue a FREE request.

Syncpoint functions: For ISC, terminal control provides the following syncpoint functions (the equivalent functions for IRC are provided by DFHZIS1):

SPR (syncpoint request) function
This request is issued by the recovery manager connector during syncpoint processing, and causes terminal control (DFHZSDR) to send a request that has a definite DR2 response requested. This tells the other side of the session that a syncpoint is required.

COMMIT function
This request is issued by the recovery manager connector when syncpoint has been completed. It causes a positive DR2 response to be sent, signaling the successful completion of syncpoint protocol.

ABORT function
This request causes either a negative DR2 response or an LUSTATUS command to be sent, indicating that a requested syncpoint operation could not be completed successfully, or that there has been an abnormal end of the current logical unit of work.

PREPARE function
This request causes an LUSTATUS command to be sent to the mirror in the remote system and indicates that a syncpoint should be taken.

z/OS Communications Server secondary half-session support: CICS acts as both the primary and the secondary halves of an LUTYPE6 session. To implement secondary half-session support, CICS z/OS Communications Server terminal control has to do two things:

1. Implement the secondary half of the data flow control and session control protocols that CICS already uses as a primary.
2. Use the secondary API provided by z/OS Communications Server.

The terminal control functions provided by CICS are independent of primary/secondary considerations. Differences between the primary and secondary z/OS Communications Server interfaces are contained within the CICS modules that issue the appropriate z/OS Communications Server request. The secondary support functions appear principally in the DFHZCP modules shown in Table 73.

Table 73. z/OS Communications Server secondary support functions

Modules	Function	Secondary function
DFHZSIM	Request LOGON	Use REQSESS macro
DFHZOPN	OPNDST	Use OPNSEC macro
DFHZSCX	SCIP exit	Receive and process BIND, STSN, SDT, CLEAR, and UNBIND commands
DFHZCLS	CLSDST	Use TERMSESS macro
DFHZRSY	Resynchronization	Build STSN responses
DFHZSKR	Respond to	Send responses to BIND, SDT, and STSN commands
DFHZRAC, DFHZRVX	Receive	Receive and process BID commands
DFHZATI, DFHZRVX, DFHZRAC	Bracket protocol	Implement secondary contention resolution using bracket protocol
DFHZNSP	Network services error exit	Handle secondary LOSTERM type of errors

NOCHECK option function handling

The transmission of a START NOCHECK command and associated data is handled in a slightly different manner from that for other CICS function shipping commands. Compared with the process described in "Function shipping" on page 1706, the major differences are:

- After DFHISP has allocated the session TCTTE to the requesting task, the transformation program DFHXFP (or DFHXFX) performs **transformation 1**. In addition, the transformation program detects that a START NOCHECK command is being processed and passes this fact to DFHISP in its return code. Accordingly, DFHISP issues a DFHTC TYPE=WRITE macro, which is deferred until syncpoint, return, or another function-shipped request on that ISC session.
- DFHISP returns to its caller.
- On the receiving system, DFHEIP handles the START NOCHECK command in the usual way and then terminates when the command has been executed; no response is sent back to the first system.

Exits

DFHISP has two global user exit points, XISCONA and XISLCLQ.

Trace

The following point ID is provided for the intersystem program:

- AP 00DF, for which the trace level is IS 1.

The following point IDs are provided for function shipping data transformation:

- AP D9xx, for which the trace level is IS 1.

Good morning message program

The CICS good morning program issues a good morning message for z/OS Communications Server logical units.

Design overview

This module is invoked by running the CSGM system transaction.

If a satisfactory OPNDST has occurred (detected in the OPNDST exit, DFHZOPX) and if a "good morning" message has been requested in the TYPETERM resource definition or the TCT TYPE=TERMINAL macro, an NACP request is queued. NACP issues a DFHIC TYPE=INITIATE for this transaction.

This module determines the terminal type, sets up the appropriate control characters, gets a TIOA, and writes the message.

For a 3270 terminal, if the operator has entered data before the message has been received, NACP may be invoked to handle intervention required. In this case the transaction is abended and the write operation terminated.

You specify the default message text in the GMTEXT system initialization parameter.

Modules

DFHGMM

Exits

The XGMTEXT global user exit point is provided in DFHGMM. For further information about this, see Good morning message program exit (XGMTEXT).

Trace

No trace points are provided for this function.

Interregion communication (IRC)

CICS multiregion operation (MRO) enables CICS regions that are running in the same MVS image, or in the same MVS sysplex, to communicate with each other. MRO does not support communication between a CICS system and a non-CICS system such as IMS. [1]

ACF/Communications Server and SNA networking facilities are not required for MRO. The support within CICS that enables region-to-region communication is called **interregion communication** (**IRC**). IRC can be implemented in three ways:

- Through support in CICS terminal control management modules and by use of a CICS-supplied interregion program, DFHIRP, loaded in the MVS link pack area. DFHIRP is invoked by a type 3 supervisory call (SVC).
- By MVS cross-memory services, which you can select as an alternative to the CICS type 3 SVC mechanism. Here, DFHIRP is used only to open and close the interregion links.

1. The external CICS interface (EXCI) uses a specialized form of MRO link to support: communication between MVS batch programs and CICS; DCE remote procedure calls to CICS programs.

- By the cross-system coupling facility (XCF) of MVS. XCF is required for MRO links between CICS regions in different MVS images of an MVS sysplex. It is selected dynamically by CICS for such links, if available.

This section describes the communication part of MRO. "Multiregion operation (MRO)" on page 1749 gives a brief description of multiregion operation.

Design overview

For information about the design and implementation of interregion communication facilities, and about the benefits of cross-system MRO, see the Getting started with intercommunication.

Control blocks

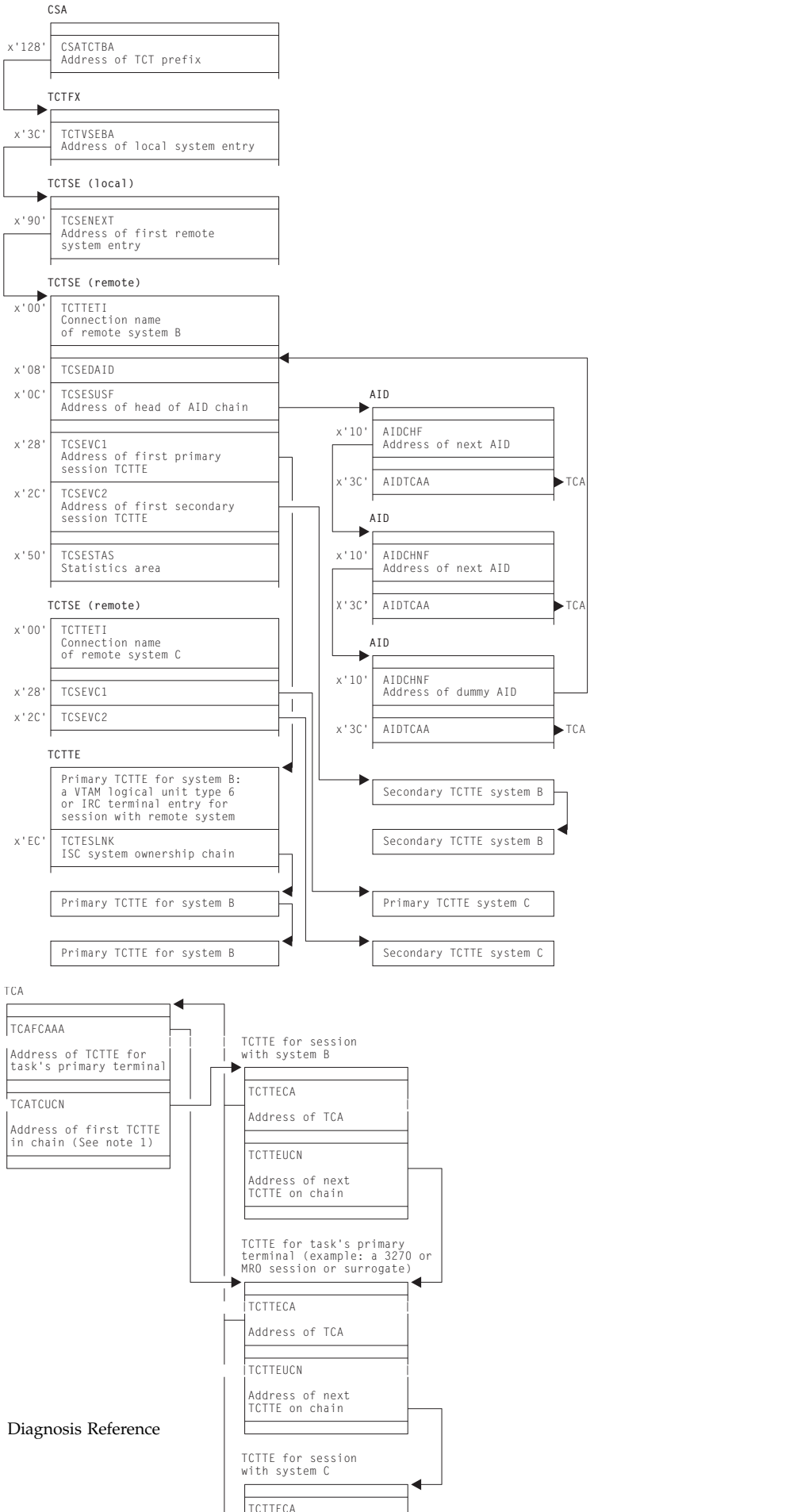
IRC uses two levels of control blocks:

1. A CICS/MRO terminal control layer
2. An interregion SVC layer interfaced by the DFHIR macro.

Terminal control layer

The CICS/MRO terminal control layer is shown in Figure 60 on page 1724.

This layer uses the cross-region block (CRB). This is a global (that is, one per CICS system) block that is created in the CICS dynamic storage area above the 16MB line (the ECDSA) when IRC is initialized, and provides information to communicate with the IRC SVC. See Figure 61 on page 1725.


CSA
x'128'
CSATCTBA
Address of TCT prefix
TCTFX
x'3C'
TCTVSEBA
Address of local system entry
TCTSE (local)
x'90'
TCSENEXT
Address of first remote
system entry
TCTSE (remote)
x'00'
TCTTETI
Connection name
of remote system B
x'08'
TCSEDAID
x'0C'
TCSESUSF
Address of head of AID chain
x'28'
TCSEVC1
Address of first primary
session TCTTE
x'2C'
TCSEVC2
Address of first secondary
session TCTTE
x'50'
TCSESTAS
Statistics area
AID
x'10'
AIDCHF
Address of next AID
x'3C'
AIDTCAA
TCA
AID
x'10'
AIDCHNF
Address of next AID
X'3C'
AIDTCAA
TCA
AID
x'10'
AIDCHNF
Address of dummy AID
x'3C'
AIDTCAA
TCA
TCTSE (remote)
x'00'
TCTTETI
Connection name
of remote system C
x'28'
TCSEVC1
x'2C'
TCSEVC2
TCTTE
Primary TCTTE for system B:
a VTAM logical unit type 6
or IRC terminal entry for
session with remote system
x'EC'
TCTESLNK
ISC system ownership chain
Primary TCTTE for system B
Primary TCTTE for system B
Secondary TCTTE system B
Secondary TCTTE system B
Primary TCTTE system C
Secondary TCTTE system C
TCA
TCAFCAAA
Address of TCTTE for
task's primary terminal
TCATCUCN
Address of first TCTTE
in chain (See note 1)
TCTTE for session
with system B
TCTTECA
Address of TCA
TCTTEUCN
Address of next
TCTTE on chain
TCTTE for task's primary
terminal (example: a 3270 or
MRO session or surrogate)
TCTTECA
Address of TCA
TCTTEUCN
Address of next
TCTTE on chain
TCTTE for session
with system C
TCTTECA

Notes:

1. The first TCTTE on the chain is not necessarily the TCTTE for the task's primary terminal.
2. A task has allocated MRO sessions to other systems.
3. TCTTEs are described more fully in "Terminal control" on page 1814.
4. Primary TCTTEs relate to Receive sessions, and secondary TCTTEs relate to Send sessions.
5. TCSEVC1 is the label on the address of the TCTTE of the first primary session. TCSEVC2 is that of the first secondary session.
6. The primary and secondary sessions each have sets of TCTTEs. These are found by using the DFHTC CTYPE=LOCATE macro.
7. A TCTTE is allocated for a surrogate session in transaction routing.

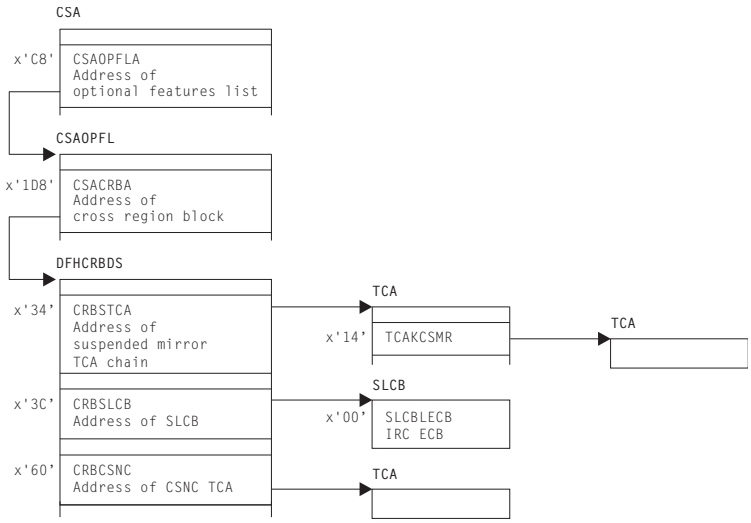


Figure 61. Cross-region block (CRB)

DFHIR layer

The interregion SVC layer interfaced by the DFHIR macro is shown in Figure 62 on page 1726.

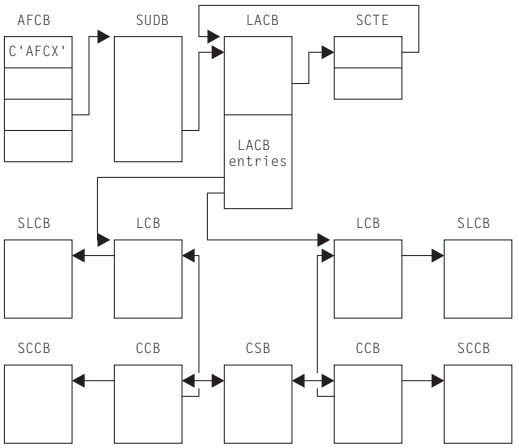


Figure 62. Interregion SVC layer of control blocks interfaced by the DFHIR macro

This layer uses the following control blocks, which, unless otherwise stated, reside in subpool 241 in MVS storage:

- Global (that is, one per MVS system) housekeeping (used by DFHIRP)
  Subsystem control table extension (SCTE)
  The SCTE is dynamically created, and contains information about the number of regions logged on to DFHIRP. It is used to locate the LACB. See also Figure 74 on page 1776, which shows the subsystem interface control blocks, including a pointer to the SCTE in the CICS subsystem anchor block (SAB).
  Logon address control block (LACB)
  The LACB contains entries to identify the regions that have logged on, and contains the address of the region's logon control block (LCB).
- Local housekeeping (used by DFHIRP)
  Logon control block (LCB)
  The LCB is created for each successful log on.
  Logon control block entry (LCBE)
  The LCBE contains the basic control information for each IRC system with which this system communicates. It addresses the connection control blocks (CCBs).
  Subsystem user definition block (SUDB)
  A SUDB provides access to IRC control blocks. There is one SUDB for each TCB that is currently logged on (so each SUDB may have multiple LCBs associated with it). The SUDB contains TCB-related data and working storage.
  Connection control block (CCB)
  A CCB is created for each IRC send-receive session, and contains information controlling the connection to the other region. When the connection is in use, it addresses the CSB.
  Connection status block (CSB)
  The CSB provides status information about the connection between two regions.
  MVS transfer buffers (MVS SRB mode)
  The MVS transfer buffers are used to transfer IRC data between regions, and reside in subpool 231 in MVS storage.

Terminal control layer and DFHIR layer

Figure 63 on page 1727 shows the control blocks that are accessed by both the terminal control layer and the DFHIR layer. Figure 64 on page 1727 shows the

location of these control blocks in MVS virtual storage.

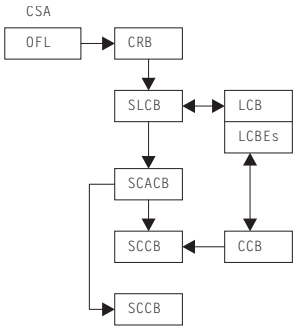


Figure 63. Control blocks accessed by CICS/MRO terminal-control layer of control blocks and by interregion SVC layer of control blocks

The following blocks are used by both the terminal control layer and the DFHIR layer. These blocks are allocated at logon time within a single MVS GETMAIN, and, unless otherwise stated, reside in subpool 251 of MVS storage.

Subsystem logon control block (SLCB)
The SLCB is used by the IRC SVC and region and contains the master ECB, posted when the region has IRC activity. It is pointed to by the CRB and LCB.

Subsystem connection address control block (SCACB)
The SCACB contains entries allowing the addressing of SCCBs from the SLCB.

Subsystem connection control block (SCCB)
The SCCB is created for each IRC send-receive session, and is allocated at logon. It contains the ECB, posted when input for the session is available.

Note: There is a one-to-one relationship between TCTTEs and SCCBs when they are in use.

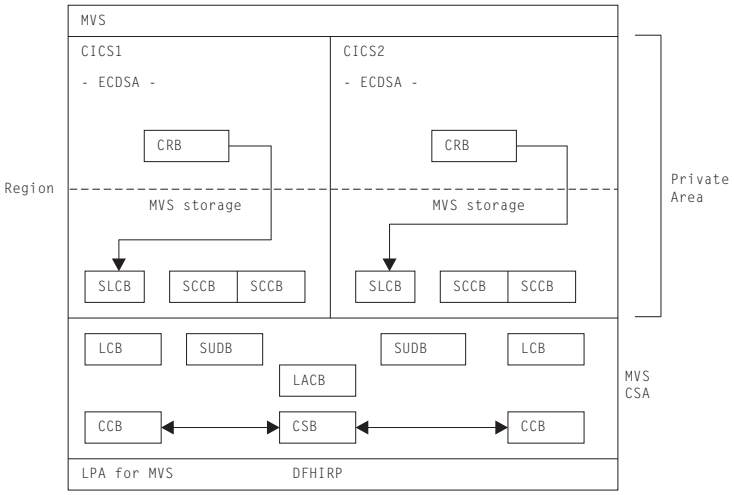


Figure 64. Location of control blocks in MVS virtual storage

MRO ECB summary

The following is a summary of the MRO event control blocks (ECBs):

```
Name           Location    Who waits                   Who posts
Dependent ECB    SCCB    Application (TC WAIT)          DFHIRP
LOGON ECB        SLCB    CICS (KCP, Op sys WAIT list)   DFHIRP
Link ECB         LCB     DFHIRP (Op sys WAIT)           DFHIRP
Work queue ECB   QUEUE   CSNC transaction               DFHIRP
                                                        DFHZIS2
                                                        DFHZLOC
```

See Data areas for a detailed description of the CICS control blocks.

Modules

Figure 65 gives an overview of the modules involved with interregion communication.

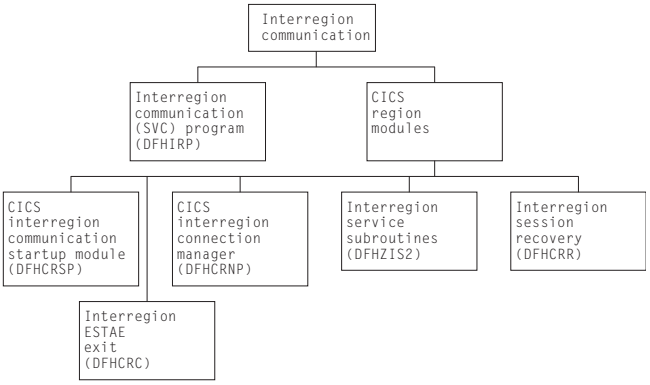


Figure 65. Interregion communication

The modules for IRC are of two types:

1. The interregion communication program: DFHIRP.
2. CICS address space modules: DFHCRC (interregion ESTAE exit), DFHCRNP (CICS interregion connection manager), DFHCRR (interregion session recovery), DFHCRSP (CICS interregion communication startup module), DFHZCP (CICS terminal management program), and DFHZCX (which includes DFHZIS2, the interregion service subroutines).

DFHIRP (interregion communication (SVC) program)

The interregion communication program (DFHIRP) is used to pass data from one region to another in the same processing unit. The programs running in the regions usually are CICS programs, but DFHIRP does not assume that to be the case.

Each user of this program must first issue a LOGON request specifying an 8-character name. This user identifier is added to a table maintained in key 0 storage. If cross-memory is being used, acquire and initialize the cross-memory resources (authorization index (AX), linkage index (LX), and entry table (ET)), unless this has already been done by a previous logon in this address space.

After the user has logged on, CONNECT requests can be issued to establish data paths to other users who have also logged on. The users must cooperate in this process by specifying, when they log on, to whom and from whom they are to be connected and by how many data paths. If cross-memory is being used, update the authority tables (ATs) of both address spaces to allow each one to establish addressability to the other, unless this was done when a previous connection was established between them.

After a connection has been established, either end of the connection can issue a SWITCH request to send data to the other end of the connection. The receiver of the data must provide a buffer into which the data is to be written. If the buffer is too small, the receiver is notified of the actual data length and, possibly having obtained a larger buffer, can issue a PULL request to retrieve as much data as is required. After the first data has been sent, the link must be used by each end alternately.

A connection can be broken by either end by issuing a DISCONNECT request. If cross-memory is being used and if the last cross-memory connection between a pair of address spaces is being removed, update the caller's AT so that the other system is no longer permitted to access the caller's address space.

When all links have been disconnected, a user can log off using a LOGOFF request. If cross-memory is being used, free the cross-memory resources acquired by logon if they are no longer required by the caller's address space.

When MVS cross-memory services are requested (ACCESSMETHOD(XM) in the RDO CONNECTION definition), communication is performed by DFHIRP running as an SVC. In this case, it is invoked by an SVC call to a startup program (DFHCSVC), which calls the required DFHIRP routine.

CICS address space modules

The CICS address space modules control the handling of requests between this address space and other address spaces. They include several MRO management modules such as DFHCRSP (see "DFHCRSP (CICS IRC startup module)") and DFHCRNP (see "DFHCRNP (connection manager—CSNC transaction)"), and several terminal-control modules (see "DFHZCX (CICS terminal control routines)" on page 1731).

These modules provide the CICS address space with a DFHTC-level interface to interregion communication (in the same way as DFHZCP provides a DFHTC-level interface to z/OS Communications Server). This enables other CICS modules (such as DFHISP) to allocate and execute input/output operations on IRC sessions. The IRC sessions are used for all forms of IRC communication, and the macro-level services available for IRC are broadly the same. Thus DFHISP works for both IRC and intersystem communication (ISC) function shipping.

The functions of each module are as follows:

DFHCRSP (CICS IRC startup module): Execution of this module makes interregion communication possible between this address space and other address spaces. DFHCRSP, which can be invoked either at system initialization or by the master terminal, allocates the cross-region block (CRB), issues a LOGON request to the SVC routine, and attaches the CSNC transaction (connection manager program, DFHCRNP).

DFHCRNP (connection manager—CSNC transaction): Interregion communication is controlled by the interregion control program, DFHCRNP, which runs as transaction CSNC. This is attached when CICS first logs on to the interregion program, and it remains attached until interregion communication is closed.

The main purpose of CSNC is to perform housekeeping and control on IRC sessions, and to simulate the access method. Its functions include the following:

1. Establish connections to other address spaces (by issuing CONNECT requests)

2. Detect unsolicited input data on connections and attach requested tasks to process such data
3. Disconnect unallocated (**between-bracket**) sessions during QUIESCE
4. Issue DFHKC AVAIL for any secondary sessions which have become available for reallocation, and are in demand
5. Issue PC RETURN when QUIESCE is complete.

CSNC is attached by DFHCRSP (IRC startup), and waits when it is not processing work. It is resumed by the dispatcher when the MRO work queue ECB has been posted, or the delay interval (if set) has expired and there is delayed work to be retried.

Whenever CSNC is posted, it checks first whether it has been invoked because quiescing of the interregion facility is complete.

- If CSNC has no been resumed to complete interregion quiesce processing, it checks each of the following:
  1. If the "delay-queue" is not empty, CSNC attempts to process any work it finds there. (An element is added to the queue whenever a transaction cannot be attached by CSNC. The system could, for example, have been at maximum tasks or short on storage when the previous attempt was made. It is also possible that a remote system tried to start a new conversation before the local system had freed the required session from an earlier conversation.)
  2. If a new conversation has been received:
     - If this is the first conversation on a new connection, and the connecting region is not a batch region, session recovery is performed. This means that if the name of the secondary connecting matches the name of the secondary connected in the previous session, the old session is bound once again.
     - If there is no match, or if a batch region is connecting, the first available session is allocated.
     - CSNC attempts to attach the required transaction, identified in the attach header included in the data stream. It is possible for a request to arrive for this session before the session has been freed from the transaction that last used it. In such a case, the transaction to be attached is added to the delay-queue.
     - The input data stream is built into a TIOA for the session.
  3. If this region is a secondary, and there is no task associated with the connection, and the connection is in quiesce, CSNC disconnects the session.
  4. If this region is a primary, and it has received a "disconnect" request from the connected secondary, CSNC disconnects the session if:
     - There is no associated TCTTE
     - There is no task associated with the link.
- If CSNC has been resumed to complete interregion quiesce processing, it:
  1. Sends message DFHIR3762 to the CSMT log.
  2. Resumes any suspended mirror tasks with a facility address of zero, so they can detach themselves.
  3. Disable immediate and delay queues. Any remaining work on those queues (for example, old retry work which has not been serviced yet) is automatically discarded.
  4. Logs off from the interregion SVC.
  5. Detaches, using a DFHLFM TYPE=RETURN request.

DFHCRR (CICS session recovery module): Whenever a new connection is established (via a successful CONNECT request), DFHCRNP links to DFHCRR at the secondary end of the connection (that is, at the source of the connection). DFHCRNP sends a data stream down to the other end of the connection (the primary end) which causes DFHCRNP to link to DFHCRR at the primary end. The two DFHCRRs exchange information in order to determine whether either end of the connection was in doubt when the previous use of the connection was terminated, and, if so, whether the two ends were in sync or out of sync. In the case of an indoubt connection, the sequence numbers are compared, diagnostics are issued, and the session is freed.

DFHCRC (interregion abnormal exit module): This module contains the ESTAE exit routine corresponding to the ESTAE macro issued by DFHKESIP. It is invoked if the ESTAE exit, DFHKESTX, decides to continue the abend, or if an X22 abend (which can't be handled by DFHKESTX) occurs.

The purpose of the exit is to free links with other subsystems to which connection has been made by the interregion SVC, and to free links with the SVC itself. This is done by issuing to the SVC a CLEAR request (to break links with other subsystems).

DFHZCX (CICS terminal control routines): DFHZCX is a load module consisting of a set of object modules, including DFHZIS1 (ISC or IRC syncpoint) and DFHZIS2 (IRC internal functions).

DFHZIS2 provides the following routines:

I/O request routine (IORENT)
Provides a WRITE/WAIT/READ interface to interregion connections.

GETDATA routine (GDAENT)
Retrieves input data from an IRC connection and puts it into a TIOA.

RECEIVE routine (RECENT)
Receives unsolicited data (**begin-bracket** in SNA terms) and checks validity.

DISCONNECT routine (DSCENT)
Cleans up this end of a connection, and issues DISCONNECT request to DFHIRP.

OPRENT routine (OPRENT)
Issues an INSRV request to DFHIRP, in order to allow future connections between this subsystem and a specified subsystem.

RECABRT routine (RCAENT)
Is invoked when an ABORT FMH (FMH07) is received (indicating that the connected transaction has abended). The routine issues a message describing the failure.

STOP routine (STPENT)
Is invoked when communication with other address spaces is to be terminated. The routine issues a QUIESCE request to DFHIRP.

LOGOFF routine (LGFENT)
Is invoked when quiesce is complete (and during system termination and abend processing). The routine issues a LOGOFF request to the SVC routine.

DFHZIS1 also contains routines representing terminal control services which are supported by IRC (in common with z/OS Communications Server). These routines include PREPARE, SPR, COMMIT, and ABORT.

DFHZCP (CICS terminal management program): DFHZCP is a load module consisting of a set of object modules, including DFHZARQ (application request handler), DFHZISP (intersystem program allocation routines), and DFHZSUP (startup task).

DFHZARQ is used (in common with all other telecommunication access methods) to handle WRITE/WAIT/READ-level requests against IRC connections (sessions). Routine ZARQIRC in DFHZARQ specifically handles IRC requests by performing SNA request header processing and invoking IORENT (see DFHZCX) in order to perform the I/O on the session.

DFHZISP includes routines such as ALLOCATE and FREE.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:

- AP DDxx, for which the trace levels are IS 1 and IS 2.

Intersystem communication (ISC)

CICS intersystem communication (ISC) allows the following:

- CICS-to-CICS communication
- CICS-to-IMS communication
- CICS-to-LUTYPE6.2 terminal or application communication.

These can be used simultaneously in the same or a different MVS image. ISC can use z/OS Communications Server LU6.1 or LU6.2 (LU6.2 is preferred for CICS operation). For information about these methods of communication, see the Getting started with intercommunication.

The facilities provided by ISC include:

- Transaction routing
- Distributed transaction processing
- Function shipping
- Asynchronous processing
- Distributed program link
- SAA Communications interface.

For information about the design and operation of intersystem communication, see “z/OS Communications Server LU6.2” on page 1885. For descriptions of the facilities provided by ISC, see “Transaction routing” on page 1845, “Distributed transaction processing” on page 1549, “Function shipping” on page 1706, and “SAA Communications and Resource Recovery interfaces” on page 1764.

Interval control

Interval control provides various optional task-related functions based on specified intervals of time, or specified time of day.

Design overview

The following services are performed by interval control in response to a specific request from either an application program or another CICS function:

Time of day

The EXEC CICS ASKTIME command retrieves the current time-of-day in either binary or packed decimal format.

Time-dependent task synchronization

Time-dependent task synchronization provides the user with three optional services:

1. The EXEC CICS DELAY command allows a task to temporarily suspend itself for a specified period of time. When the time has elapsed, the task resumes execution.
2. The EXEC CICS POST command allows a task to be notified when the specified interval of time has elapsed or the specified time of day occurs. The task proceeds to execute while the time interval is elapsing.
3. The EXEC CICS CANCEL command allows a task to terminate its own or another task's request for a DELAY, POST or START service.

Automatic time-ordered transaction initiation

Automatic time-ordered transaction initiation provides for the automatic initiation of a transaction at a specified time of day (or after a specified interval of time has elapsed) and for the sending of data that is to be accessed by the transaction. The user can also cancel a pending request for automatic time-ordered transaction initiation.

Optional user exits are provided as follows:

- Before determining what type of request for time services was issued
- Upon expiration of a previously requested time-dependent event
- If a START request names an unknown terminal.

Time-of-day control

The **EXEC CICS PERFORM RESETTIME** command causes CICS to reset its internal date and time of day information in accordance with that of the operating system.

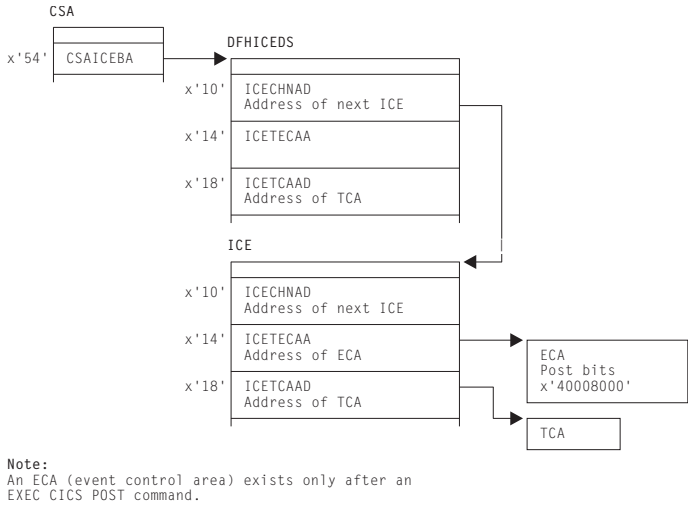
The **EXEC CICS PERFORM RESETTIME** command calls DFHICP with a DFHIC TYPE=RESET macro. This macro is also issued by DFHAPTIM - the program run by the "midnight task" attached by interval control initialization - whenever it is resumed by the TI domain, i.e. at midnight.

DFHICP issues a KETI RESET_LOCAL_TIME call to the TI domain if the reason for the reset was a time of day change. This allows the TI domain to readjust its clocks to the operating system time. DFHICP then calls DFHTAJP to readjust other CICS clocks to match the operating system time and to make any necessary changes to the ICE chain resulting from possible changes in the time-to-expiry of time controlled ICEs. Finally DFHICP scans the ICE chain in order to process any that may have become expired as a result of the time change, and to reset the time interval for which the expiry task, DFHAPTIX, will wait, until the next ICE expires.

Control blocks

An interval control element (ICE—see Figure 66 on page 1734) is created for each time-dependent request received by interval control. These ICEs are chained from

the CSA in expiration time-of-day sequence.



Note: An ECA (event control area) exists only after an EXEC CICS POST command.

Figure 66. Interval control element (ICE)

Expired time-ordered requests are processed by Interval Control when called from the DFHAPTIX module, which runs under a system task that has been resumed by the timer domain. The type of service represented by the expired ICE is initiated, if all resources required for the service are available, and the ICE is removed from the chain. If the resources are not available, the ICE remains on the chain and another attempt to initiate the requested service is made later.

Modules

DFHAPTIM, DFHAPTIX, DFHICP, DFHICRC, and DFHTAJP

Exits

There are three global user exit points in DFHICP: XICEXP, XICREQ, and XICTENF.

Trace

The following point ID is provided for DFHICP:

- AP 00F3, for which the trace level is IC 1.

Language Environment interface

This section describes the run-time interface between CICS and Language Environment.

Design overview

Communication between CICS and Language Environment is made by calling a special Language Environment interface module (CEECCICS) and passing to it a parameter list (addressed by register 1), which consists of an indication of the function to be performed and a set of address pointers to data values or areas.

Module CEECCICS is distributed in the Language Environment library, but must be copied to an authorized library defined in the STEPLIB concatenation of the CICS startup job stream.

All calls to Language Environment are made directly from the CICS language interface module DFHAPLI. This module is called by several components of CICS to perform specific functions. Table 74 lists those functions, and shows the name of the CICS module initiating each function call and the Language Environment call made by DFHAPLI to support the function. The format of each call parameter list is given in "External interfaces" on page 1739.

Table 74. Language Environment interface calls

Function	Module	Language Environment call
Terminate Languages	DFHSTP	Partition Termination
Establish Language		Establish Ownership Type
	DFHPGLK	
	DFHPGLU	
	DFHPGPG	
Start Program		
	DFHPGLK	Thread Initialization
	DFHPGLU	Run Unit Initialization
		Run Unit Begin Invocation
		Run Unit End Invocation
		Run Unit Termination
		Thread Termination
Goto	DFHEIP	Perform Goto
Find Program Attributes	DFHEDFX	Determine Working Storage
Initialize Languages	DFHSIJ1	Partition Initialization

The logical relationship between the different calls is shown in Figure 67 on page 1736.

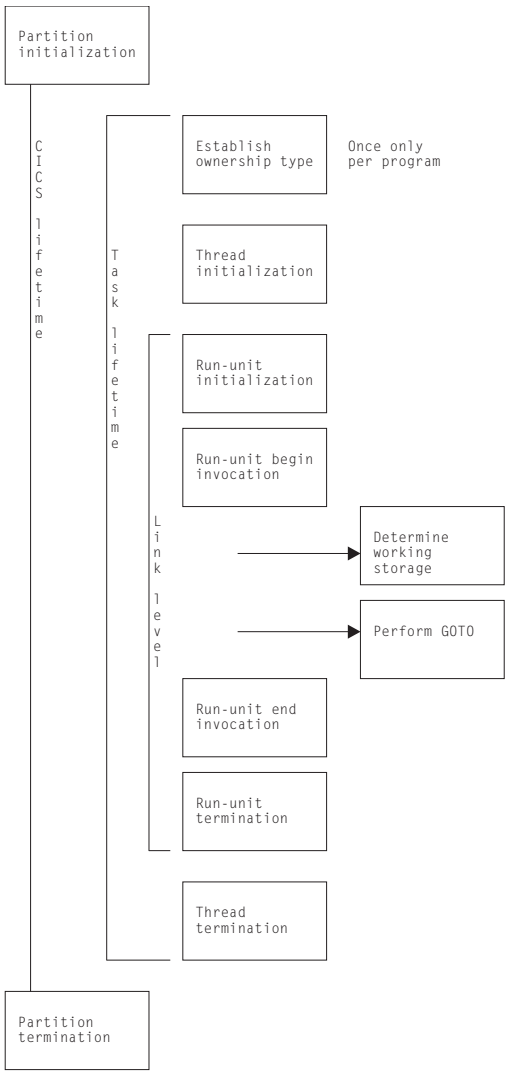


Figure 67. Language Environment interface components

Note: The actual passing of control to CEECCICS is made from the CICS language interface program (DFHAPLI), which provides a single point of contact between CICS and Language Environment. Other modules call DFHAPLI to initiate the intended function.

All calls to DFHAPLI use either the DFHAPLIM macro (for calls from outside the CICS application domain), or the DFHLILIM macro (for calls made from within the application domain).

Establishing the connection

The procedure for establishing the initial connection with Language Environment is as follows:

1. **Load CEECCICS.** At CICS startup, DFHSIJ1 invokes DFHAPLI to "initialize languages". DFHAPLI issues a BLDL for CEECCICS, followed by an MVS LOAD macro.
2. **Initialize contact with Language Environment.** Contact is first made with Language Environment by having CICS drive the partition initialization

function. DFHAPLI attempts partition initialization only if the earlier load of CEECCICS was successful; otherwise, the logic is bypassed.

If the Language Environment partition initialization is successful, and Language Environment indicates that it can support the running of programs in languages supported by CICS, a flag is set and no further processing takes place.

If the partition initialization function fails, CICS issues error message DFHAP1200.

Application program contact with Language Environment. Whenever a program written in a supported language is run, the application's attempt to make contact with Language Environment fails if the "Language Environment initialization bits" flag is not set. CICS then tries to run the program itself using the basic support for the language. If this fails, CICS then abends the transaction and sets the associated installed resource definition as disabled.

Storage for the transaction

A set of work areas is required during the lifetime of any task that includes one or more programs supported by Language Environment. This set is known as the "language interface work area".

The language interface work area contains storage for the following:

- The largest possible Language Environment interface parameter list (currently 15 parameter elements, but with space allowed for a further three elements)
- A general-purpose register save area for use by DFHAPLI
- A general-purpose register save area for use by Language Environment
- A 240-byte special work area for use by Language Environment as the equivalent of DFHEISTG for CICS
- A 4-byte Language Environment reason code field
- The IOINFO area (see "IOINFO" on page 1743)
- The PGMINFO1 area (see "PGMINFO1" on page 1744)
- The program termination block (see "Program termination block" on page 1745).

Also, a thread work area is required if Language Environment is involved in the running of the task. The length of a thread work area is a constant value that is notified to CICS by Language Environment during the partition initialization processing. This additional work area is built contiguous with the language interface work area if the transaction is known to contain one or more programs that use Language Environment. When such a program is first encountered, DFHAPLI:

1. Gets from the transaction manager the address of the transaction-related instance data.
2. Flags the data to tell the transaction manager that the transaction runs Language Environment application programs.
3. Adds the length of the language interface work area to the total user storage length for that transaction.

This forces the transaction manager to acquire extra storage, during task initialization, as an extension to the language interface work area. For the first occurrence only, DFHAPLI acquires the thread work area.

Further areas known as run-unit work areas (RUWAs) are required at run time if the transaction includes one or more programs that use Language Environment.

The length of an RUWA varies for each program. The lengths required for work areas above and below the 16MB line by Language Environment are notified to CICS during the processing to establish ownership type for that program; thereafter they can be found in the program's installed resource definition. CICS adds to the length for the RUWA above the 16MB line a fixed amount for its own purposes before acquiring the storage.

Storage acquisition

During task initialization, the transaction manager acquires an area of storage, the language interface work area, which is large enough to hold all required data for calls to Language Environment. This area is contiguous with the EXEC interface storage (EIS), and its address is saved in TCACEEPT in the TCA.

The thread work area is usually contiguous with the language interface work area. Its address is always held in CEE_TWA in the language interface work area.

For every link level entered during the execution of the application, a run-unit work area must be acquired by CICS and its address passed to Language Environment during run-unit initialization. Its address is placed in EIORUSTG in the EXEC interface storage (EIS).

Control blocks

The main control block is the language interface work area. It is addressed by TCACEEPT in the TCA. For programs supported by Language Environment, the work area is mapped by the Language_Interface_Workarea DSECT.

Modules

The Language Environment interface is accessed in the language interface program (DFHAPLI) in response to calls from the following modules:

`DFHSIJ1, DFHEIP, DFHEDFX, and DFHSTP.`

Exits

No global user exit points are provided for this interface.

Trace

Trace entries are made on entry to and exit from DFHAPLI.

Point IDs AP 1940 to AP 1945, with a trace level of PCÂ 1, correspond to these trace entries.

The function information is always interpreted.

For entry trace records, the program name and link level are also interpreted where applicable.

For exit trace records, the returned reason code is interpreted.

Also, all calls into and out of the language environments are traced at level 1. The point IDs are: AP1948 to AP 1952.

The ERTLI function named in the DFHAPLI entry trace is the function requested on the call, while that named in the DFHAPLI exit trace is the ERTLI function most recently processed. There are some situations in which a trace record made on entry to DFHAPLI is not matched by a corresponding exit trace for the same

ERTLI function. In particular, after making a call to Language Environment for thread initialization, DFHAPLI does ***not*** return to the caller, but proceeds with "run-unit initialization" and "run-unit begin invocation" before finally returning. Another example is the successful execution of a "perform GOTO" function, which results in DFHAPLI not returning to the caller.

Note: ERTLI refers to the Extended Run-Time Language Interface. This is an extension of the Run-Time Language Interface (RTLI) protocols that were defined to assist communication between CICS and both VS COBOL II and C/370. ERTLI includes communication between CICS and Language Environment.

External interfaces

This section describes the parameter lists and work areas used for the functions provided by the Language Environment interface.

Language Environment interface parameter lists

The following tables show the layout and contents of the parameter lists for the functions provided by the Language Environment interface module CEECCICS.

Table 75. Language Environment PARTITION_INITIALIZATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"10" (= Partition initialization)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token	Yes	8
6	EIBLEN	Length of CICS EIB		F'word
7	TWALEN	Thread work area length	Yes	F'word
8	CELLEVEL	Language Environment-CICS interface level	Yes	F'word
9	GETCAA	Get-CAA routine address		4
10	SETCAA	Set-CAA routine address		4
11	LANGDEF	Language modules defined		32
12	LANGBITS	Language availability bits	Yes	F'word

Table 76. Language Environment ESTABLISH_OWNERSHIP_TYPE parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"50" (= Establish ownership type)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	reserved			
7	reserved			
8	PGMINFO1	CICS-Language Environment program information		48

Table 76. Language Environment ESTABLISH_OWNERSHIP_TYPE parameter list (continued)

No.	Parameter name	Description	Receiver field	Data length
9	PGMINFO2	Language Environment-CICS program information	Yes	20

Table 77. Language Environment THREAD_INITIALIZATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"20" (= Thread initialization)		F′word
2	RSNCODE	Reason code	Yes	F′word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token	Yes	8
7	PREATWA	Address of preallocated thread work area		4
8	PGMINFO1	CICS-Language Environment program information		48
9	PGMINFO2	Language Environment-CICS program information		20

Table 78. Language Environment RUNUNIT_INITIALIZATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"30" (= Run-unit initialization)		F′word
2	RSNCODE	Reason code	Yes	F′word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token	Yes	8
8	PGMINFO1	CICS-Language Environment program information		48
9	PGMINFO2	Language Environment-CICS program information		20

Table 79. Language Environment RUNUNIT_BEGIN_INVOCATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"32" (= Run-unit begin invocation)		F′word
2	RSNCODE	Reason code	Yes	F′word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8

Table 79. Language Environment RUNUNIT_BEGIN_INVOCATION parameter list (continued)

No.	Parameter name	Description	Receiver field	Data length
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token		8
8	PGMINFO1	CICS-Language Environment program information		48
9	PGMINFO2	Language Environment-CICS program information		20
10	IOINFO	Input/output queue details		18
11	RSA	RSA at last EXEC CICS command		F'word

Table 80. Language Environment DETERMINE_WORKING_STORAGE parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"60" (= Determine working storage)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token		8
8	LANG	Program language bits		F'word
9	PGMRSA	Register save area address		4
10	WSA	Working storage address	Yes	4
11	WSL	Working storage length	Yes	F'word
12	SSA	Static storage address (reserved)	Yes	4
13	SSL	Static storage length (reserved)	Yes	F'word
14	EP	Program entry point	Yes	4

Table 81. Language Environment PERFORM_GOTO parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"70" (= Perform GOTO)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token		8
8	LANG	Program language bits		F'word
9	LABEL	Label argument at Handle		F'word
10	RSA	RSA at last EXEC CICS command		F'word
11	CALLERR	Cross call error flag	Yes	F'word

Table 81. Language Environment PERFORM_GOTO parameter list (continued)

No.	Parameter name	Description	Receiver field	Data length
12	ABCODE	Address of TACB abend code		F'word
13	R13	Register 13 value at abend		F'word

Table 82. Language Environment RUNUNIT_END_INVOCATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"33" (= Run-unit end invocation)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token		8
8	PGMINFO1	CICS-Language Environment program information		48
9	PGMINFO2	Language Environment-CICS program information		20
10	PTB	Program termination block		64
11	RSA	RSA at last EXEC CICS command		F'word

Table 83. Language Environment RUNUNIT_TERMINATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"31" (= Run-unit termination)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token		8
7	RTOKEN	Run-unit token	Yes	8

Table 84. Language Environment THREAD_TERMINATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"21" (= Thread termination)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8
6	TTOKEN	Thread token	Yes	8

Table 85. Language Environment PARTITION_TERMINATION parameter list

No.	Parameter name	Description	Receiver field	Data length
1	FUNCTION	F"11" (= Partition termination)		F'word
2	RSNCODE	Reason code	Yes	F'word
3	SYSEIB	Address of system EIB		4
4	PREASA	Preallocated save area		240
5	PTOKEN	Language Environment partition token		8

Work areas

The following sections describe the work areas required during the lifetime of any task that includes one or more programs that use the Language Environment interface.

IOINFO: The IOINFO area, which is built by DFHAPLI in the CICS-Language Environment work area, is passed to Language Environment on a RUNUNIT_BEGIN_INVOCATION call.

CICS applications cannot use the SYSIN and SYSPRINT data streams because such usage would conflict with the way CICS handles I/O. However, an application may require a general input or output data stream in some situations, for example, where it is necessary to output a message to a program and the program has not been written to expect such output under normal operation.

Three such data streams are designed for this purpose: input, output (normal), and error output. The destinations must be either spools or queues. CICS uses queues, so the file type is always set to "Q". Table 86 shows the transient data queue names that are passed to Language Environment.

Table 86. Transient data queues for use by Language Environment

File type	Language Environment queue name
Input	CESI
Output	CESO
Error output	CESE

Each data stream is identified by a 6-byte control block, and the three control blocks are concatenated to form the IOINFO area, which CICS passes to Language Environment.

IOINFO has this format (in assembler-language code):

```
IOINFO   DS    0CL18    Input/output queue details

STD_IN   DS    0CL6     Standard input file
QORS_IN  DS    CL1      ..file type - "Q" = transient data
TDQ_IN   DS    CL4      ..queue name
SPO_IN   DS    CL1      ..spool class - not used

STD_OUT  DS    0CL6     Standard output file
QORS_OUT DS    CL1      ..file type - "Q" = transient data
TDQ_OUT  DS    CL4      ..queue name
SPO_OUT  DS    CL1      ..spool class - not used

STD_ERR  DS    0CL6     Standard error output file
```

```
QORS_ERR DS    CL1      ..file type - "Q" = transient data
TDQ_ERR  DS    CL4      ..queue name
SPO_ERR  DS    CL1      ..spool class - not used
```

PGMINFO1:

The PGMINFO1 area, which is built by DFHAPLI in the CICS-Language Environment work area, is passed to Language Environment during specific interface calls.

The area is passed during these interface calls:
ESTABLISH_OWNERSHIP_TYPE
THREAD_INITIALIZATION
RUNUNIT_INITIALIZATION
RUNUNIT_BEGIN_INVOCATION
RUNUNIT_END_INVOCATION

When both CICS and Language Environment are capable of supporting it, the separate calls to Language Environment for RUNUNIT_INITIALIZATION and RUNUNIT_BEGIN_ INVOCATION are combined into a single call. This single call is a RUNUNIT_INITIALIZATION call with additional parameters to indicate the following information:

1. Whether it is the combined call
2. Whether the RUWA being passed has already been passed to Language Environment, and so need not be completely initialized by Language Environment.

PGMINFO1 has the following format (in Assembler language):

```
PGMINFO1      DS    0F
P1_LENGTH     DS    F         Length of PGMINFO1
RULANG        DS    XL4       Language as defined by user
ASSEMBLER     EQU   X'80'     ..Assembler
C             EQU   X'40'     ..C
COBOL         EQU   X'20'     ..COBOL
PLI           EQU   X'10'     ..PL/I
LE370         EQU   X'04'     ..Language Environment

RULOADMOD     DS    0F
RULOADA       DS    A         Run-unit load module address
RULOADL       DS    F         Run-unit load module length

ENTRY_STATIC  DS    0F
RUENTRY       DS    A         Run-unit entry point address
RUSTATIC      DS    A         Modified entry address
RWA_31        DS    A         Address of run-unit storage
                                  above 16MB
RWA_24        DS    A         Address of run-unit storage
                                  below 16MB
APAL          DS    A         Application argument list
                                  address
RTOPTS        DS    A         Run-time options
RTOPTSL       DS    F         Length of run-time options
RUNAMEP       DS    A         Pointer to the program name
PGMINFO1L     EQU   *-PGMINFO1
```

PGMINFO2

The PGMINFO2 area is filled in by Language Environment on successful completion of an ESTABLISH_OWNERSHIP_TYPE call, and is subsequently passed by CICS to Language Environment during these interface calls:
THREAD_INITIALIZATION

RUNUNIT INITIALIZATION
RUNUNIT_BEGIN_INVOCATION
RUNUNIT_END_INVOCATION

PGMINFO2 has this format (in assembler-language code):

```
PGMINFO2   DS    0F
PRGINLEN   DS    FL4       Length of PGMINFO2 extension
PLBRWA31   DS    F         Length of 31-bit RUWA
PLBRWAA    EQU   X'80'     ..31-bit storage required (C/370)
PLBRWAL    DS    FL3       ..Length of 31-bit RUWA
PLBRWA24   DS    F         Length of 24-bit RUWA

PLBLANG    DS    0CL4      Language availability byte
PLBLANG1   DS    X
PLBCEEEN   EQU   X'80'     ..Language Environment
                               enabled
PLBCEELA   EQU   X'40'     ..Language Environment
                               language known
PLBMIXED   EQU   X'20'     ..Mixed/single language
PLBCOMPT   EQU   X'10'     ..Compatibility
PLBEXECU   EQU   X'08'     ..Language Environment
                               executable
PLBASSEM   EQU   X'04'     ..Assembler language program
PLBC370    EQU   X'02'     ..C program
PLBCOBL2   EQU   X'01'     ..Enterprise COBOL or VS COBOL II program
PLBLANG2   DS    X
PLBOSCOB   EQU   X'80'     ..OS/VS COBOL program
PLBPLI     EQU   X'40'     ..PL/I program
PLBTYPE3   DS    X         Reserved
PLBTYPE4   DS    X         Reserved
PLBMEMID   DS    FL4       Language member ID
PLBED      EQU   *-PGMINFO2
```

Program termination block

The program termination block (PTB), which is built by DFHAPLI in the CICS-Language Environment work area, is passed to Language Environment on a RUNUNIT_END_INVOCATION call.

It has this format (in Assembler-language code):

```
CELINFO    DS    0F
PCHK       DS    0CL32     Abend information
           DS    CL8
PCHK_PSW   DS    CL8       ..PSW
PCHKINTS   DS    0CL8      ..Interrupt data
PCHK_LEN   DS    XL2       ../..Instruction length
PCHK_INT   DS    XL2       ../..Interrupt code
PCHK_ADR   DS    FL4       ..Exception address
PCHK_GR    DS    AL4       ..A(GP registers at abend)
PCHK_FR    DS    AL4       ..A(FP registers at abend)
PCHK_AR    DS    AL4       ..A(AX registers at abend)
PCHK_EX    DS    AL4       ..A(Registers at the last time
                               a CICS command was issued)
CNTCODE    DS    0CL4      Continuation code
CONT1      EQU   X'40'    ..retry using registers
CONT2      EQU   X'20'    ..retry using PSW
           DS    BL3      Reserved
RTRY       DS    0CL20
RTRY_AD    DS    FL4      ..Retry address
RTRY_PM    DS    AL4      ..A(Program mask)
RTRY_GR    DS    AL4      ..A(GP registers)
RTRY_FR    DS    AL4      ..A(FP registers)
RTRY_AR    DS    AL4      ..A(AX registers)
```

Master terminal program

The master terminal program enables dynamic control of the system. Using this function an operator can change the values of parameters used by CICS, alter the status of system resources, terminate tasks, and shut down the CICS system.

Design overview

The master terminal program is invoked by the CEMT transaction. The user enters a command to INQUIRE about or SET the status of a set of resources, and the command outputs a display that shows the resultant status of the resources. For a CEMT SET command, this display can be overtyped to alter the status of most of the resources displayed.

Commands are analyzed using the same techniques as the command interpreter described in "Command interpreter" on page 1532. A language table is used to define the syntax of commands and the contents of parameter lists which must be passed to DFHEIP to allow execution. In effect, each CEMT command results in the execution of a series of EXEC CICS INQUIRE and SET commands.

The master terminal program is also used by the CEST and CEOT transactions, which provide subsets of the functions available with CEMT. CEST is for supervisory operators and allows access to a limited set of resources. CEOT only allows changes to the status of the operator's own terminal.

Modules

Module	Function
DFHEMTP	Invoked by CEMT. Checks that the terminal is suitable. Obtains and initializes working storage. Loads the language table DFHEITMT. Links to DFHEMTD.
DFHEOTP	Same as DFHEMTP but invoked by CEOT and loads the language table DFHEITOT.
DFHESTP	Same as DFHEMTP but invoked by CEST and loads the language table DFHEITST.
DFHEMTD	Receives data from the terminal and sends back a display. Analyzes commands and overtypes. Constructs parameter lists for DFHEIP, which it calls. Deals with PF keys.
DFHEITMT	Command language table for CEMT.
DFHEITOT	Command language table for CEOT.
DFHEITST	Command language table for CEST.

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

Message generation program

The message generation program provides an interface for sending CICS messages to the terminal user only.

Design overview

The input to the message generation program (DFHMGP) consists of the binary number of the message to be produced, the identifier of the component issuing the message, and any information to be inserted in the message. DFHMGP builds the complete message using a prototype held in the message prototype control table, also known as the message generation table (DFHMGT). The message text itself is held not in DFHMGT but in the message domain, from which it is retrieved by the DFHMGPME routine (a component of the DFHMGP load module) when required. DFHMGP finally sends the message to the appropriate terminal.

The prototype statements are invocations of the DFHMGM TYPE=TEXT macro, and are contained in copybooks held in DFHMGT.

The message prototype control table consists of a series of copybooks, DFHMGTnn, each of which contains 1 through 100 messages. They are arranged in such a way that each DFHMGTnn copybook contains prototypes for messages that have identifiers of the form DFHccnnxx, where cc is the 2-character identifier of the component issuing the message, nn is the numerical part of the copybook name, and xx is in the range 00 through 99. For example, the prototype for message DFHAC2214 (belonging to the AC component) is in copybook DFHMGT22.

Within each copybook are invocations of DFHMGM in ascending message number order. All messages sent to the terminal user have both OPTION=NLS and COMPID specified on their DFHMGM invocations.

The main operands of the DFHMGM TYPE=TEXT macro are:

- MSGNO = actual message number
- COMPID = 2-character identifier of component issuing the message (this forms part of the message identifier)
- OPTION = any special options, for example, (NLS) for messages that require NLS enabling.

Other operands are provided on the DFHMGM invocations, but in general these are no longer used.

Modules

DFHMGP, DFHMGT

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:

- AP 00E0, for which the trace level is AP 1.

Message switching

This function provides the user with a general-purpose message-switching capability while CICS is running.

The facility, which can route messages to one or more destinations, is initiated by the CMSG transaction, or a user-chosen replacement, read from the terminal. For further information about this transaction, see CMSG - message switching.

Design overview

Message switching runs as a task under CICS. A terminal operator requests activation of this task by entry of the CMSG transaction identifier (or another installation-defined 4-character transaction identifier), followed by appropriate parameters. After it has been initiated, message switching interfaces with CICS basic mapping support (BMS) and CICS control functions.

Although message switching appears conversational to the terminal operator, the message switching task is terminated with each terminal response. Conversation is forced, if continuation is possible, by effectively terminating the transaction with an EXEC CICS RETURN TRANSID(xxxx) command, where xxxx is the transaction identifier specified in the task's TRANSACTION resource definition.

Figure 68 shows the message-switching interfaces.

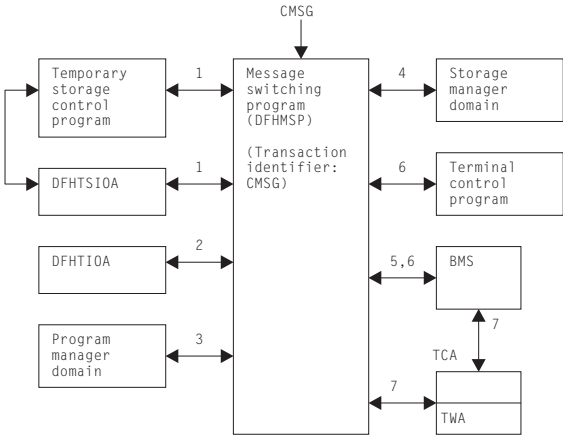


Figure 68. Message-switching interfaces

Note:

1. If the first 4 characters of the terminal input/output area (TIOA) (not including a possible set buffer address (SBA) sequence from an IBM 3270 Information Display System) do not match the transaction identifier for the task, this task must have started as part of a conversation in which a previous task has set up the next transaction identifier. A "C" immediately following the transaction identifier is also a forced continuation. In such a case, information has been stored in, and has to be retrieved from, temporary storage (using a record key of 1-byte X'FC', 4-byte terminal identifier, and 3-byte C"MSG") to allow the task to resume from where it was.
2. The operands in the input TIOA are processed and their values and status are stored in the TWA.
3. If a ROUTE operand specifies terminal list tables (TLTs) for a standard routing list, the program manager domain is called to load the requested TLTs.
4. Message switching requests storage areas for:
   - Building route lists (one or more segments, each of which has room for the number of destinations specified by MSRTELNG, an EQU within the program).

- Constructing a record to be placed in temporary storage.
- Providing the message text to BMS in any of the following situations:
  - Message parts from previous inputs exceed the current TIOA size
  - A message is completed in the current TIOA but has parts from previous inputs
  - A heading has been requested but the message in the current TIOA is too close to TIOADBA to allow the header to be inserted.

5. Message switching requests BMS routing functions by means of the DFHBMS TYPE=ROUTE macro. The message text is sent using DFHBMS TYPE=TEXTBLD, and completion of the message is indicated by DFHBMS TYPE=PAGEOUT. BMS returns the status of destinations and any error indications in response to the DFHBMS TYPE=CHECK macro.
6. Message switching interfaces with BMS using DFHBMS TYPE=(EDIT,OUT) and with CICS terminal control using DFHTC TYPE=WRITE for the IBM 3270 Information Display System only, in providing responses to terminals. These can indicate normal completion, signal that input is to continue, or provide notification of input error.
7. Like any other task, message switching has a task control area (TCA) in which values may be placed before issuing CICS macros, and from which any returned values can be retrieved after an operation. All values for the DFHBMS TYPE=ROUTE macro are placed in the TCA because they are created at execution time. The TWA is used for storing status information (partly saved in temporary storage across conversations) and space for work. The DFHMSP module is reentrant.

Control blocks

See the list of control blocks in "Basic mapping support" on page 1468.

Modules

DFHMSP (the message switching program) is invoked by the CMSG transaction. DFHMSP's purpose is to route a message entered at the terminal to one or more operator-defined terminals or to other operators.

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

External interfaces

See Figure 68 on page 1748 for external calls made to other areas or domains.

Multiregion operation (MRO)

CICS multiregion operation (MRO) enables CICS regions that are running in the same MVS image, or in the same MVS sysplex, to communicate with each other. MRO does not support communication between a CICS system and a non-CICS system such as IMS.[2]

2. The external CICS interface (EXCI) uses a specialized form of MRO link to support: communication between MVS batch programs and CICS; DCE remote procedure calls to CICS programs.

ACF/Communications Server and SNA networking facilities are not required for MRO. The support within CICS that enables region-to-region communication is called **interregion communication**

The facilities provided by MRO include:

- Transaction routing
- Distributed transaction processing
- Function shipping
- Asynchronous processing
- Distributed program link.

For more information about the design and implementation of interregion communication facilities, see "Interregion communication (IRC)" on page 1722. For descriptions of the facilities provided by MRO, see:

- "Distributed program link" on page 1547
- "Distributed transaction processing" on page 1549
- "Function shipping" on page 1706
- "Transaction routing" on page 1845.

Node abnormal condition program

DFHZNAC is a CICS program used by terminal control to analyze abnormal terminal conditions that are logical unit or node errors detected by z/OS Communications Server. z/OS Communications Server notifies the CICS terminal control program that there is a terminal error, and the terminal control program places the terminal out of service. The terminal control program then invokes DFHZNAC, which writes any error messages to the CSNE transient data destination.

Design overview

The node abnormal condition program (DFHZNAC) can be called for any of several reasons:

- As a central point of control for most z/OS Communications Server-related error situations, error actions can be standardized in table form, allowing for easy addition and alteration to the way conditions are processed.
- Some exception conditions that are not errors are also processed by DFHZNAC, but some exception conditions that are errors are not processed by DFHZNAC.
- It provides a single point of user interface to those who want to change the default actions for an error requiring at most one user program (NEP) - see "Node error program" on page 1754.

To process conditions that are not associated with a known terminal, the dummy TCTTE is used. It is invoked by placing a TCTTE on the system error queue with a 1-byte code relating to the condition. Placing it on the queue makes the TCTTE 'temporary OUTSERV' (TCTTESOS); that is, the decision is pending the outcome of DFHZNAC.

The activate scan routine (DFHZACT) is responsible for attaching the CSNE transaction to run DFHZNAC; this is done during CICS initialization. The CSNE transaction remains in the system until CICS or z/OS Communications Server is

quiesced. If DFHZNAC itself abends, or z/OS Communications Server is closed and then restarted, DFHZACT attaches a new CSNE transaction when there is more work for DFHZNAC to do.

There is only ever one CSNE transaction in the system at any one time. (This should not be confused with the CSNE transaction that is attached by the remote delete processing of autoinstall.)

Once DFHZNAC has been called, it runs down the system error queue, processing each error for each TCTTE on the queue. When there is no more work to be done, DFHZNAC suspends itself, to be resumed by DFHZACT when further processing is required.

Note that the system error queue need not be empty before DFHZNAC terminates; errors can remain on the queue to be processed later.

Node abnormal condition program (NACP) processing involves mapping the error code (placed into the TCTTE by a DFHZERRM macro call) to a set of actions, performing any specific processing for that error code, accumulating the actions for all the error codes in that TCTTE, and then performing the actions.

Figure 69 on page 1752 shows the NACP error code processing. The numbers in Figure 69 on page 1752 refer to the following notes, which use the table entry for DFHZC3424 as the example:

```
DFHZNCM MSGNO=3424,
        E1=S88,
        E2=NULL,
        E3=NULL,
        E4=NULL,
        ACT=(ABSEND,ABRECV,ABTASK,CLSDST,SIMLOG),
        CODE=NSP02,
        TYPE=ENTRY
```

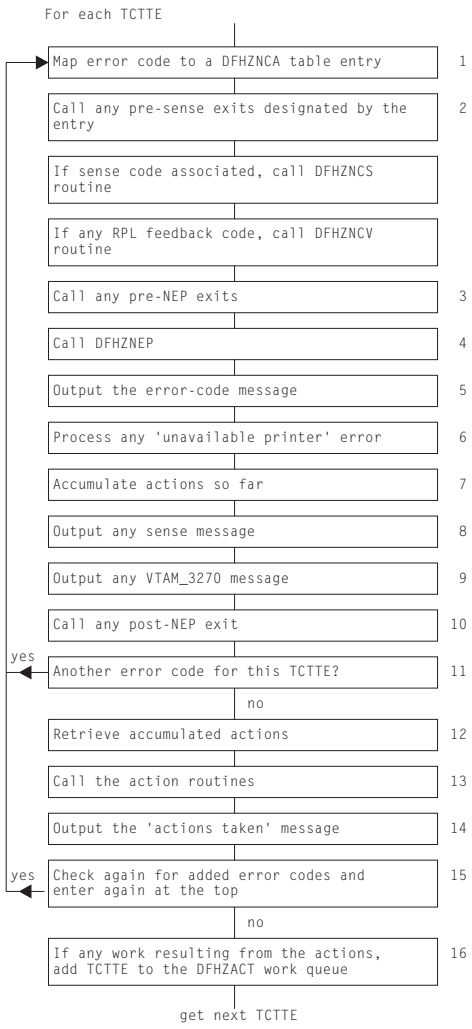


Figure 69. NACP error code processing

Note:

1. The error codes in TCTEVRC* and default actions are defined in the z/OS Communications Server-associated errors section of the documentation.

   In the example, TCTVRC5 contains X'5C', which equates to TCZNSP02 (ref CODE=NSP02).
2. Errors that involve SNA sense have it saved in TCTEVNSS. It is processed by code in copy book DFHZNCS.
3. Call any pre-NEP exits specified by the table entry; for example, E1=S88 references routine NAPES88.
4. Call the node error program (NEP), passing a parameter list via a COMMAREA. This call may or may not change the default actions. The operation of the NEP is described in Writing a node error program and “Node error program” on page 1754.
5. Output error-code message associated with the table entry (DFHZC3424 from MSGNO=3424) to the CSNE log.
6. Check for ‘unavailable printer error'—this caters for a screen copy request that is unable to find an eligible printer if the first choice is unavailable.

7. Because there can be multiple error codes, the actions are accumulated now and performed together later.
8. Output any sense message resulting from the DFHZNCS call, to the CSNE log.
9. Output any z/OS Communications Server_3270 message resulting from the DFHZNCS call (if it was non-SNA) to the CSNE log.
10. Call the post-NEP exit, if any (E4=NULL, no routine).
11. Loop for each error code in TCTEVRC*.
12. When all the error codes for this TCTTE that can be processed at this time have been processed, retrieve the actions that have been accumulated, such as ACT=(ABSEND, ABRECV, ABTASK, CLSDST, SIMLOG).
13. Call the action routine to process each of the actions.
14. Output the 'actions taken' message DFHZC3437 to the CSNE log.
15. Check again for any error codes added asynchronously while the CSNE transaction was running.
16. Queue any work resulting from the actions to the activate scan routine.

Control blocks

DFHZNAC references CSA, its own TCA, JCA, TCT prefix, TIOA, NIB, PCT, SIT, TCTWE, z/OS Communications Server RPL, z/OS Communications Server ACB, and the NACP/NEP communication area. However, the processing mainly concerns access to the TCTTE, and to the NACP/NEP communication area (COMMAREA), which is mapped by the DFHNEPCA DSECT.

See An overview of writing a NEP for a detailed description of the NEP communication area.

Modules

Module	Function
DFHZNAC	Processes the system error queue of TCTTEs and contains the central structure of NACP, outlined in Figure 69 on page 1752. It contains the following copy books:
DFHZNCA	This copy book contains the exit routines for each error code and the error code table itself built by DFHZNCM macros.
DFHZNCE	Links to the user node error program (DFHZNEP) and responds to the action flag settings in the NACP/NEP COMMAREA.
DFHZNCS	Processes the SNA sense codes and contains the sense code tables built using a combination of DFHZMJM and DFHZNCM macros.
DFHZNCV	Contains the z/OS Communications Server return code table.
DFHZNCM	The macro to build the error code table.
DFHZMJM	The macro to build the sense code table.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for the node abnormal condition program, as part of terminal control:

- AP FCxx, for which the trace levels are TC 1, TC 2, and Exc.
- AP FD7E, for which the trace level is TC 1.

Statistics

The only statistical field that DFHZNAC updates is TCTTETE. Because DFHZNAC is the main module for terminal errors, it has primary responsibility for updating the node error count.

Node error program

CICS provides a user-replaceable node error program, DFHZNEP, which assists the user in the following ways:

- It provides a general environment within which it is easy for users to add their own error processors.
- It provides the fundamental error recovery actions for a z/OS Communications Server 3270 network.
- It serves as the default node error program (NEP), where the user selects a NEP at system initialization.

The DFHZNEP program can be one of the following:

- The CICS-supplied default NEP
- A skeleton sample NEP generated using the DFHSNEP macro
- A user-written NEP generated using the DFHSNEP macro.

Design overview

The purpose of the NEP is to allow user-dependent processing whenever a communication system event is reported to CICS. An example of the processing that can be done is to analyze the event and override the default action set by DFHZNAC. When NEP processing is complete, control returns to DFHZNAC.

The default node error program sets the 'print TCTTE' action flag (TWAOTCTE in the user option byte TWAOPT1, defined in DFHNEPCA) if a z/OS Communications Server storage problem has been detected; otherwise, it performs no processing, and leaves the action flags set by DFHZNAC unchanged.

The skeleton sample NEP provided by CICS can provide extended error handling for z/OS Communications Server terminals, and is generated by means of the DFHSNEP macro. This procedure is described in Sample node error program.

The DFHSNEP macro can also be used to generate a user-written NEP. Interactions between the applications and z/OS Communications Server depend on characteristics of the transactions and the installation. Each system has different characteristics. The CICS-provided skeleton NEP is a framework for a user-written NEP to handle network error conditions that may be unique to a particular installation.

Reference information about NEP coding is given in Writing a node error program.

Modules

DFHZNEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided specifically for this function; however, trace entries are made from DFHZNAC immediately before and after calling the node error program.

Point IDs AP FC71 and AP FC72, with a trace level of TC 1, correspond to these trace entries.

Program control

The program control program, DFHPCP, is an interface routine which supports DFHPC LINK, ABEND, SETXIT and RESETXIT calls issued in other CICS modules and invokes the appropriate program manager domain function.

In previous releases DFHPCP provided the functions that are now provided by the Program Manager Domain, and other domains.

Design overview

Services in response to requests

The following services are performed by DFHPCP in response to DFHPC requests from other CICS functions, where those functions have not been converted to use domain interfaces :

Link (LINK)
Builds a parameter list and issues DFHPGLK FUNCTION(LINK) domain call.

Handle Abend (SETXIT)
If SETXIT macro specifies an abend routine address, then DFHPCP builds a parameter list and issues a DFHPGHM FUNCTION(SET_ABEND) OPERATION(HANDLE) call. If SETXIT macro does not specify an abend routine address, then DFHPCP builds a parameter list and issues a DFHPGHM FUNCTION(SET_ABEND) OPERATION(CANCEL) call.

RESETXIT
DFHPCP builds a parameter list and issues a DFHPGHM FUNCTION(SET_ABEND) OPERATION(RESET) call. If SETXIT macro does not specify an abend routine address, then DFHPCP builds a parameter list and issues a DFHPGHM CANCEL call.

Abend (ABEND)
If it is an ABEND request without an existing TACB, then the parameter list is built for this abend. A DFHABAB(CREATE_ABEND_RECORD) is issued to build the TACB. Else a DFHABAB(UPDATE_ABEND_RECORD) is issued with the name of the failing program is issued. A DFHABAB(START_ABEND) call is then made to issue the abend. If the DFHABAB(START_ABEND) call returns control to this module, it is because the exit XPCTA has been invoked and modified the return address. Control is passed to the modified address in the requested execution key.

Modules

DFHEPC

Call mechanism: Branched to from DFHEIP.

Entry address: DFHEPCNA. Stored in the CSA in a field named CSAEPC.

Purpose: DFHEPC is DFHEIP's program control interface. It supports the following EXEC CICS requests

- LINK
- XCTL
- RETURN
- LOAD
- RELEASE
- ABEND
- HANDLE ABEND

It routes a local request to the PG domain, or to DFHABAB (EXEC CICS ABEND) It routes a remote EXEC CICS LINK request to the intersystem module, DFHISP.

Called by: DFHEPC is called exclusively by DFHEIP.

Inputs: The application parameter list.

Outputs: Updated EIB.

Operation:

LINK If SYSID is remote, ships the link request through the DFHISP module.

If SYSID is local:

- Builds parameter list and calls DFHPGLE FUNCTION(LINK_EXEC)
- Checks the response.
- If response indicates the program is remote, ships the link request through the DFHISP module.
- Sets up EIBRESP (and, if needed, EIBRESP2).
- Returns control to DFHEIP.

XCTL Builds parameter list and calls DFHPGXE FUNCTION(PREPARE_XCTL_EXEC)

Checks the response

Sets up EIBRESP (and, if needed, EIBRESP2).

If the PGXE request failed, then returns control to DFHEIP

If the PGXE request was successful, then return control to DFHAPLI as for EXEC CICS RETURN. (DFHAPLI will then invoke the program specified on EXEC CICS XCTL).

RETURN

Builds parameter list and calls DFHPGRE FUNCTION(PREPARE_RETURN_EXEC) (this call is bypassed if there are no options (COMMAREA, TRANSID, INPUTMSG) specified on EXEC CICS RETURN

- Checks the response
- Sets up EIBRESP (and, if needed, EIBRESP2)
- If the PGRE request failed, then returns control to DFHEIP

- If the PGRE request was successful (or was bypassed), then return control to DFHAPLI which completes the return to the calling program or to Transaction Manager

LOAD

Builds parameter list and calls DFHPGLD FUNCTION(LOAD_EXEC)

Checks the response

Sets up EIBRESP (and, if needed, EIBRESP2).

If the PGLD request was successful, then set the return parameters in the application parameter list.

Returns control to DFHEIP.

RELEASE

Builds parameter list and calls DFHPGLD FUNCTION(RELEASE_EXEC)

Checks the response

Sets up EIBRESP (and, if needed, EIBRESP2).

Returns control to DFHEIP.

HANDLE ABEND

For HANDLE ABEND PROGRAM, perform resource security check and check whether program name is known.

Builds parameter list and calls DFHPGHM FUNCTION(SET_ABEND)
- OPERATION(HANDLE) for HANDLE ABEND PROGRAM or LABEL
- OPERATION(CANCEL) for HANDLE ABEND CANCEL
- OPERATION(RESET) for HANDLE ABEND

Checks the response

Sets up EIBRESP (and, if needed, EIBRESP2).

Returns control to DFHEIP.

ABEND

Builds parameter list and calls DFHABAB FUNCTION(CREATE_ABEND_RECORD) and FUNCTION(START_ABEND).

DFHABAB START_ABEND does not normally return, as control is passed to a program or label specified on a HANDLE ABEND, or the program is terminated abnormally.

The XPCTA user exit can request retry. In this case DFHABAB START_ABEND returns to DFHEPC passing back the retry parameters. DFHEPC sets the registers and other values and branches to the specified retry address.

How loaded: At CICS startup, as part of the building of the CICS nucleus. The nucleus is built by DFHSIB1, which uses its nucleus build list to determine the content and characteristics of the CICS nucleus.

Exits

Program	Global user exit points
DFHEPC	XPCERES XPCREQ XPCREQC
DFHABAB	XPCABND XPCTA

Program	Global user exit points
DFHAPLI1	XPCFTCH XPCHAIR
DFHERM	XPCHAIR
DFHUEH	XPCHAIR

Trace

The following point IDs are provided for entry to and exit from DFHPCPG:

- AP 2000, for which the trace level is PC 1
- AP 2001, for which the trace level is PC 1.

Program error program

CICS provides a dummy program error program (DFHPEP) that does nothing except give control back to the abnormal condition program (DFHACP), which is invoked during transaction abend processing.

You can provide some additional routines to handle programming errors. For instance, it is possible to disable the transaction code associated with the program in error, thus preventing the recurrence of the error until it can be corrected; send messages to the end-user terminal; initiate a new transaction; or record abend information in transient data.

Design overview

To provide corrective action in response to a programming error, you can code a program error program (DFHPEP). This program can then be assembled and link-edited to replace the dummy DFHPEP.

If provided, this program is invoked by the abnormal condition program (DFHACP) whenever a task terminates due to a task abnormal condition. However, it will **NOT** be called if a task is terminated due to an attach failure (for example the transaction is not defined) or when CICS deliberately terminates a task to alleviate a stall.

The user can perform any kind of corrective action within a program error program.

Guidance information about PEP coding is given in the Troubleshooting for recovery processing.

Control blocks

The control block associated with the program error program is:
DFHPEP_COMMAREA, the commarea passed to DFHPEP.

See Data areas for a detailed description of this control block.

Modules

DFHPEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

Program preparation utilities

The program preparation utilities consist of the command-language translators, which are utility programs that run offline to translate CICS application programs that use command-level CICS requests.

The translators convert the EXEC CICS commands into call statements in the language in which the EXEC commands are embedded. Versions of the translator program are available for the following languages:

- COBOL (DFHECP1$)
- PL/I (DFHEPP1$)
- C (DFHEDP1$)
- Assembler language (DFHEAP1$).

Design overview

The command-language translators manage storage by creating a stack from a single area allocated at the start of the program.

Because the input is free-format, the translators move it into a buffer area that can hold data spanning two or more source records. The analysis of the source is mainly table driven.

The translators build the replacement source code for each EXEC command in a form appropriate to the language:

- For COBOL, the replacement code contains a series of MOVE statements, followed by a CALL statement.
- For PL/I, the replacement code contains a declaration of an entry variable followed by a CALL statement. These statements are contained within a DO group.
- For C, the replacement code contains a function call (dfhexec) and may also contain assignment statements.
- For assembler language, the replacement code is an invocation of the DFHECALL macro.

Errors in the source can be detected. Spelling corrections are made to the source, and any unrecognizable or duplicate keywords and options are ignored. For COBOL, PL/I, and C, the translator produces error diagnostics that are collected together on the output listing. The assembler language translator, however, produces error diagnostics in the translated output following the EXEC command in which the error occurred.

Modules

DFHECP1$, DFHEPP1$, DFHEDP1$, DFHEAP1$

Exits

Global user exit points are not applicable to offline utilities.

Trace

Trace points are not applicable to offline utilities.

Remote DL/I

An overall description of DL/I database support is given in "DL/I database support" on page 1559. This section gives information that is specific to remote DL/I.

Design overview

This section outlines what you must do to define remote DL/I support, and describes the functions of remote DL/I.

System definition

For a CICS system that supports only remote databases you must, in addition to providing the usual definitions that are required for function shipping, code a PSB directory (PDIR) using the DFHDLPSB macro. Every PDIR entry must have SYSIDNT specified. The PDIR system initialization parameter must be coded specifying the suffix of the PDIR.

DL/I PSB scheduling

When a CICS task requests the scheduling of a DL/I PSB by means of an EXEC DLI SCHEDULE request or DL/I PCB call, and the request is for a remote PSB, control is passed to DFHDLIRP. DFHDLIRP allocates a remote scheduling block (RSB) and issues a DFHIS TYPE=CONVERSE macro to ship the scheduling request to the owning system.

Database calls

For a remote DL/I database call, a DFHIS TYPE=CONVERSE macro is issued to ship the request to the owning system. The return codes are passed back to the user in the user interface block (UIB).

DL/I PSB termination

If a remote PSB is terminated, the actions performed are:

1. Free the RSB and local program communication block (PCB) storage.
2. If the DL/I PSB termination was not caused by a CICS syncpoint, request one now.

Control blocks

Figure 70 on page 1761 illustrates some of the control blocks used to support remote DL/I.

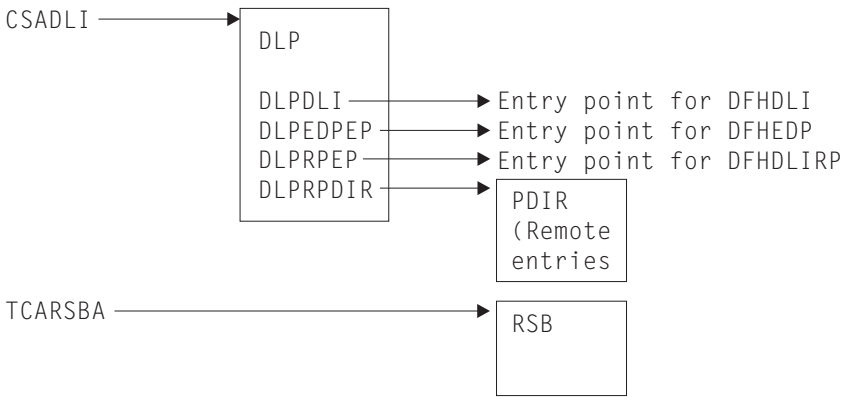


Figure 70. Some control blocks used for remote DL/I support

The DL/I interface parameter list (DLP) is described in "DL/I interface parameter list (DLP)" on page 1561.

The remote PSB directory (PDIR) contains an entry for each remote PSB that can be used from an application program.

The remote scheduling block (RSB) is acquired when a CICS task issues a PSB schedule request for a remote PSB. The RSB is freed when the task issues a SYNCPOINT or a DLI TERM request.

Resource definition online (RDO)

The CEDA transaction creates and alters the definitions of system resources in the CICS system definition (CSD) data set.

RDO provides:

- Online transactions that can be used to **inspect**, **change**, and **install** resource definitions:
  - CEDA (inspect, change, and install)
  - CEDB (inspect and change)
  - CEDC (inspect only).
- A programmable interface to the CEDA transaction, using an EXEC CICS LINK command in the application program to invoke DFHEDAP directly. (For further information, see Using the programmable interface to CEDA.)
- A set of system programmer API command (the EXEC CICS CREATE commands) for creating CICS resources dynamically.
- An offline utility, DFHCSDUP, to inspect or change resource definitions. (For a description of this utility, see "CSD utility program (DFHCSDUP)" on page 1533.)

Design overview

Resource definitions are maintained on the CICS system definition (CSD) data set. The resource definitions in the CSD data set can be viewed and changed using either the online CEDx transactions, or the offline utility DFHCSDUP.

Installation of resource definitions makes the definitions available to the running CICS system. Resource definitions can be installed at these times:

- When a cold start of CICS is performed, using the GRPLIST system initialization parameter.
- During a run of CICS, using the CEDA transaction.

When resource definitions are installed, they are made available through the appropriate resource managers.

Modules

The relationships between the components of RDO, and the components of some of the services it uses, are shown in Figure 71.

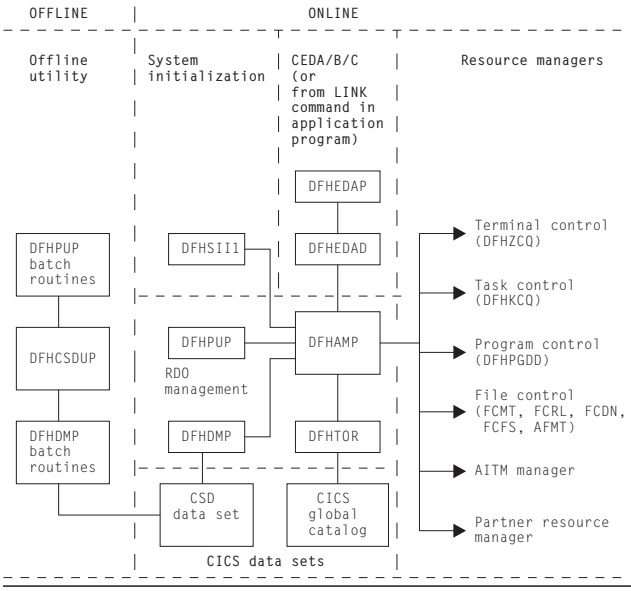


Figure 71. RDO interfaces

DFHEDAP and **DFHEDAD** control the CEDA, CEDB, and CEDC transactions. They provide screen management for the transactions, and invoke DFHAMP to implement any actions that are required.

DFHSII1 invokes DFHAMP when CICS is cold started, to install resource definitions for the current run. These resource definitions are specified by the GRPLIST system initialization parameter. DFHSII1 passes the GRPLIST system initialization parameter to DFHAMP.

DFHAMP, the allocation management program, manages all requests to view, change, and install resources. It uses the services provided by other parts of RDO, and by the resource managers:

- DFHAMP invokes **DFHPUP** and **DFHDMP** to read, write, and update resource definitions on the CSD data set:
  - DFHPUP, the parameter utility program, converts resource definition data between the parameter list format provided by DFHAMP and the record format needed by the CSD.
  - DFHDMP, the CSD management program, manages I/O of resource definition data to and from the CSD data set.
- DFHAMP invokes **DFHTOR**, the terminal object resolution program, to merge TERMINAL, TYPETERM, CONNECTION, and SESSION definitions:

  - When requests are made to install TERMINALs, TYPETERMs, CONNECTIONs, and SESSIONs, DFHTOR merges TYPETERM and TERMINAL information, and also CONNECTION and SESSION information, and passes this merged information back to DFHAMP.
  - DFHAMP passes the merged definitions to DFHZCQ to install in the running CICS system. Any merged TERMINAL definitions that are to be used as autoinstall terminal models are passed to the autoinstall terminal model (AITM) manager.
  - When TYPETERM definitions are installed, DFHTOR records the information about the CICS global catalog for subsequent use.
  - When the CHECK command is issued, DFHTOR checks the appropriate TERMINAL, TYPETERM, CONNECTION, and SESSION definitions for consistency.
- DFHAMP calls the appropriate resource managers to install resources in the running CICS system:
  - **DFHZCQ** is invoked to install CONNECTION, SESSION, and TERMINAL definitions.
  - **DFHAMXM** is invoked to install TRANSACTION and PROFILE definitions.
  - **DFHPGDD** is invoked to install PROGRAM, MAPSET, and PARTITIONSET definitions.
  - These subroutine "gates" are called to install resources related to file control:
    - **FCMT**, for FCT entries
    - **FCRL**, for LSR pools
    - **FCDN**, for DSN blocks
    - **FCFS**, to open and close files
    - **AFMT**, for AFCT entries for files.
  - The **AITM manager** is invoked, using an AITM ADD_REPL_TERM_MODEL subroutine call (see "Autoinstall terminal model manager" on page 1463), to install autoinstall terminal models.
  - The **partner resource manager** is invoked to install partner resources for the SAA communications interface.

DFHEICRE processes all the EXEC CICS CREATE commands. It builds an internal DEFINE command for the resource to be created, and passes it to DFHCAP for interpretation. The encoded command is then passed directly to DFHAMP to install the resource in the running system. The CSD file is not accessed at all during this processing.

DFHCSDUP, the offline CICS system definition utility program, uses batch versions of routines from DFHPUP and DFHDMP to read, write, and update resource definitions on the CSD data set (see "CSD utility program (DFHCSDUP)" on page 1533).

For a detailed description of how the CEDA transaction handles terminal resources, see "Terminal control" on page 1814.

Exits

The XRSINDI global user exit is invoked at each install or EXEC CICS CREATE.

Trace

The following point IDs are provided, with a trace level of AP 1:

- AP 00EB (DFHAMP)
- AP 00EC (DFHDMP)
- AP 00EF (DFHTOR)
- AP 00E2 (DFHPUP).

SAA Communications and Resource Recovery interfaces

This section describes the CICS implementation of the Communications and Resource Recovery elements of the Systems Application Architecture® Common Programming Interface (also known as the SAA Communications and SAA Resource Recovery interfaces respectively).

The SAA Communications and Resource Recovery interfaces are both call-based application programming interfaces that are common across all programming languages and across hardware systems.

The common programming interface (CPI) component of CICS, also sometimes known as the CP component, provides application programming interfaces that conform to SAA specifications for Communications and Resource Recovery interfaces.

Note: This CICS component does **not** currently handle any other SAA interface elements.

The CPI component is part of the AP domain, and is shipped as object code only (OCO).

The **SAA Communications interface** allows CICS applications to communicate via APPC (LU6.2) links to partner applications on any system that conforms to SAA standards. This interface consists of a set of defined verbs as program calls that are adapted for the language being used. For further information about the general call-based API, see the *SAA CPI Communications Reference* manual, SC26-4399.

The SAA Communications interface in CICS provides an alternative to the existing application interface for distributed transaction processing (see "Distributed transaction processing" on page 1549). A single transaction can use EXEC CICS commands for one conversation while using SAA Communications calls for another (separate) conversation. Also, one end of a conversation can use EXEC CICS commands while the other end uses SAA Communications calls. However, it is not possible to use a mixture of EXEC CICS commands and SAA Communications calls on the same end of a conversation.

The **SAA Resource Recovery interface** provides an SAA application programming interface for commit and backout of recoverable resources. This interface consists of two defined verbs as program calls that are adapted for the language being used:

SRRCMIT
: Commit

SRRBACK
: Backout

For further information, see the *SAA CPI Resource Recovery Reference* manual, SC31-6821.

The SAA Resource Recovery interface in CICS provides an alternative to the use of EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK commands. The SRRCMIT call is equivalent to the EXEC CICS SYNCPOINT command, and the SRRBACK call is equivalent to the EXEC CICS SYNCPOINT ROLLBACK command. A single application can use SAA Resource Recovery calls, EXEC CICS commands, or a mixture of both.

Design overview

This section describes the SAA Communications and Resource Recovery interfaces.

The SAA Communications interface

When an application issues an SAA Communications call, control passes via the DFHCPLC application link-edit stub to the common programming interface program (DFHCPI), which in turn passes the request to the DFHCPIC program load module. DFHCPIC verifies the parameters, checks the conversation state, and (if required) issues a DFHLUC macro call to invoke the LU6.2 application request logic module (DFHZARL). For details of DFHZARL, see "Distributed transaction processing" on page 1549.

Figure 72 shows how the SAA Communications interface support relates to CICS intersystem communication (ISC) using z/OS Communications Server LU6.2. The numbers in Figure 72 refer to the notes that follow it. CMxxxx represents a program call defined in the SAA Communications interface.

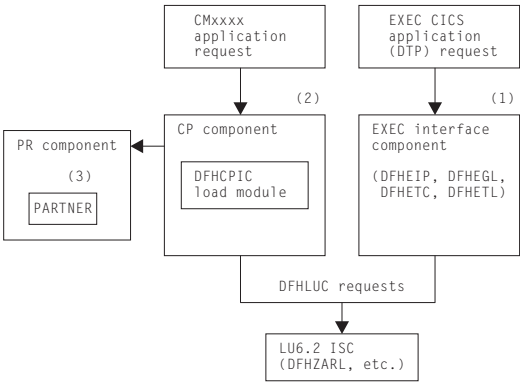


Figure 72. SAA Communications application request processing

Note:

1. Distributed transaction processing (DTP) allows a transaction using EXEC CICS commands to communicate with a transaction running in another system. This is carried out by DFHEIP and related EXEC interface processor modules. For a z/OS Communications Server LU6.2 intersystem link, each request is converted into DFHLUC macro requests that call DFHZARL.
2. The SAA Communications interface is implemented by the DFHCPIC load module within the CP (or CPI) component. DFHCPIC maps the CMxxxx application requests into DFHLUC macro calls.
3. To begin a conversation, the SAA Communications interface requires specific information (side information) about the partner program, including its name

and system details. This is implemented within CICS as an RDO object called the PARTNER, which is encapsulated by the partner resource manager (PR) component.

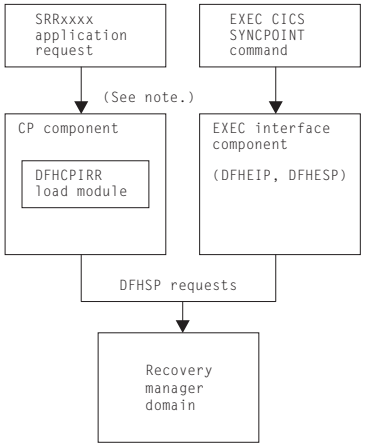
Using the SAA Communications interface on recoverable conversations: When using the SAA Communications interface on recoverable conversations (that is, conversations with the synclevel set to CM_SYNC_POINT), DFHLUC syncpoint requests are routed to DFHZARL via the SAA Communications interface syncpoint request handler (DFHCPSRH) in the DFHCPIC load module. This allows the conversation state to be tracked.

For the equivalent EXEC CICS synclevel 2 conversations, DFHLUC syncpoint requests pass directly to DFHZARL.

The SAA Resource Recovery interface

When an application issues an SAA Resource Recovery call, control passes via the DFHCPLRR application link-edit stub to the common programming interface program (DFHCPI), which in turn passes the request to the DFHCPIRR program load module. DFHCPIRR verifies the parameters, and (if required) issues an appropriate DFHSP macro call: DFHSP TYPE=USER for SRRCMIT, or DFHSP TYPE=ROLLBACK for SRRBACK.

Figure 73 shows how the SAA Resource Recovery interface support relates to the processing of EXEC CICS SYNCPOINT commands. The number in the figure refers to the accompanying note. SRRxxxx represents a program call defined in the SAA Resource Recovery interface, namely, SRRBACK or SRRCMIT.



Note: The SAA Resource Recovery interface is implemented by the DFHCPIRR load module within the CP (or CPI) component. DFHCPIRR maps SRRxxxx application requests into DFHSP macro calls.

Figure 73. SAA Resource Recovery application request processing

Functions provided by the CPI component

Table 87 on page 1767 summarizes the external subroutine interfaces provided by the CPI component. It shows the subroutine call formats, the level-1 trace point IDs of the modules providing the functions for these formats, and the functions provided.

Table 87. CPI component's subroutine interfaces

Format	Trace	Function
CPIN	AP 0C01 AP 0C02	START_INIT COMPLETE_INIT
CPSP	AP 0CD0 AP 0CD1	SYNCPOINT_REQUEST

CPIN format, START_INIT function

The START_INIT function of the CPIN format is used to attach a CICS task to perform initialization of the CPI component.

Input parameters: None.

Output parameters:
RESPONSE
is the subroutine's response to the call. It can have any of these values:
OK|DISASTER|KERNERROR
[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

RESPONSE	Possible REASON values
DISASTER	GETMAIN_FAILED ADD_SUSPEND_FAILED

CPIN format, COMPLETE_INIT function

The COMPLETE_INIT function of the CPIN format is used to wait for the initialization task attached by the START_INIT function to complete processing.

Input parameters: None.

Output parameters:
RESPONSE
is the subroutine's response to the call. It can have any of these values:
OK|DISASTER|KERNERROR
[REASON]
is returned when RESPONSE is DISASTER. It has this value:
INIT_TASK_FAILED

CPSP format, SYNCPOINT_REQUEST function

The SYNCPOINT_REQUEST function of the CPSP format is used to send LU6.2 syncpoint flows on recoverable conversations using the SAA Communications interface, and to update the conversation state as required.

Input parameters:
CPC_ADDRESS
is the address of the SAA Communications conversation control block (CPC).
LUC_ADDRESS
is the address of the DFHLUC parameter list.

Output parameters:
RESPONSE
is the subroutine's response to the call. It can have either of these values:
OK|KERNERROR

Modules

Module	Function
DFHAPTRF	Trace interpreter for the SAA Communications and Resource Recovery interfaces
DFHCPARH	SAA Communications application request handler (entry processor for all application calls to the DFHCPIC load module, routing them to the appropriate DFHCPCxx module)
DFHCPCxx	Components of the DFHCPIC load module, each object module typically handling a different CMxxxx application request
DFHCPDUF	Offline system dump formatter for CP keyword
DFHCPI	Common programming interface program (link-edited with DFHEIP and DFHAICBP to form the DFHAIP load module)
DFHCPIN1	Initialization management program for the SAA Communications and Resource Recovery interfaces
DFHCPIN2	Runs as a CICS task to perform initialization for the SAA Communications and Resource Recovery interfaces
DFHCPIR	SAA Resource Recovery entry processor, handling all calls to the DFHCPIRR load module
DFHCPLC	Link-edit stub for applications using the SAA Communications interface
DFHCPLRR	Link-edit stub for applications using the SAA Resource Recovery interface
DFHCPSRH	SAA Communications syncpoint request handler (part of the DFHCPIC load module)

Exits

No global user exit points are provided for this component.

Trace

The following point ID is provided for this component:

- AP 0Cxx, for which the trace levels are CP 1, CP 2, and Exc.

Statistics utility program (DFHSTUP)

This section provides a general overview of the collection of CICS statistics as well as describing the operation of the offline statistics utility program (DFHSTUP). For more information about using the DFHSTUP utility program, see Statistics utility program (DFHSTUP).

An operator interface to all online statistics functions is provided by the CEMT transaction. The equivalent programmable interface is provided by the EXEC API.

Statistics may be collected at user-specified intervals from the startup to the shutdown of a CICS system. Statistics may also be requested, resulting in the collection of data for the period between the last time statistics were reset and the time the request was made.

Statistics are also collected at system quiesce or logical end of day; this data is written to the SMF data set as for a normal interval collection.

An option is provided by the statistics domain to allow the user to specify whether interval statistics are to be collected. The statistics domain calls each domain in turn to reset the statistics fields at every interval when statistics are collected. Statistics (particularly interval statistics) can be used for capacity planning and performance tuning. For further information, see CICS statistics in Monitoring.

There is a great similarity between CICS statistics data and CICS performance class monitoring data. Statistics data is collected on a resource basis, whereas performance class monitoring collects similar data on a transaction basis. Statistics can therefore be viewed as resource monitoring.

Design overview

CICS statistics support is divided into the following components:

- The operator interface. This component is responsible for interfacing to the various CICS-supported terminals, analyzing the input string and then invoking the statistics domain to perform the appropriate management operation. This function is provided by the CEMT transaction, and also by the EXEC API.
- The statistics domain. This component is responsible for managing statistics interfaces, for example, SMF and EXEC API.
- The statistics update logic. This code is inline in the relevant resource manager. In this way, the control function of statistics is centralized, but the management and updating of the statistics fields is given to the resource owner.
- The statistics data collection and reset. For all collection types except unsolicited, the collection mechanism is the same. The owning domain is invoked by statistics domain to supply a record that contains the domain's statistics. When this record has been formed, the domain then calls statistics domain to place the data on the SMF data set.

  There are five types of collections:

  – Interval. The collection interval default is 1 hour. A user can change this value. The minimum value is 1 minute, the maximum 24 hours. On an interval collection, each called domain collects and resets its statistics counters. No action is taken if the statistics recording status is OFF.

  – Requested. Statistics can be requested using the PERFORM STATISTICS function provided by the CEMT transaction or the EXEC API. The data recorded is for the period between the last time statistics were reset and the time the request was made. Statistics are reset on an interval, end-of-day, or requested-reset collection; they can also be reset, without a collection, when changing the statistics recording status from ON to OFF, or from OFF to ON.

    This type of collection can obtain statistics from some or all domains, as requested. Each called domain collects, but does not reset, its statistics counters.

    Requested statistics are collected even if the statistics recording status is OFF.

  – Requested-reset. This collection is similar to requested statistics, except that it always obtains statistics for all domains, and each called domain resets its statistics counters after collection.

    Requested-reset statistics are collected even if the statistics recording status is OFF.

  – End-of-day. This collection occurs when the system is quiescing. A logical end-of-day time may be specified. The default time is midnight. This is primarily for continually running systems. The collection is then made at this time, and the called domain collects and resets its statistics counters.

    End-of-day statistics are collected even if the statistics recording status is OFF.

Daily systems that are taken down after midnight should change the logical end of day to a time when the system is not operational.

If the user wants to simulate shutdown statistics, the interval can be set to 24 hours. An end-of-day report, which contains total figures for the CICS run up to the end of the day, can then be printed by DFHSTUP.

– Unsolicited. For dynamically allocated and deallocated resources, the resource records its statistics just before it is deleted; for example, an autoinstall terminal that logs off and is thereby deleted. USS statistics are written to SMF regardless of the statistics recording status (STATRCD).

By default, DFHSTUP formats the statistics for all types of collection, for all the specified APPLIDs. However, if you specify the EXTRACT control parameter but not COLLECTION TYPE, only the extract exit is invoked and no other statistics output is produced.

- The statistics formatting control. The offline utility DFHSTUP opens the statistics data set, which is an unloaded SMF data set, and the I/O interfaces to that data set. This routine then browses the data set and formats the statistics.

  Reports may be produced for any or all of the five types of statistics collections.

  DFHSTUP also provides the option of producing a summary report for selected CICS applids. The summary report is constructed from all the statistics contained in the interval, requested-reset, end-of-day, and unsolicited collections. Requested statistics are not involved in the production of the summary report.
- The extract statistics reporting function. This is a DFHSTUP exit that takes statistics data from the input SMF data set and passes it to a user program for processing in order to create tailored statistics reports. DFH0STXR is a sample program designed to exploit the extract reporting function. There are also two skeleton exits; an assembler extract exit called DFH£STXA, and a COBOL extract exit called DFH0STXC. These show the format and structure of the interface between DFHSTUP and the extract exit.

  Specifying the extract statistics reporting function changes the default DFHSTUP report settings. If you specify only the EXTRACT control statement, only the extract exit is driven; other DFHSTUP reports are suppressed. If EXTRACT is specified, other statistics report control statements, such as SUMMARY, must also be specified to ensure that the appropriate reports are produced.

DFHSTUP operation

DFHSTUP runs as a separate MVS job and extracts all or selected entries from the unloaded SMF data set. The types of entries to be processed by this program are specified in the SYSIN data set. Entries that can be selected for processing include:

- All entries—the default
- Entries written for specified applids
- Entries written for specified resource types
- Entries written for specified collection types, that is, interval, requested, requested-reset, end-of-day, or unsolicited
- Entries written during a specified period of time.

You can also select:

- The page size; the default is 60 lines per page.
- Whether output is to be printed in mixed case or all uppercase; the default is to print in mixed case.
- The summary report option; by default, it is not selected.

Further information about using DFHSTUP is given in Statistics utility program (DFHSTUP).

Modules

Module	Function
DFH£STXA	Skeleton assembler extract exit
DFH0STXC	Skeleton COBOL extract exit
DFH0STXR	DFHSTUP extract sample program
DFHST03X	z/OS Communications Server statistics summary formatter
DFHST04X	Autoinstall terminals statistics summary formatter
DFHST06X	Terminal statistics summary formatter
DFHST08X	LSRPOOL resource statistics summary formatter
DFHST09X	LSRPOOL file statistics summary formatter
DFHST14X	ISC/IRC statistics summary formatter
DFHST16X	Table manager statistics summary formatter
DFHST17X	File control statistics summary formatter
DFHST21X	ISC/IRC attach-time statistics summary formatter
DFHST22X	FEPI statistics summary formatter
DFHSTD2X	CICS DB2 statistics summary formatter
DFHSTDBX	DBCTL statistics summary formatter
DFHSTDSX	Dispatcher domain statistics summary formatter
DFHSTDUX	Dump domain statistics summary formatter
DFHSTE15	DFSORT interface to E15 user exit
DFHSTE35	DFSORT interface to E35 user exit
DFHSTEJX	Enterprise Java domain statistics summary formatter
DFHSTIIX	IIOP domain statistics summary formatter
DFHSTIN	DFSORT E15 user exit input routine
DFHSTISX	IPCONN statistics summary formatter
DFHSTLDX	Loader domain statistics summary formatter
DFHSTLGX	Log manager domain summary statistics formatter
DFHSTMNX	Monitoring domain statistics summary formatter
DFHSTMQX	CICS-MQ statistics summary formatter
DFHSTOT	DFSORT E35 user exit output routine
DFHSTPGX	Program manager domain statistics summary formatter
DFHSTRD	Read interface subroutines
DFHSTRMX	Recovery manager domain statistics summary formatter
DFHSTSJX	JVM domain statistics summary formatter
DFHSTSMX	Storage manager domain statistics summary formatter
DFHSTSOX	Sockets domain statistics summary formatter
DFHSTSTX	Statistics domain statistics summary formatter
DFHSTTQX	Transient data statistics summary formatter
DFHSTTSX	Temporary storage domain statistics summary formatter
DFHSTU03	z/OS Communications Server statistics formatter
DFHSTU04	Autoinstall terminals statistics formatter
DFHSTU06	Terminal statistics formatter
DFHSTU08	LSRPOOL resource statistics formatter
DFHSTU09	LSRPOOL file statistics formatter
DFHSTU14	ISC/IRC statistics formatter
DFHSTU16	Table manager statistics formatter
DFHSTU17	File control statistics formatter
DFHSTU21	ISC/IRC attach-time statistics formatter
DFHSTU22	FEPI statistics formatter
DFHSTUD2	CICS DB2 statistics formatter
DFHSTUDB	DBCTL statistics formatter

Module	Function
DFHSTUDS	Dispatcher domain statistics formatter
DFHSTUDU	Dump domain statistics formatter
DFHSTUEJ	Enterprise Java domain statistics formatter
DFHSTUII	IIOP domain statistics formatter
DFHSTUIS	IPCONN statistics formatter
DFHSTULD	Loader domain statistics formatter
DFHSTULG	Log manager domain statistics formatter
DFHSTUMN	Monitoring domain statistics formatter
DFHSTUMQ	CICS-MQ statistics formatter
DFHSTUP1	PRE_INITIALIZE
DFHSTUPG	Program manager domain statistics formatter
DFHSTURM	Recovery manager domain statistics formatter
DFHSTURS	User domain statistics formatter
DFHSTURX	User domain statistics summary formatter
DFHSTUSJ	JVM domain statistics formatter
DFHSTUSM	Storage manager domain statistics formatter
DFHSTUSO	Sockets domain statistics formatter
DFHSTUTQ	Transient data statistics formatter
DFHSTUST	Statistics domain statistics formatter
DFHSTUTS	Temporary storage domain statistics formatter
DFHSTUXM	Transaction manager domain statistics formatter
DFHSTWR	Write interface subroutines
DFHSTXMX	Transaction manager domain statistics summary formatter

Storage control macro-compatibility interface

DFHSMSCP is responsible for handling all requests for storage services that are made by using the routine addressed by CSASCNAC in the CICS common system area (CSA). DFHSMSCP is called by some parts of the CICS AP domain containing DFHSC macros.

DFHSMSCP converts all requests into calls to the storage manager domain, and its main function is to get or free storage.

Design overview

The input to DFHSMSCP, set up by the macro used for the invocation, or directly by the calling program, consists of the following TCA fields:

- TCASCTR—the storage control request byte. This can contain one of the following values:
  - X'80' GETMAIN, in conjunction with:
    - X'40' Initialize storage
    - X'20' Conditional
    - Storage class in bits 3 through 7 (the resulting SMMC GETMAIN storage class name is given in parentheses where this differs from the first name):
      - X'00' 1WD, treated as SHARED
      - X'04' LINE
      - X'05' TERMINAL or TERM
      - X'0C' USER (becomes CICS24)
      - X'0D' TRANSDATA or TD
      - X'13' SHARED (becomes SHARED_CICS24)

        - X'14' CONTROL
    - X'40' FREEMAIN, in conjunction with:
        - X'01' TCTTE address supplied.
- TCASCIB—the 1-byte value to which storage is to be initialized.
- TCASCNB—the 2-byte field giving the number of bytes requested on the GETMAIN.
- TCASCSA—the 4-byte address of the storage that was obtained or the storage to be freed.

Modules

DFHSMSCP

Exits

No global user exit points are provided for this function.

Trace

The point IDs for this function are of the form AP F1xx; the corresponding trace levels are SC 1 and Exc.

Subsystem interface

The subsystem interface is a mechanism by which the MVS operating system communicates with its underlying subsystems at certain critical points in its processing.

CICS is required to be defined as a formal MVS subsystem for the following purposes:

- Multiregion operation (MRO)
- Shared database support
- Console message handling.

Functional overview

An MVS subsystem consists of two control blocks and a set of functional routines, all resident in common memory. The control blocks are:

SSCT The subsystem communication table, which contains the 4-character name of the subsystem and a pointer to the SSVT.

SSVT The subsystem vector table, which contains a list of the subsystem function codes that the subsystem supports, and the addresses of the functional routines that support them.

The subsystem is **active** when the SSCT contains a nonzero pointer to the SSVT, and **inactive** when the pointer is zero.

Subsystem definition

Each subsystem is defined to MVS by an entry in an IEFSSNxx member of SYS1.PARMLIB. (See the *z/OS MVS Initialization and Tuning Guide*.) Each subsystem can be defined with an initialization routine and some initialization parameters. The CICS subsystem is defined with an initialization routine of DFHSSIN, and an initialization parameter that specifies the name of an additional member of SYS1.PARMLIB, which contains further CICS-specific subsystem parameters. These parameters specify whether the console message handling facility is required.

Design overview

When the recommended initialization routine DFHSSIN is specified, the CICS subsystem is initialized during the master scheduler initialization phase of the MVS IPL. The CICS-specific subsystem parameters are read from SYS1.PARMLIB, and the subsystem vector table is created. The supporting subsystem function routines are loaded into common memory and their addresses are stored into the subsystem vector table. If everything is successful, the CICS subsystem is made active by storing the address of the subsystem vector table in the subsystem communication table.

Console message handling

At startup, a CICS region that supports console message handling notifies the CICS subsystem of its existence, by using the CICS SVC to issue a subsystem interface call for the 'generic connect' function with a CONNECT subfunction. The subsystem notes the creation of the new region and, if this is the first such CICS region to become active, invokes a service of MVS console support called "subsystem console message broadcasting". The message broadcasting service causes all subsequent console messages to be broadcast to all subsystems that have expressed an interest in receiving them, including the CICS subsystem. This MVS service can also be activated by other products, for example, NetView®.

If the message broadcasting service has been activated, either by CICS or by another product, the CICS subsystem examines ***all*** messages issued by WTO macros in any address space, but it intercepts and modifies only the following:

- Messages beginning with "DFH" that are issued under any CICS TCB, including those CICS regions that do not have console message handling support.

  These messages are reformatted to contain the CICS applid for the region in a standard position in the message.

  Because the CICS subsystem receives control after JES has recorded a console message in the job's message log, messages in the job log do not appear to be reformatted. The messages are only reformatted on the operator consoles and in the MVS system log.

  If the original message is a long one, inserting the CICS applid can cause the message to exceed the maximum length for an MVS console message. In this case, the original message is suppressed (that is, does not appear on the console), and the reformatted message is issued using the MVS multiple-line console message service to split the message text over several lines. Both the original message and perhaps several instances of the reformatted multiple-line message appear in the job log, but only one copy of the reformatted message is displayed on the console.
- Messages that redisplay, on operator consoles or in the MVS system log, MODIFY commands that are directed towards CICS and contain signon passwords for the CESN transaction.

  These messages are reformatted with the passwords replaced by asterisks, so that the original passwords are not exposed.

As each TCB terminates, it issues an 'end of task' subsystem call, which is broadcast to all active subsystems. Likewise, as each address space terminates, it issues an 'end of memory' subsystem call, which is also broadcast to all active subsystems. When it receives either of these calls, the CICS subsystem first calls the end-of-memory routine in DFHIRP; then, if the terminating address space is known by the subsystem, it invokes the 'generic connect' function with a DISCONNECT subfunction.

The DISCONNECT subfunction notes the termination of the CICS address space and, if this is the last CICS containing console message handling support to terminate, notifies the "subsystem console message broadcasting" support that the CICS subsystem is no longer interested in receiving broadcast console messages. Nevertheless, if another product has kept console message broadcasting active, the CICS subsystem continues to reformat messages from CICS regions that do not have console message handling support.

Control Blocks

DSECT	Function
DFHSABDS	The CICS subsystem anchor block (SAB). This is used to contain global subsystem-related information that is common to all CICS regions in the MVS image. It is used to record the options specified in the DFHSSInn member of SYS1.PARMLIB. It contains a pointer to a bit map that records which MVS address spaces contain an active CICS. It also contains the address of the subsystem control table extension (SCTE) used by IRC, and the address of the MVS status tracking information used by XRF.
IEFJSCVT	The subsystem communication table (SSCT). This is an MVS control block. There is one SSCT for each subsystem, including the primary job entry subsystem (JES) as well as CICS.
IEFJSSVT	The subsystem vector table (SSVT). This is an MVS control block. There is one SSVT for each active subsystem. It contains a lookup table for determining which function codes are supported by the subsystem, and a list of the entry points for all the supporting function routines.

Figure 74 on page 1776 shows these control blocks.

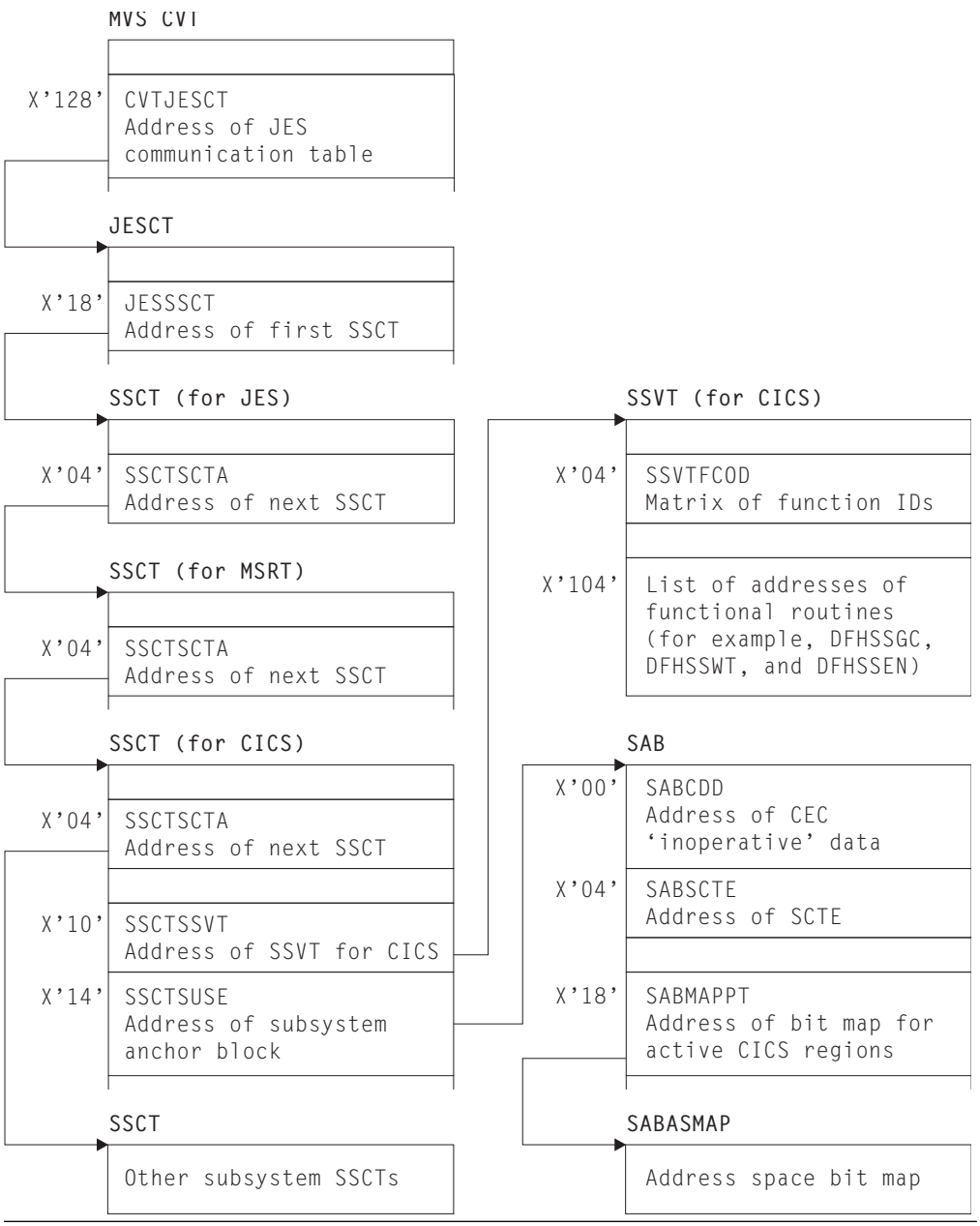


Figure 74. Control blocks associated with the subsystem interface

Modules

Module	Function
DFHSSIN	Subsystem initialization routine for the CICS subsystem. Reads in subsystem parameters from member DFHSSInn of SYS1.PARMLIB, creates SSVT, loads function modules into MVS common storage.
DFHSSEN	End-of-task and end-of-memory functional module. Calls DFHIRP's EOT/EOM routine. Issues 'generic connect' if terminating region or job-step task is in the CICS address space map.

Module	Function
DFHSSGC	The generic connect functional module. CONNECT subfunction sets the bit for the current address space in the address space map. If this is the first CICS region to start, it invokes IEAVG700 to initiate message broadcasting. DISCONNECT subfunction unsets the bit for the current address space in the address space map. If this is the last CICS region to finish, it invokes IEAVG700 to terminate message broadcasting.
DFHSSMGP	Message routine for DFHSSIN.
DFHSSMGT	Message table for DFHSSIN.
DFHSSWT	Router module for the console message handler. Calls DFHSSWTO for messages beginning with DFH. Calls DFHSSWTF for messages that echo MODIFY commands.
DFHSSWTF	Suppresses passwords from the echoed copies of MODIFY CICS commands that contain signon passwords.
DFHSSWTO	Inserts the applid into all DFH messages issued under a TCB with a valid AFCB.

Exits

There are no user exits in the subsystem interface support.

Trace

No tracing is performed by the subsystem interface support.

External interfaces

Module DFHSSIN invokes the MVS module IEEMB878 to read its initialization parameters from SYS1.PARMLIB.

Module DFHSSGC invokes the MVS module IEAVG700 to start and stop console message subsystem broadcasting.

Modules DFHCSVC and DFHSSEN use the IEFSSREQ interface to communicate with the CICS subsystem.

Subtask control

Subtask control is the interface between a CICS task and a subtask. It avoids suspending CICS execution, and improves the response time.

This function is invoked by the DFHSK macro with the following calls:

- CTYPE=PERFORM activates an exit routine under a new TCB.
- CTYPE=WAIT waits for subtask to complete.
- CTYPE=RETURN returns control to the main CICS TCB.

Design overview

Some synchronous operating system requests issued by CICS modules could cause CICS to be suspended until the requests had completed. To avoid the resulting response-time degradation, certain requests are processed by the general-purpose subtask control program, DFHSKP. A CICS module calls DFHSKP to execute a routine within the module under a subtask of the operating system.

DFHSKP does the following:

- Schedules a subtask to execute a routine (called an SK exit routine)
- Allows an SK exit routine to wait on an event control block (ECB) of the operating system
- Manages subtask creation, execution, and termination
- Handles program checks or abends within the SK exit routine.

DFHSKP consists of the DFHSKM, DFHSKC, and DFHSKE programs.

DFHSKM (subtask manager program)

A DFHSK macro invokes DFHSKM to cause a routine to be executed under a subtask of the operating system. DFHSKM chooses a subtask to execute the request unless the caller has specified a particular subtask.

DFHSKM determines whether the subtask is inoperative, not started, or running. The subtask is called inoperative if it has terminated itself, or could not be attached. If the subtask is inoperative and the user coded SYNC=YES in the DFHSK macro, the request is processed synchronously; that is, DFHSKM executes the request under the CICS task control block (TCB).

If the subtask has not started, DFHSKM attaches a CICS task specifying the entry point of DFHSKC to execute. DFHSKM then waits on an ECB in the subtask control area (SKA) for the subtask and continues when the ECB is posted by DFHSKC, indicating that the subtask has been initialized.

DFHSKM then creates a work queue element (WQE) that represents the work to be performed under a subtask. The WQE is added to the work queue for the subtask. When the work ECB of the subtask is posted, signaling work to do, DFHSKM issues a wait on the work-complete ECB in the WQE. This ECB is posted when the WQE has been processed by the subtask. DFHSKM returns control to the caller, indicating the outcome of the processing.

If the subtask processing the WQE fails before completion, DFHSKM is informed and attempts to execute the request synchronously if the caller so specified.

When CICS terminates, it issues a DFHSK CTYPE=TERMINATE macro to terminate the subtasking mechanism. DFHSKM sets a flag in each subtask control area (in DFHSKP static storage) indicating that the subtask should terminate. DFHSKM then posts the subtask work ECB to signal the subtask to examine this flag.

DFHSKM is also invoked by deferred work element (DWE) processing.

When DFHSKM decides to process a WQE synchronously, control is passed to the routine specified by the caller. This routine may not complete normally and, so that DFHSKM does not lose the WQE because the task abended, it creates a DWE containing the address of the WQE. If the task abends, the DWE processor adds the WQE to the free queue.

DFHSKC (subtask control program)

DFHSKM invokes DFHSKC using the DFHKCP attach logic to start a subtask of the operating system, and wait for its completion. DFHSKM passes the address of the subtask control area in the facility control area address (TCAFCAAA) in the TCA.

DFHSKC issues an EXEC CICS GETMAIN for shared storage to pass to the subtask for use as its automatic storage. The length required is in a field in DFHSKE containing the automatic storage requirements. DFHSKC issues the ATTACH macro with the ECB option to attach the operating system subtask, and passes the address of the subtask control area.

DFHSKC issues the CICS SVC to authorize the TCB of the subtask to use the SVC.

DFHSKC issues a KC wait on the attach ECB. The module is suspended until subtask termination, when the ECB is posted. On termination, the subtask puts a return code in the subtask control area.

When the subtask completes, DFHSKC cleans up the subtask work queue. It then frees the automatic storage and terminates.

DFHSKC writes messages to CSMT from this module if it was unable to attach a subtask of the operating system, or the subtask indicated that its termination was not normal.

DFHSKE (subtask exit program)

When the subtask manager DFHSKM, executing on behalf of a CICS task, decides that a subtask is to be started, it attaches a CICS task using the DFHKC ATTACH macro and specifying the entry point of DFHSKCNA. This CICS task attaches the subtask and waits for subtask completion by means of the ECB parameter coded in the ATTACH macro.

The ATTACH macro specifies an entry point in DFHSIP (known to MVS by an IDENTIFY macro issued in DFHSIP). DFHSIP then branches to the entry point of DFHSKE, whose address is in the subtask control area.

Note: DFHSIP remains in storage after initialization has completed.

The subtask reverses the order of the in-progress queue to service requests on a first-come, first-served basis. It then loops round the in-progress queue and, for each WQE, branches to the program specified in the WQE (the SK exit routine).

The exit routine returns control to DFHSKE, either indicating that the exit routine has completed by issuing a DFHSK CTYPE=RETURN macro or requesting that execution of the SK exit is suspended until an ECB specified by the exit is posted by some component of the operating system.

When a return is requested, the ECB in the WQE is posted, causing the dispatcher domain to resume the CICS task that was waiting for the SK exit to be complete. When a wait is requested, the WQE is added to the waiting queue, which is processed later.

When all WQEs in the in-progress queue have been processed, DFHSKE examines the waiting queue. If any WQEs are on this queue, their ECB addresses are inserted into an operating-system multiple-wait queue. The subtask work ECB (posted when a WQE is added to the work queue) is put at the top of this multiple-wait list. An operating-system multiple-wait is then issued.

When the subtask regains control, an ECB has been posted. This can be because more work has arrived or because an ECB belonging to an exit routine has been posted.

The WQEs on the waiting queue are scanned, and those whose ECB has been posted are moved to the in-progress queue, with a flag on indicating that an SK exit routine is to be resumed.

Control returns to the beginning of this program which examines the work queue and proceeds as described earlier.

DFHSKE handles program checks and operating system abends. If an abend exit is driven when processing a WQE, that WQE is blamed and processing of it terminates. The CICS task requesting the processing is informed of the problem.

If an abend exit is driven when a WQE is not being processed, it is assumed to be a problem in the subtasking program. The abend is handled, and a count of failures is increased. When a threshold is reached, the subtask terminates.

The MVS exits are ESTAE and SPIE.

For normal termination, DFHSKE loops, processing WQEs and waiting when there is no work to do. The subtask checks a flag in the subtask control area to see if it has been requested to terminate. If the flag is set, the subtask terminates, indicating normal termination by setting a response code in the subtask control area for the attacher, DFHSKC.

Abnormal termination may occur when the error threshold has been reached. The subtask terminates, but sets an error return code in the subtask control area for the attacher to see. The attacher, DFHSKC, then cleans up any outstanding WQEs on the subtask queues.

Control blocks

This function has the following control blocks:

- SK static storage contains pointers to free work queue elements (WQEs) and to work queue elements.
- SKRQLIST is the parameter area passed to DFHSKP on a request. It contains the address of the code to be executed, and the address of the ECB.
- DFHSKWPS is the WKE structure containing the address of the next WQE in the chain, the contents of the parameter field from CTYPE=PERFORM, the save area for registers, and the work-complete ECB.
- DFHSKAPS is the subtask control area. Each instance of this control block describes the state of one subtask and contains the address of automatic storage to be used by DFHSKE, pointers to the WQE used by the subtask, the current WQE being processed, and the ECB for work and completion.

Modules

Module	Function
DFHSKC	The subtask control program is run by DFHSKM to start a subtask of the operating system
DFHSKE	The general-purpose multitask program is run as a subtask of the operating system
DFHSKM	The subtask manager program causes the routine to run under a subtask.

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:

- AP 00DE, for which the trace level is AP 1.

External interfaces

The following external calls are made by DFHSKC:

MVS ATTACH
To attach a new TCB

MVS DETACH
To detach a TCB

MVS POST
To post a CICS TCB.

The following external calls are made by DFHSKE:

MVS ESTAE
To establish an error exit

MVS WAIT
To synchronize with the TCB

MVS SETRP
To retry after a failure.

Syncpoint program

This allows the user to specify logical units of work by means of **syncpoints**. Any processing performed between syncpoints (provided the resources are declared as recoverable) can be reversed in the event of an error; but ***after*** a given syncpoint has been reached, the processing performed ***before*** that syncpoint cannot be reversed.

A syncpoint is also taken automatically at the end of each task.

Design overview

The syncpoint program works in conjunction with the Recovery Manager domain to provide the user with the ability to establish points in application programs at which all recoverable updates are committed. (The user can, at any time, back out any uncommitted changes by means of the rollback function.)

The syncpoint interface is provided by the DFHSPP module. DFHSPP is invoked, via the EXEC Interface module DFHEISP, when an application program issues an **`EXEC CICS SYNCPOINT`** or **`EXEC CICS SYNCPOINT ROLLBACK`** command. It is also called from other CICS modules, such as DFHMIRS.

Further important information about syncpoint processing is given in “Function shipping” on page 1706 and “Recovery manager domain (RM)” on page 948.

DFHSPP implements syncpoint calls by in turn calling the Recovery Manager domain with DFHRMUWM COMMIT_UOW or BACKOUT_UOW requests. RM calls its clients with prepare, commit, start backout etc. calls. One of RM's clients is 'APUS', serviced by module DFHAPRC. Depending on the call from RM DFHAPRC calls DFHSPP or DFHDBP to process Deferred Work Elements (DWEs). DWEs provide a mechanism whereby resource owners can record their need to

perform actions at a syncpoint. Most resource owners provide their own RM client routines, but a few, such as interval control, use DWEs.

Note that the implicit syncpoint or backout performed at task termination is effected by a direct call to the RM domain, not by issuing a DFHSP macro.

Task-related user exit resynchronization

Task-related user exit resynchronization enables a resource manager to ask CICS for the resolution of UOWs about which it is indoubt.

Task-related user exit resynchronization is called as a result of an EXEC CICS RESYNC command to restore the CICS end of the thread that was interrupted by the failure of the connection with the resource manager.

DFHERMRS passes a parameter list to DFHRMSY that consists of the following:

- rmi entryname (8 bytes). The name of the TRUE to be called for resynchronization.
- rmi qualifier (8 bytes). The qualifier to the name of the TRUE to be called for resynchronization.
- uowid (8 bytes). The id of the UOW to be resynchronized.
- resync type (1 byte). A flag that indicates whether the resynchronization is a result of an EXEC CICS RESYNC command or because of a Recovery manager domain unshunt.

DFHRMSY calls the named TRUE with a resync call giving the resolution of the named UOW. The resolution can be commit or backout. It should not be indoubt or lost to initial start. (Lost to initial start means that a START=INITIAL has been performed subsequent to the indoubt UOW being created. Initial start clears the log and the catalog meaning that Recovery manager has no knowledge of the UOW.)

To find the outcome of the UOW, DFHRMSY issues a INITIATE_RECOVERY call to Recovery manager domain for the named UOW, which returns the UOW status. DFHRMSY then builds the resync plist to pass to the TRUE, and calls the TRUE using a DFHRMCAL macro.

On return from the TRUE, if the TRUE returns an OK response indicating that it has successfully resynced with its resource manager, DFHRMSY issues a TERMINATE_RECOVERY call to RECOVERY manager domain specifying FORGET(YES). This tells RM domain it can remove this TRUE's involvement in the UOW. If no other components or TRUEs are waiting resync for the UOW, RM domain deletes its knowledge of the UOW. If the TRUE does not return with an OK response, FORGET(NO) is specified on the TERMINATE_RECOVERY call, and RM domain retains this UOW for this TRUE. A subsequent resync will be required.

Control blocks

This section describes the control blocks used by the syncpoint program:

- Deferred work element (DWE)

See Data areas for a detailed description.

Deferred work element (DWE)

A deferred work element (DWE) is created and placed on a DWE chain to save information about actions that must be taken when the unit of work terminates. These actions may depend upon whether the UOW commits or backs out.

DWEs are created by CICS control modules, and chained off field TCADWLBA in the task's TCA using DWECHAN as the chain field. The module that creates a DWE inserts the entry address of a DWE processor in field DWESVMNA of that DWE. Control is passed to this DWE processor by the syncpoint program at the end of the task or UOW.

DWEs can be used for work to be done before or after the syncpoint is logged or in the event of transaction backout.

The layout of DWEs is defined by the DFHDWEPS structure and by the DFHDWEDS assembler DSECT.

Modules

DFHSPP, DFHAPRC, DFHDBP

DFHSPP

DFHSPP can be invoked by the following macros:

DFHSP TYPE=USER
: Take a syncpoint

DFHSP TYPE=ROLLBACK
: Roll back the current unit of work

DFHSP TYPE=PHASE_1
: Do DWE processing for prepare

DFHSP TYPE=PHASE_2
: Do DWE processing for commit

When DFHSPP is called by means of a DFHSP TYPE=USER or TYPE=ROLLBACK macro the request is converted into a call to the Recovery Manager domain to commit or backout the current UOW. If the RM request fails SPP calls DFHAPAC to select an abend code corresponding to the failure reported by RM (for example ASP1 for an indoubt failure) and, in most cases, issues a PC ABEND with this abend code.

In the case of a commit or backout failure, however, no PC ABEND is issued and the transaction continues normally. In these cases CICS has, for the present, been unable to bring all local resources to the committed state for this unit of work. It has recorded any data necessary to re-attempt this at some later time, and has retained any locks necessary to preserve data integrity until then.

When DFHSPP is called by means of a DFHSP TYPE=PHASE_1 or TYPE=PHASE_2 macro SPP processes any DWEs in the DWE chain (TCADWLBA). The TYPE=PHASE_1 call is issued by DFHAPRC in response to an RM prepare or end_backout request. For each DWE in the chain that is not marked as cancelled (DWECNLM ON) or phase_2 only (DWEPHS2 ON) the DWE processor (entry address DWESVMNA) is called. In the prepare case SPP collects 'votes' and may return a YES, NO or READ-ONLY vote to its caller. Also, if necessary, a DL/I TERM call is issued to allow DFHDLI to perform end-of-UOW actions. The TYPE=PHASE_2 call is issued by DFHAPRC in response to an RM commit or shunt request. For each DWE in the chain that is marked phase 2 and not cancelled

the DWE processor is called. In the shunt case any DWE that is marked for shunting (DWESHUNT ON) is retained in the DWE chain. All other DWEs are freed.

DFHDBP

DFHDBP is link-edited with DFHAPRC and is called by DFHAPRC in response to an RM start_backout request. For each DWE in the task's DWE chain that is not marked cancelled it marks the DWE as 'backout' (DWEDYNB ON). For any BMS DWE it issues a DFHBMS TYPE=PURGE request to discard the incomplete message, otherwise it calls the DWE processor then marks the DWE as cancelled.

DFHAPRC

DFHAPRC is the module that provides the gate for the APUS Recovery Manager client. It provides keypoint and restart support for user written log records, which is described elsewhere, and syncpoint support where it serves as a receiver for RMRO calls from the RM domain for prepare, commit, etc. which it converts into appropriate calls to SPP or DBP, as described in "DFHSPP" on page 1783 and "DFHDBP."

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:

- AP 00CB, for which the trace level is AP 1.
- AP D8xx, for which the trace level is AP 1.

System dump formatting program

The system dump formatting program is for use on MVS system dump (SYS1.DUMP) data sets that record system dumps requested by CICS via the MVS SDUMP macro.

The program is invoked via the interactive problem control system (IPCS). You can use IPCS either interactively or from an MVS batch job.

The CICS-supplied sample system dump formatting program for use with CICS Transaction Server for z/OS, Version 5 Release 4 control blocks is called DFHPD710.

For further information about the system dump formatting programs, about using IPCS to format and analyze CICS dumps, and about the dump exit parameters available, see Dump utilities (DFHDU680 and DFHPD680).

Design overview

The system dump formatting program produces a formatted listing of CICS control blocks grouped within functional area. CICS dump exit parameters can be specified on the IPCS VERBEXIT subcommand to indicate whether the control block output is to be produced or suppressed for each functional (component) area. Summary reports are available for certain of the functional areas, and the dump exit parameters can also indicate whether these are to be produced or suppressed.

Modules

Module	Function
DFHAIDUF	Autoinstall terminal model manager formatter
DFHAPTRA	Application domain multiregion operation trace interpreter
DFHAPTRB	Application domain extended recovery facility trace interpreter
DFHAPTRC	Application domain user exit trace interpreter
DFHAPTRD	Application domain trace interpreter
DFHAPTRE	Application domain data tables trace interpreter
DFHAPTRF	Application domain SAA Communications and Resource Recovery interfaces trace interpreter
DFHAPTRG	Application domain ZC exception and z/OS Communications Server exit trace interpreter
DFHAPTRI	Application domain trace interpretation router
DFHAPTRJ	Application domain ZC z/OS Communications Server interface trace interpreter
DFHAPTRL	Application domain CICS OS/2 LU2 mirror trace interpreter
DFHAPTRN	Application domain autoinstall terminal model manager trace interpreter
DFHAPTRO	Application domain LU6.2 application request logic trace interpreter
DFHAPTRP	Application domain program control trace interpreter
DFHAPTRR	Application domain partner resource manager trace interpreter
DFHAPTRS	Application domain DFHEISR trace interpreter
DFHAPTRV	Application domain DFHSRP trace interpreter
DFHAPTRW	Front End Programming Interface feature trace interpreter
DFHAPTR0	Application domain old-style trace entry interpreter
DFHAPTR2	Application domain statistics trace interpreter
DFHAPTR4	Application domain transaction manager trace interpreter
DFHAPTR5	Application domain file control trace interpreter
DFHAPTR6	Application domain DBCTL DL/I trace interpreter
DFHAPTR7	Application domain LU6.2 transaction routing trace interpreter
DFHAPTR8	Application domain security trace interpreter
DFHAPTR9	Application domain interval control trace interpreter
DFHCCDUF	CICS catalog formatter
DFHCCTRI	CICS catalog trace interpreter
DFHCPDUF	SAA Communications and Resource Recovery interfaces formatter
DFHCSDUF	CSA and CSA optional features list formatter
DFHDBDUF	DBCTL and remote DL/I dump formatter
DFHDDDUF	Directory manager formatter
DFHDDTRI	Directory manager trace interpreter
DFHDMDUF	Domain manager formatter
DFHDMTRI	Domain manager trace interpreter
DFHDSDUF	Dispatcher domain formatter
DFHDSTRI	Dispatcher domain trace interpreter
DFHDUDUF	Dump domain formatter
DFHDUF	Formatting router
DFHDUFUT	Service functions routine
DFHDUTRI	Dump domain trace interpreter
DFHERDUF	Error message index processor
DFHFCDUF	File control formatter
DFHFRDUF	File control recoverable work elements formatter
DFHICDUF	Interval control formatter
DFHIPCSP	Table of CICS entries for the IPCS exit control table
DFHIPDUF	Kernel stack internal procedure formatter
DFHKEDUF	Kernel domain formatter
DFHKELOC	Routine for locating domain anchors

Module	Function
DFHKETRI	Kernel domain trace interpreter
DFHLDDUF	Loader domain formatter
DFHLDTRI	Loader domain trace interpreter
DFHLMDUF	Lock manager formatter
DFHLMTRI	Lock manager trace interpreter
DFHMEDUF	Message domain formatter
DFHMETRI	Message domain trace interpreter
DFHMNDUF	Monitoring domain formatter
DFHMNTRI	Monitoring domain trace interpreter
DFHMRDUF	Multiregion operation formatter
DFHNXDUF	Control block index processor
DFHPADUF	Parameter manager formatter
DFHPATRI	Parameter manager trace interpreter
DFHPDKW	Input parameter string validation routine
DFHPDX1	Control program
DFHPGDUF	Program manager formatter
DFHPGTRI	Program manager trace interpreter
DFHPRDUF	Partner resource manager formatter
DFHPTDUF	Program control table formatter
DFHRMDUF	Resource recovery manager formatter
DFHSMDUF	Storage manager formatter
DFHSMTRI	Storage manager trace interpreter
DFHSNTRI	Application domain signon trace interpreter
DFHSSDUF	Static storage area formatter
DFHSTDUF	Statistics domain formatter
DFHSTTRI	Statistics domain trace interpreter
DFHSUDUF	Dump domain summary formatter
DFHSUTRI	Subroutine trace interpreter
DFHSZDUF	Front End Programming Interface feature dump formatter
DFHTCDUF	Terminal control formatter
DFHTDDUF	Transient data formatter
DFHTDTRI	Transient data trace interpreter
DFHTIDUF	Timer domain formatter
DFHTITRI	Timer domain trace interpreter
DFHTKDUF	Task summary and dump formatter
DFHTMDUF	Table manager formatter
DFHTRDUF	Trace domain formatter
DFHTRFFD	Trace entry data field formatter
DFHTRFFE	Trace entry formatter
DFHTRFPB	Routine to process blocks of trace entries
DFHTRFPP	Routine for selecting trace entries to be printed
DFHTRIB	Trace entry interpretation string builder
DFHTRTRI	Trace domain trace interpreter
DFHTSDUF	Temporary-storage formatter
DFHUEDUF	User exit formatter
DFHUSDUF	User domain dump formatter
DFHUSTRI	User domain trace interpreter
DFHXMDUF	Transaction manager domain formatter
DFHXMTRI	Transaction manager domain trace interpreter
DFHXSDUF	Security domain dump formatter
DFHXSTRI	Security domain trace interpreter
DFHXRDUF	Extended recovery facility (XRF) formatter
DFHZXDUF	XRF ZCP queue formatter

Exits

Global user exit points are not applicable to offline utilities.

Trace

Trace points are not applicable to offline utilities. However, the output obtained and any messages issued by the system dump formatting program might provide clues to problems associated with corrupted data.

External interfaces

The following external calls are used by the system dump formatting program:

- MVS GETMAIN and FREEMAIN for storage management
- OPEN SVC to open DFHSNAP
- CLOSE SVC to close DFHSNAP
- MVS IPCS service routines.

System recovery program

The system recovery programs, DFHSR1, DFHSRP, and DFHSRLI, together form the default CICS recovery routine for the application (AP) domain. This routine is, in particular, the recovery routine for program checks, operating system abends, and runaway tasks that occur in user application code.

Design overview

The CICS kernel intercepts program checks, runaway tasks, operating system abends and some other internal errors for all CICS domains. The kernel then selects which CICS recovery routine to pass control to. The selected recovery routine can then process the error as appropriate.

The DFHSR1 module is the default recovery routine for the application domain. It receives control if any of the errors listed earlier occur in CICS system application programs, user application programs and some CICS nucleus modules. It processes internal errors itself but, when dealing with program checks, operating system abends, and runaway task abends, it calls the DFHSRP module. The DFHSRP module, in turn, converts the error into a transaction abend, if possible; if not possible, it terminates CICS. The DFHSRP module uses subroutines in DFHSRLI.

The transaction abend codes that can be issued are as follows:

AEYD
error detected by command protection

AICA task runaway

AKEF domain gate not active

AKEG kernel stack storage GETMAIN failure.

ASRA program check

ASRB
operating system abend

ASRD illegal macro call or attempt to access the CSA or TCA

ASRK
TCA not available

xxxx as set by issuers of deferred abend

The processing associated with each of these abends is described in "Error handling."

For further information about the abends, see CICS messages.

System recovery table

Associated with DFHSRP is the system recovery table (SRT). This is a table that the user can provide, containing operating system abend codes. It controls whether CICS recovers from program checks and operating system abends in noncritical code.

You specify the name of the system recovery table by the SRT system initialization parameter, as either SRT=NO or SRT=xx, where xx is the two-character suffix of the SRT:

- If NO is coded, CICS does not recover from program checks or operating system abends, and terminates if one occurs.
- If a suffix is coded, CICS attempts to recover from all types of program check, but can only recover from an operating system abend if the abend code appears in the SRT identified by the suffix (for example, DFHSRT1A where 1A is the suffix). If the abend code is not in the SRT, CICS terminates.

For information about how to create the SRT, see System recovery table (SRT).

Recovery initialization

The DFHSII1 module calls the DFHSR1 module during AP Domain initialization. The DFHSR1 module tells the Kernel that it is the default recovery routine for the AP domain and adds the ABAB gate.

If any error occurs when informing the kernel, CICS is terminated with message DFHSR0605 and a system dump because it is not possible to run CICS without AP domain recovery.

Error handling

The DFHSR1 module gets control from the kernel or from other AP domain modules. It determines whether it is dealing with an internal error, or an external error such as a program check. Internal errors are dealt with by exiting from the recovery environment and issuing the appropriate kernel call. If the DFHXFP or DFHEMS module has caused a program check, the DFHSR1 module exits from the recovery environment and passes control to DFHXFP or DFHEMS, respectively. All other external errors are passed on to the DFHSRP module. If control returns from the DFHSRP module, DFHSR1 issues a transaction abend. If control returns from the abend call, it is because the XPCTA exit has requested retry; in which case, DFHSR1 restores the registers and other information, and branches to the resume address.

The DFHSRP module makes an exception trace entry, ensures that it is running on the QR TCB and then deals with one of the following:

- Program check (see "Program check" on page 1789)
- Operating system abend (see "Operating system abend" on page 1790)
- Runaway task (see "Runaway task" on page 1791)
- Kernel gate error (see "Kernel gate error" on page 1791)
- Deferred abend. (see "Deferred abend" on page 1791).

Note: The kernel recovery environment is terminated very soon after DFHSRP receives control. This ensures that DFHSRP gets driven again if a subsequent error occurs in DFHSRP itself (rather than the kernel percolating the error to the next kernel stack entry). DFHSRP is therefore in a position to detect such recursive errors, and can take the appropriate action.

If DFHSRP can abend the transaction, it builds a Transaction Abend Control Block (TACB) to describe the abend. The TACB is a task-lifetime control block that records details of a transaction abend. This TACB may be used by the rest of AP domain that needs information about the abend. DFHSRP builds the TACB, rather than letting Program Control build it as part of DFHPC TYPE=ABEND processing. This enables DFHSRP to include extra information in the TACB that would otherwise be lost, such as GP registers, PSW, and FP registers at the time of the error.

Program check: For a program check, the following processing takes place in the following order:

1. If this program check occurred while DFHSRP was processing a previous program check, CICS is terminated with message DFHSR0602 and a system dump. Otherwise, DFHSRP might get caught in a recursive loop.
2. If this program check occurred while DFHSRP was processing an operating system abend, CICS is terminated with message DFHSR0615 and a system dump. This action traps program checks in global user exit XSRAB.
3. If DFHEIP or DFHEIG hired gun checking caused the program check, create an abend record for abend code AEYD and return to DFHSR1.
4. If the program check was an 0C4 protection exception, DFHSRP diagnoses the 0C4 abend further to establish whether it was caused by an attempt to access or overwrite CICS-managed protected storage. Such storage is as follows:
   - The fetch-protected dummy CSA block
   - CDSA
   - RDSA
   - UDSA
   - ECDSA
   - ERDSA
   - ETDSA
   - EUDSA
   - GCDSA
   - GUDSA

   An 0C4 abend on the CDSA, ECDSA, ETDSA, or GCDSA can occur only if storage protection is active. An 0C4 abend on the UDSA, EUDSA, or GUDSA can occur only if transaction isolation is active.

   This diagnosis is accomplished by disassembling the failing instruction, and examining the instruction operands in conjunction with the execution conditions at the time of the 0C4 abend (such as execution key). If the dummy CSA caused the 0C4 abend (that is, an attempt was made to access the CSA or TCA, or an illegal macro call was issued), message DFHSR0618 is issued. If a DSA caused the 0C4 abend, message DFHSR0622 is issued.
5. If the SRT=NO system initialization parameter was specified, you have disabled recovery, and CICS terminates with message DFHSR0603 and a system dump.
6. If a CICS system task was in control at the time of the program check, indicated by a non-numeric transaction number, CICS is terminated with message DFHSR0601 and a system dump.
7. Some special processing is performed that applies only to PL/I programs.

8. DFHSRLI is called to determine the following information:
   - The program in which the program check occurred
   - The offset in that program
   - The execution key.
9. The results of the diagnosis (program, offset, execution key, and, for an 0C4 abend, any affected DSA) are output in an exception trace.
10. Message DFHAP0001 or DFHSR0001 is issued, depending on the execution key at the time of the program check, and a system dump is taken. If the program was running in user key, message DFHSR0001 is issued; otherwise, message DFHAP0001 is issued. (See also "System dump suppression" on page 1791.)
11. Finally, DFHSRP creates an abend record and returns to DFHSR1.

Operating system abend: The following processing takes place for an operating system abend, in the order given:

1. If this abend occurred while DFHSRP was in the middle of processing a previous operating system abend, then CICS is terminated with message DFHSR0612 and a system dump. Otherwise, DFHSRP may get caught in a recursive loop.
2. If the SRT=NO system initialization parameter was specified, you have disabled recovery, and CICS terminates with message DFHSR0606. A system dump may be taken, if specified on the operating system abend.
3. If the SRT=xx system initialization parameter was specified, DFHSRP searches the SRT with the suffix xx (that is, DFHSRTxx) for the abend code. If it does not find the abend code, CICS terminates with message DFHSR0606. A system dump may be taken, if specified on the operating system abend.
4. When the abend code has been located, the next check is to see if the operating system abend occurred in a CICS system task, indicated by a non-numeric transaction number. If so, CICS terminates with message DFHSR0613 and a system dump.
5. Otherwise, the default decision is to abend the transaction with code ASRB. However, you can modify this decision by coding a global user exit program at exit point XSRAB. In addition to performing any processing that might be required for particular operating system abends, the XSRAB exit point allows you to specify whether to:
   - Terminate CICS
   - Abend the transaction ASRB
   - Abend the transaction ASRB, but cancel any active HANDLE ABEND exits.
6. If you choose to terminate CICS, CICS terminates with message DFHSR0606. A system dump may be taken, if specified on the operating system abend.
7. DFHSRLI is called to determine the following information:
   - The program in which the program check occurred
   - The offset in that program
   - The execution key.
8. The results of the diagnosis (program, offset, and execution key) are output in an exception trace.
9. Message DFHAP0001 or DFHSR0001 is issued and a system dump is taken. (See also "System dump suppression" on page 1791.)

   Whether message DFHAP0001 or DFHSR0001 is issued is governed by the execution key at the time of the program check. If the program was running in user key, message DFHSR0001 is issued; otherwise, message DFHAP0001 is issued.

10. The DFHSRP module creates an abend record with abend code ASRB, and returns to DFHSR1.

Runaway task: One of the following processing options takes place for a runaway task:

- If this runaway task occurred while DFHSRP was in the middle of processing an operating system abend, CICS terminates with message DFHSR0612 and a system dump. This traps runaway tasks caused by errors in global user exit XSRAB.
- Otherwise, the DFHSRP module creates an abend record with abend code AICA and returns to DFHSR1.

Kernel gate error: One of the following processing options takes place for a kernel gate error:

- If this error occurred while DFHSRP was in the middle of processing an operating system abend, CICS terminates with message DFHSR0612 and a system dump. This traps kernel gate errors from XPI calls in global user exit XSRAB.
- Otherwise, the DFHSRP module issues message DFHAP0001, creates an abend record with abend code AKEF, and returns to DFHSR1.

kernel stack GETMAIN error: The processing that takes place for a kernel stack GETMAIN error is identical to the processing for a kernel gate error, except that the transaction is abended with abend code AKEG.

Deferred abend: The DFHSRP module creates an abend record using the abend code set by the code that issued the deferred abend and returns to DFHSR1.

DFHSRLIM interface

This interface is used to call program DFHSRLI. It provides the following functions for DFHSRP:

INVOKE_XSRAB: This function invokes the XSRAB global user exit if active, and passes the structure SRP_ERROR_DATA to it. SRP_ERROR_DATA contains details of the operating system abend that occurred. The abend recovery option that the exit selects is returned, which is to terminate CICS, abend the transaction with abend code ASRB, or abend the transaction with abend code ASRB and cancel any active abend exits.

You can map the SRP_ERROR_DATA that is passed to the XSRAB exit by using the DFHSRED TYPE=DSECT macro.

DIAGNOSE_ABEND: This function diagnoses a program check, operating system abend, or other error, to establish the location of the error. It returns the program in which the error occurred, the offset within that program, and whether the error occurred in CICS or user application code. (A decision based on the execution key; user key implies user application code.)

System dump suppression

When message DFHAP0001 or DFHSR0001 is issued before the transaction is abended with ASRA, ASRB, ASRD, AKEF, or AKEG, the default is to take a system dump with dumpcode AP0001 or SR0001 respectively. Message DFHSR0001 is issued if CICS is running with storage protection active and is running in user key at the time of the error; otherwise, message DFHAP0001 is issued.

Therefore, it is possible to suppress the system dumps taken for errors occurring in code that is being run in user key (user application code), while retaining system dumps for errors occurring in code that is being run in CICS key (CICS code), by adding SR0001 to the dump table specifying that no system dump is to be taken.

Note that the XDUREQ Global User Exit can be used to distinguish between AP0001 situations in application and non-application code. This allows selective dump suppression when storage protection is not active or when it is active but some applications run in CICS key.

Modules

Module	Function
DFHSRP	Called by DFHSR1 to process program checks, operating system abends, runaway tasks, and so on.
DFHSRLI	Provides functions for DFHSRP, via the DFHSRLIM interface.
DFHSR1	The default recovery routine for the AP Domain.

Exits

There is one global user exit point in DFHSR1: XSRAB. This exit can be called if an operating system abend has occurred and the abend code is in the SRT.

For further information about using the XSRAB exit, see System recovery program exit XSRAB.

Trace

The following trace point IDs are provided for DFHSRP and DFHSRLI:

- AP 0701, for which the trace entry level is AP 2
- AP 0702, for which the trace entry level is AP 2
- AP 0780, for which the trace entry level is Exc
- AP 0781, for which the trace entry level is Exc
- AP 0782, for which the trace entry level is Exc
- AP 0783, for which the trace entry level is Exc.
- AP 0790, for which the trace entry level is Exc
- AP 0791, for which the trace entry level is Exc
- AP 0792, for which the trace entry level is Exc
- AP 0793, for which the trace entry level is Exc.
- AP 0794, for which the trace entry level is Exc
- AP 0795, for which the trace entry level is Exc
- AP 0796, for which the trace entry level is Exc
- AP 0797, for which the trace entry level is Exc.
- AP 0798, for which the trace entry level is Exc
- AP 0799, for which the trace entry level is Exc.
- AP 079A, for which the trace entry level is Exc.

System spooler interface

A system programmer can communicate with the local system spooler and, consequently, with other system spoolers via the system spooler network facilities. The system spooler interface single-threads its input, and it is the user's responsibility to see that all transactions get the chance to run. One high-priority transaction should not use the interface exclusively.

Further information about the system spooler interface is given in the CICS Transaction Server for z/OS Application Programming Reference.

Design overview

The system spooler interface program opens a system spooler file for either input or output, reads or writes a file, and closes a file. These functions are for system programmer use. The input is single-threaded, so only one transaction can use it at a time.

An application can send files to a remote location by specifying the node of the location, and the userid (or external writer name) of the user at that location. To retrieve a file at the remote location, you specify the external writer name, and you can then retrieve reports from that writer. For security reasons, the external writer name must begin with the same four characters as the CICS applid. The remote system to which a file or report is sent, or from which it is received, must have JES under MVS, or VM.

System spooler interface modules

The SPOOLOPEN command dynamically allocates input or output files using the CICS SVC, and an application control block (ACB) is opened to process the file. For an input file, the IEFSSREQ macro is also issued to determine which file to process. The SPOOLREAD or SPOOLWRITE commands cause GETs or PUTs to be issued using the ACB. The SPOOLCLOSE command dynamically deallocates a file, and causes it to be either transmitted or deleted. All processing which could cause CICS to be suspended is performed under an operating system subtask which is initiated by subtask control, DFHSKP.

DFHPSPST runs under CICS, but DFHPSPSS, and modules called as a result, run under the subtask.

Normal flow

When a system spooler interface command is executed, the normal sequence of invocation of modules is:

1. DFHEIP
2. DFHEPS
3. DFHPSP
4. DFHPSPSS
5. DFHPSPST
6. DFHPSSVC.

DFHPSP is called by:

- Application programs via DFHEPS issuing the DFHPS macro.
- Syncpoint program and dynamic transaction backout program to the deferred work element (DWE) module (DFHPSPDW). The entry address of DFHPSPDW is stored in the DWE. DFHPSPDW then calls DFHPSPST via DFHPS.

Abnormal flow

If a user transaction terminates without issuing a SPOOLCLOSE command, DFHPSPDW is invoked to process a DWE that was set up when the SPOOLOPEN command was processed. This closes the file in the usual way.

Modules

Module	Name
DFHEIP	DFHEIP initializes the EXEC interface structure (EIS) and then invokes the application program. Each EXEC CICS command invokes DFHEIP (nucleus) which in turn invokes the appropriate interface processor. DFHEIP also returns information to the application program through EIB (within EIS).
DFHEPS	DFHEPS is the link between DFHEIP and the JES interface program, DFHPSP.
DFHPSP	DFHPSP is the system spooler interface control module.
DFHPSPCK	DFHPSPCK is the JES interface termination processor.
DFHPSPDW	DFHPSPDW is the DWE processor.
DFHPSPSS	The system spooler interface subtask module attaches a subtask to check that a writer name and a token have been supplied. It opens and closes JES data sets, reads a record, and writes a record.
DFHPSPST	DFHPSPST is the JES interface controller.
DFHPSSVC	DFHPSSVC is the system spooler interface module that retrieves a data set name for a given external writer name, dynamically allocates it, and returns its DDNAME.

Exits

No global user exit points are provided for this interface.

Trace

The following point ID is provided for this interface:

- AP 00E3, for which the trace level is AP 1.

Table manager

The table manager controls the locating, adding, deleting, locking, and unlocking of entries in certain CICS tables. These operations can be performed while CICS is running.

Design overview

Locating, adding, deleting, locking, and unlocking entries in tables such as the terminal control table (TCT) are performed by the table manager program, DFHTMP.

Entries in these tables are also called “resources”. Because the structures of tables vary as entries are added or deleted, and a quick random access is required, a hash table mechanism is used to reference the table entries. In addition because fast access is needed for generic locates and ordered lists of entries, a getnext chain with a range table is used.

Hash table

The hash table is a set of pointers that are the addresses of directory elements of table entries. A directory element is a set of pointers; one of these pointers is the address of the table entry, the remaining pointers are the addresses of the next elements of various chains used in the different operations of the table manager. An example of a hash table is shown in Figure 75 on page 1796.

The table manager logically combines the characters of the name of the resource, and transforms the result to give an integer that is evenly distributed over the hash table size.

When an entry is located or added, the table manager places it at the head of its chain. Thus frequently used entries tend to have the minimum search times.

If the hash chains become very long, the table manager creates a larger hash table if storage is available. The hash table is enqueued before and dequeued after the reorganization, so that no references to the table can be made during reorganization.

Note: Certain TMP hash tables are not reorganized because they are also used in z/OS Communications Server SRB exits.

Range table and getnext chain

Some requests to TMP are not full key locates, but rather generic locates with a partial key. For example, requests to find all terminals whose Termid starts with two specified characters. To enable these requests, a getnext chain is maintained which orders all the directory elements alphabetically by key. There is also a 'range table' which holds pointers to certain elements along the getnext chain and a count of how many intermediate elements there are in each range.

This range table is hunted with a binary search to find the range in which a given key (full or partial) will reside, and then the getnext chain is used to find a match (if one exists) for the search condition.

A range will be split into two equal ranges if the number of intermediate elements exceeds a threshold that depends on the number of ranges and the number of elements in the table. So the ranges are dynamic, and do not depend on any particular key distribution.

The number of ranges in the table is determined when the hash table is created, and if all the ranges are full, but a range should be split, a reorganization of the ranges takes place, which increases the range threshold by a factor of 2.

Secondary indexes

A separate hash table, called the secondary index, is created for certain TMP tables, which allows the same entry to be located by another key. In certain secondary indexes, the names do not need to be unique (whereas in the primary index the name is always unique). The secondary index entry is deleted at the same time the entry in the primary index is deleted.

For example, a secondary index is created for DSNAME blocks. This allows table entries to be accessed via secondary keys, using the DSNAME block number in the case of DSNAME blocks.

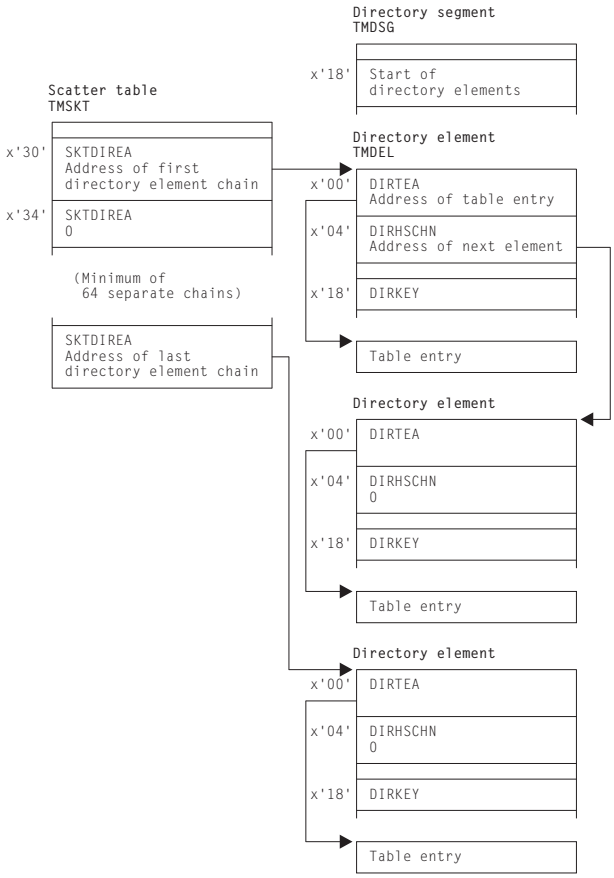


Figure 75. Example of a hash table

Certain tables also have aliases as distinct from secondary indexes. These are alternative names for the table entry, which can be used to locate a table entry. They exist in the same index as the primary name, and are not included in a getnext chain, rather they form an alias chain from the primary entry.

Functions of the table manager

The table manager performs the following functions:

Locate table entry
: For a given name, find the address of the table entry.

Get next table entry
: For a given name, find the address of the next table entry in collating sequence. This can be used repeatedly to find all entries in a range (or all elements in the whole table).

Add table entry
: For a given table entry, add it into the table.

Quiesce a table entry
: For a given name, mark its directory segment as busy.

Unquiesce a table entry
: For a given name, remove its directory segment from the 'quiesce' state.

Delete a table entry
: For a given name, delete it and any associated alias. The entry must have been quiesced first.

Create an index for a table
: Create a hash table of a given type.

Add a name into a secondary index
Given a primary name and a secondary name, add the names to the secondary index.

Add an alias name
For a given name, assign an alias name.

Get next alias name
For a given a name, find the next alias name (if any).

Lock a table entry
For a given a name, assign a read lock to it.

Unlock a directory entry
For a given a name, remove the associated read lock.

Reset lock slots
For a given name, reset the lock slots.

Transfer lock to target task
For a given a name and the address of a target TCA, transfer the read lock to the target task.

Process deferred work element
Make the changes made by the logical unit of work (LUW) visible at task syncpoint time.

Read locks

Read locks are used to prevent a table entry being deleted by the table manager.

A read lock is a fullword of storage. When DFHKCP attaches a task, it allocates storage for a number of local read locks; this storage is addressed by TCATMRLP in the TCA. Local read locks are not acquired for table entries that cannot be deleted.

Global read locks are used by the CICS modules that are executed independently of any task. They reside in the table manager static storage area (TMS) that is addressed by SSATMP in the static storage address list (SSA).

These locks are released by:

- an Unlock call,
- a Getnext call,
- a Reset call,
- the termination of the task,
- or a DWE call.

Read locks are always obtained against the primary index entry even if the request is against a secondary index or an alias.

Browse token

For Getnext requests on secondary indexes, a browse token is used to hold the name of the previously found entry. The token consists of the name found in the secondary index (which may not be unique) and the name in the primary index (which is unique).

The address of the directory entry cannot be used instead of this logical name because the entry may be returned unlocked, and so may be deleted when the next getnext request is received.

The getnext consists of locating the entry in the secondary index which has a the correct primary index, if it exists, and then moving forward in the getnext chain. If it does not, an entry with a matching secondary index name, but a higher primary

index name is located, if one exists. If that also does not exist, an entry with a higher name in the secondary index is located. This requires that entries on the getnext chain for ordered both by secondary index name and also when identical secondary index names exist, by primary index name.

Quiesce state

A table entry is moved into quiesce state by a quiesce request if no read locks (including ones obtained by the issuing task) exist for the entry. When a table entry moves into quiesced state, it is unable to be located. Locating tasks can choose to ignore or wait for quiesced entries to be unquiesced or deleted.

If the quiesce request is performed with the commit option, the only ways to release the quiesced state are:

- Unquiesce
- Delete

For commit requests, the delete takes place immediately the request completes. Otherwise, if an entry is not deleted or unquiesced by the end of the UOW the TM DWE will unquiesce the entry. In this case, a delete does not take effect until the end of the UOW.

Finding table entries in a partition dump

Figure 76 on page 1800 shows the relationship of the table manager control blocks. A general procedure for finding the required table entries in a partition dump is as follows:

1. Find the CSA.
2. Find the CSA optional features list, CSAOPFL, from its address in field CSAOPFLA (offset X'C8') in the CSA.
3. Find the static storage area address list (SSA) from its address in field CSASSA (offset X'1C0') in the CSAOPFL.
4. Find the table manager static storage area (TMS) from its address in field SSATMP (offset X'14') in the SSA.
5. Look at TMS in Data areas. The fields TMASKT1 through TMASKT24 hold the addresses of the hash tables for various control blocks. Find the hash table for the control block you are interested in:

TMASKT1	Reserved
TMASKT2	Reserved
TMASKT3	Reserved
TMASKT4	Address of profile table (PFT) entries
TMASKT5	Address of file table (FCT) entries
TMASKT6	Address of destination control table (DCT) entries for transient data queues
TMASKT7	Address of local terminal (TCTE) entries
TMASKT8	Address of remote terminal and connection (TCNT) entries
TMASKT9	Address of local connection (TCTS) entries
TMASKT10	Reserved
TMASKT11	Address of DSN
TMASKT12	Address of DSNA

TMASKT13	Address of partner resource table (PRT) entries
TMASKT14	Reserved
TMASKT15	Address of local terminal NETNAME table (TCNT) entries
TMASKT16	Address of autoinstall terminal model (AITM) table entries
TMASKT17	Address of signon table (SNT) entries
TMASKT18	Address of session (TCSE) entries
TMASKT19	Address of remote connection (TCSR) entries (secondary index)
TMASKT20	Address of indirect connection (TCSI) entries (secondary index)
TMASKT21	Address of connection NETNAME entries (TCSN) (secondary index)
TMASKT22	Address of remote terminal (TCTR) entries (secondary index)
TMASKT23	Address of generic connection NETNAME entries (TCSM) (secondary index)
TMASKT24	Address of remote terminal NETNAME (TCNR) entries (secondary index)

Use the following formula to find the offset of the individual scatter table:

```
Length(TMATTV) * (n-1) + X'08'
```

Where n = position in the table (for example, TMASKTn)

To find Length(TMATTV) (and the value of n) see Data areas.

6. Find the first directory element from its address in field SKTFDEA (offset X'10') in the hash table area.
7. Directory elements are chained together in alphabetic order. The address of the next element is in field DIRGNCHN (offset X'10').
8. Look at each directory element until you find the name of the control block you are looking for. The name is in field DIRKEY (offset X'18'). Field DIRTEA (offset X'0') holds the address of the intended control block.

Control blocks

Figure 76 on page 1800 shows the table manager control blocks.

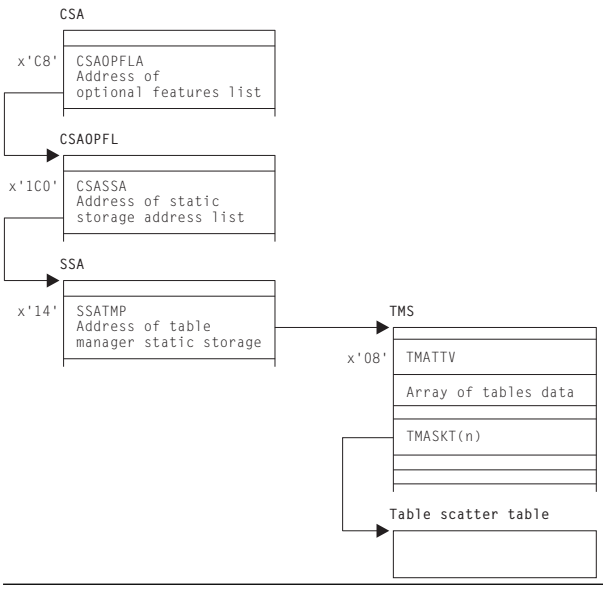


Figure 76. Table manager control blocks

See Data areas for a detailed description of these control blocks.

Modules

DFHTMP

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:

- AP 00EA, for which the trace level is AP 1.

Table Management Statistics

The statistics utility program, DFHSTUP, provides, for table management, statistics (for each table) on the amount of storage (expressed in bytes) used by the table manager to support each table (excluding storage used for the tables themselves).

Task-related user exit control

Task-related user exit support in CICS, also known as the resource manager interface (RMI), provides an interface that non-CICS resource managers can use to communicate with CICS applications. The exit program can be enabled or disabled dynamically, and useful information can be transferred to a user work area.

Functional overview

The following operations can be performed on a task-related user exit from application programs:

ENABLE

This is a global operation that names the task-related user exit and causes the task-related user exit to be loaded into storage, if it has not already been loaded. It also causes the exit program control block (EPB), which

represents the task-related user exit, and the exit's global storage to be set up by the user exit manager module, DFHUEM. The EPB also holds a TALENGTH argument and a bit-string profile for use in an exit operation. The ENABLE operation does not pass control to the task-related user exit. DFHUEM is used to enable both global user exits and task-related user exits.

The ENABLE operation is performed in two stages:
1. ENABLE
2. START.

An exit is not made available for execution until it has been both enabled and started.

You can use the TASKSTART keyword on the ENABLE command to enable a task-related user exit so that it is invoked at task start for all tasks in the CICS system.

You can also enable a task-related user exit with the FORMATEDF keyword, which means that the task-related user exit can provide formatted screens for EDF to display, whenever a DFHRMCAL request to the task-related user exit takes place.

The task-related user exit is invoked in the addressing mode of its original caller unless the LINKEDITMODE keyword is specified on the ENABLE command, in which case the exit is invoked in its own link-edit AMODE. LINKEDITMODE is only valid on the first ENABLE command for an exit program.

EXTRACT

Information concerning an "enabled and started" task-related user exit is returned to an application when it issues this command.

DISABLE

This is a global operation which in general terms is the reverse of an ENABLE request. The DISABLE operation can be performed in two stages:
1. STOP: This is the reverse of the START keyword on the ENABLE request. It causes the task-related user exit to remain in main storage together with all its associated control blocks; however it is not available for execution until an ENABLE command with the START option is specified.
2. EXITALL: This causes the EXIT and its control blocks to be deleted from main storage. The EPB however is added to a chain of re-usable EPB's anchored in the UETH. This function should not be used until all tasks that have used the exit have ended; the results of EXITALL before that point are unpredictable.

DFHRMCAL

After an exit has been enabled and started, it can be invoked from an application using a DFHRMCAL request directly, or by passing control to a stub which performs the DFHRMCAL request. A register 1 parameter list may be supplied to the task-related user exit from the application.

The task interface element (TIE) control block is created for the task and task-related user exit combination when the task issues its first DFHRMCAL request, unless the TIE has already been created because the task-related user exit was enabled for TASKSTART.

When a DFHRMCAL request is issued, control passes to DFHEIP, to DFHERM (the external resource manager interface program), and then to the task-related user exit. DFHERM manages the TIEs.

ENABLE, DISABLE, and EXTRACT are all EXEC CICS requests. DFHRMCAL is a macro.

A task-related user exit can express interest in certain types of events, and be invoked when these events take place. These events are:

- Application invocations (DFHRMCAL), associated with which are optionally the EDF screen format invocations
- System Programming interface events i.e. INQUIRE EXITPROGRAM commands
- Syncpoint related events
- Task termination events
- CICS termination.

By default, it is assumed that task-related user exits are interested in application invocations only.

Design overview

The task-related user exit interface is comparable with the EXEC interface. When an application program requests the services of a non-CICS resource manager, it does so by a module called the task-related user exit. The exit receives arguments from the application program, and passes them on to the resource manager in a suitable form.

The advantage of this method is that if the resource manager is changed, the application program that invokes the resource manager should not need to be changed too.

The exit is part of the resource manager programs. The name of the exit, or the name of the entry to the exit, is specified by the resource manager, and each application program that invokes the resource manager must be link-edited with an application program stub that refers to that name.

The exit is enabled and disabled by using the user exit manager (DFHUEM). For enabling, the resource manager can specify the size of a task-related work area that it requires.

The exit, when enabled and then driven, receives arguments in the form specified by the DFHUEXIT TYPE=RM parameter list (see Task-related user exit programs or the manual). Register 1 points to this parameter list. Register 13 points to the address of a save area, rather than the address of the CSA. The save area is 18 words long, with registers 14 through 12 stored in the fourth word onward.

Responses to the request are indicated by values placed in register 15, and also by means that are specific to the architecture of the application interface, for example, by moving data into storage areas passed by the call, or into the caller's register 15.

The main control blocks used by the task-related interface are the task interface element (TIE):

- A TIE is created by DFHERM on the first call by a task to each resource manager, and it is chained to the TCA for that task.

Task-related user exit implementation

The state of an exit is managed by DFHUEM, which is described under “User exit control” on page 1871. For an exit, the TALENGTH argument and a profile in the form of a bit-string are held in the exit program block (EPB). These arguments are

not processed until the occurrence of an application program CALL that explicitly names the exit, unless the TASKSTART keyword is used on the ENABLE request.

Entry to the exit is through the task-related user exit interface, which comprises:

- An application stub provided with the exit, but generated using the CICS-provided macro DFHRMCAL. It is this stub which explicitly names the exit, and which is link-edited with each application program that uses the application program interface (API) of the resource manager.
- DFHEIP, which is entered at DFHEIPCN by the application stub, in much the same way as EXEC CICS commands are routed at execution time.
- DFHERM, which receives control when DFHEIP discovers that the call is not for a CICS control function, but for a named exit.

DFHERM receives a set of registers (those of the caller, for example, the application program), and a routing argument which names the exit. This routing argument is constructed by DFHRMCAL, in the application stub, and is not normally visible to the application programmer. DFHERM retrieves the name of the requested exit from the routing argument, and scans any existing task interface elements (TIEs) that are chained from the task's TCA, looking for a TIE associated with the named exit. If such a TIE is not found, it searches the installed exits on a chain of EPBs, looking for the matching name. On finding a match, DFHERM constructs a TIE to represent the connection between that task and the exit. The TIE is initialized from information provided in the EPB; the TALENGTH argument defines the size of a task-local work area which can be thought of as a logical extension of the TIE. The profile string is also copied into the TIE.

DFHERM stacks (stores in a last-in, first-out manner) various parts of the program execution environment; for example, the status of HANDLE commands, file browse cursors, the EXEC interface block (EIB), and so on, and builds a parameter structure which is essentially a superset of that built by DFHUEH. Additional arguments include the task-local work area, the profile referred to earlier, and an 8-byte UOW identifier supplied by Recovery Manager.

DFHERM then passes control to the exit's entry point using standard CALL conventions, in which register 13 addresses a save area for DFHERM's own registers, register 14 addresses DFHERM's next sequential instruction, and register 1 addresses the passed parameters. This is a vector of addresses which include that of the caller's register save area. Any changes the exit makes to arguments of the application program interface (API), or to the contents of the caller's register save area, are not examined by DFHERM when it regains control, because they are not part of the CICS task-related user exit interface: rather they are the concern of the caller and the exit. However, the exit can request DFHERM to schedule certain actions by means of the profile argument. For example, the exit can request that it be informed (driven) when commitment of resources (syncpointing) is taking place, or the exit can request that DFHERM no longer routes API calls to it from this task.

Finally, on regaining control from the exit, DFHERM unstacks the objects that it had previously stacked, and returns to the caller. The state of the cursors, HANDLE labels, and so on, is apparently unchanged by the actions of DFHERM or the exit. Note that the exit may have used EXEC CICS HANDLE commands; this does not interfere with the caller's HANDLE status.

In the discussion of DFHERM so far, the term *caller* has been used for the application program. However, a caller can be a function such as syncpoint

(DFHERMSP), task control (DFHAPXM or DFHERMSP), system programming interface (DFHUEIQ), CICS termination (DFHAPDM or DFHSTP) or EDF (DFHERM). The exit can set appropriate bits in the profile (schedule flag word) so that, if the corresponding function is subsequently invoked, it in turn calls the exit. The exit can determine the identity of the caller from the first argument (called the *function definition*). This argument, passed by DFHERM, always has its first byte equal to X'00'. (If the first byte is other than X'00', the exit has been entered from DFHUEH as a global user exit.) DFHERM sets the second byte of this argument according to the type of caller, thus indicating which interface is addressed by the caller's register save area. The second byte is:

X'01' For system programming interface

X'02' For an application program

X'04' For the syncpoint program

X'08' For CICS task control

X'0A' For a CICS termination call

X'0C' For an EDF call.

Any remaining arguments are specific to each individual caller.

The flow of control for the task-related user exit interface is shown in Figure 77.

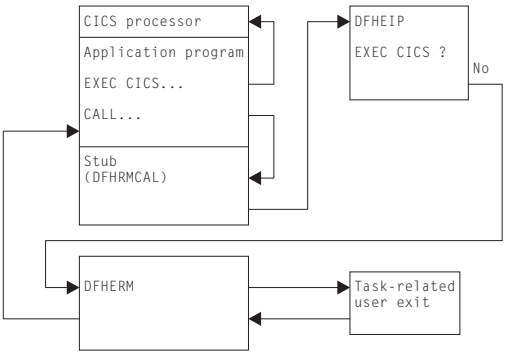


Figure 77. Task-related user exit control flow

Processors

The term "processor" is used to refer to two different types of object:

1. For the EXEC interface, it refers to the function-dependent modules associated with the EXEC interface nucleus, DFHEIP. These processors usually have names such as DFHEPC, DFHETC, DFHETD, and so on, and each of these is invoked by DFHEIP. DFHERM is also a processor of this type.
2. In various contexts, including task-related user exits, it refers to a piece of code that is link-edited with an application program and serves the dual function of:
   - Satisfying the CALL requirement for a target address—its entry resolves a V-type ADCON
   - Finding the entry point of DFHEIP.

Both these types of processor are part of the path between an application call and the functional control module that supports the request. This path appears as follows:

```
Application call
 Application processor (type 2)
  DFHEIP
   EXEC interface processor (type 1)
    Functional control module
```

Examples of the interface are:

```
EXEC CICS SYNCPOINT ... CICS API
 DFHECI     CICS COBOL EIP router
  DFHEIP
   DFHEISP  CICS syncpoint router
    DFHSPP  CICS syncpoint manager
       CICS Recovery manager domain

EXEC DLI TERM ... DLI HLPI
 DFHECI     CICS COBOL EIP router
  DFHEIP
   DFHERM   CICS RMI module
    DFHEDP  DLI HLPI manager
             (implemented as a task-related
                user exit)
```

Control blocks

The control blocks used in task-related user exit control are the exit program control block (DFHEPB), the task interface element (DFHTIEDS).

Figure 78 shows the main control blocks associated with task-related user exits.

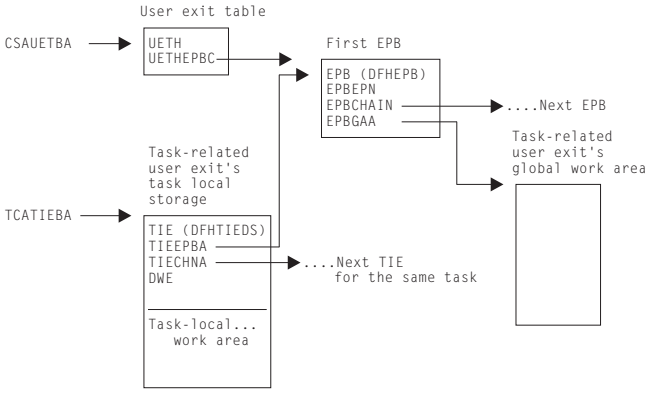


Figure 78. Control blocks associated with task-related user exits

Field CSAUETBA in the CSA points to the user exit table (UET); UETHEPBC in the UET points to the first exit program block (EPB); and EPBCHAIN in each EPB points to the next EPB in the chain.

Each EPB holds:
- The address of the exit's entry point (EPBEPN)
- The address of the global work area
- The halfword length of the global work area
- The halfword length of the task-local work area.

One EPB is associated with each enabled task-related user exit program or entry name.

EPBs used for global user exits and for task-related user exits are held on the same EPB chain.

The task-related user exit's global storage is optional. It is associated with an individual enabled task-related user exit program or entry name. Several task-related user exit programs or entry names can share the same global storage.

The task interface element (TIE) is associated with each associated pair of CICS task and task-related user exit. The first time a CICS task passes control to a particular task-related user exit, a TIE is created. The TIE lasts until task termination.

Note that all TIEs relating to a single task are chained together (more than one TIE is set up when a single CICS task makes use of more than one task-related user exit). The TIEs corresponding to a single EPB (that is, to a single task-related user exit program or entry name) are not chained together.

A global user exit may only use global storage; a task-related user exit may use both global storage and task-local work area.

Field TCATIEBA in the TCA points to the first TIE, and TIECHNA in each TIE points to the next TIE in the chain.

The TIE holds information relevant to all invocations of the task-related user exit for the task concerned. For example, TIEFLAGS holds information concerning the events for which the task-related user exit should be invoked, for example, API calls, syncpoint, and task start.

Figure 79 gives a closer look at the TIE control block chain that is used during the lifetime of a task-related user exit.

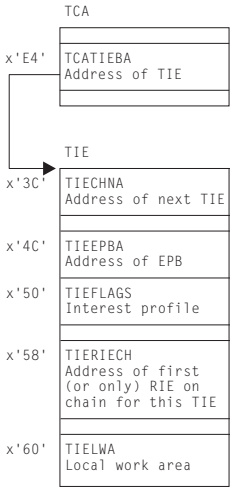


Figure 79. Control blocks used during the lifetime of a task-related user exit

Modules

Module	Function
DFHUEM	The EXEC interface processor for the ENABLE, DISABLE, and EXTRACT user exit commands.

Module	Function
DFHERM	Interfaces with task-related user exit.
DFHTIEM	Handles the TIE subpools.

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:
- AP 2520) for which the trace level is RI 1.
- AP 2521)
- AP 2522) for which the trace level is RI 2.
- AP 2523)

External interfaces

Calls are made to the task-related user exit via DFHEIP and DFHERM from the following modules:

DFHAPXM
: Task start

DFHERMSP
: Task end

DFHERMSP
: Syncpoint and backout

DFHRMSY
: For syncpoint resynchronization

DFHAPDM
: CICS termination

DFHSTP
: CICS termination

DFHUEIQ
: System programming interface for inquire exitprogram calls

Applications
: Application calls to resource manager

DFHERM
: EDF invocations for application calls to resource manager

Task-related user exit recovery

Task-related user exit recovery, also known as the resource manager interface (RMI) recovery, ensures that changes to recoverable resources performed by an external resource manager in a logical unit of work are either all committed or all backed out.

Design overview

During the execution of a CICS task, the CICS recovery manager communicates with the resource manager task-related user exit to prepare to commit, to commit unconditionally, or to back out. The purpose of these calls is to ensure that changes to recoverable resources performed in a unit of work (UOW) are either all committed or all backed out, if there is a failure anywhere in CICS or in any of the external resource managers.

Each UOW created by Recovery Manager Domain is identified by a UOW_ID and a Local UOW_ID. The LOCAL UOWID is an eight byte value whose format is easy for CICS to identify whether the UOW originated before or after an initial start.

When the resource manager receives the call to commit unconditionally or to back out, it takes the corresponding irreversible step, if possible. If the action is successful, the resource manager sends the appropriate return code. If not, it sends a return code which requests that CICS record the state of the UOW, and tries to resolve the status at a later time.

Recovery manager domain maintains the status of UOWs that require resynchronization, until all participants in the UOW have successfully resynchronized. Recovery manager domain maintains these UOWs across cold, warm and emergency start of CICS. An initial start of CICS however will mean that Recovery manager domain will lose this information and resynchronization will not be possible.

The RMI also supports an optimized syncpoint process to improve performance under certain conditions where a single-phase commit can be used. With single phase commit Recovery manager does not have to maintain resynchronization information for the RMI. This optimized process is described in more detail later in thissection .

The two-phase commit process

The RMI supports the two-phase commit process.

The following is a brief summary of the two-phase commit process and other related processing as seen from the perspective of the RMI.

- When a unit of work is first created, Recovery manager creates a local_uow_id, which is used by the RMI.
- When the task syncpoints, a prepare-to-commit request is then issued to each task-related user exit used during the current UOW. For each task-related user exit, issuing the prepare request indicates the start of phase 1 of commit processing from the perspective of CICS.
- If all syncpoint participants vote YES to the prepare requests, Recovery manager will commit the UOW. CICS then invokes each task-related user exit with a commit request. This indicates the start of phase 2 commit processing for the task-related user exit.

  If the task-related user exit cannot commit the UOW, Recovery manager will maintain a record of the status of the UOW status so that the task-related user exit can resync later.
- If one or more of the task-related user exits votes NO to the prepare-to-commit request, all the recoverable resources of the tasks are backed out.

Resolution of in-doubts: An external resource manager might be in doubt about the disposition of UOWs, for example, if the resource manager abnormally terminated after receiving a prepare request for an UOW, but before receiving the commit or backout request. The resource manager, at any time while interfacing with CICS, can supply a list of recovery tokens representing the indoubt UOWs to the task-related user exit. The task-related user exit (or other related code) can then issue an EXEC CICS RESYNC request with the indoubt list and the name of the task-related user exit as parameters.

As a result of a the EXEC CICS RESYNC command, DFHERMRS initiates a CRSY task (running program DFHRMSY) for each UOW named in the indoubt list

passed from the TRUE. DFHRMSY interfaces with Recovery manager to find out the status of the UOW, and calls the task-related user exit with the appropriate resolution, for example 'Commit', 'Backout' and so on. For each successful commit or backout, DFHRMSY informs Recovery manager that it can delete the TRUEs involvement in the UOW. When all interested parties in a UOW complete such processing, Recovery manager deletes its record of the UOW.

If an EXEC CICS RESYNC request is issued without an indoubt list or with an indoubt list of length zero, then DFHERMRS informs Recovery manager that it can remove the TRUE (identified by its name and qualifier) from all UOWs in the resynchronization set, i.e. delete all resync information for a TRUE.

A resynchronization set is first established when a TRUE is enabled. The next resynchronization set is identified on completion of an EXEC CICS RESYNC command, and is used for the next RESYNC command. A resynchronization bounds how many UOWs resync information is deleted for because RESUNC commands execute at the same time as new work is processed by a TRUE. A RESYNC command with a zero list should not delete resync information new UOW created since the resync command was issued.

The single-phase commit process

The RMI also supports the single-phase commit process for UOWs that are read-only, and for UOWs where CICS detects that only one external resource manager has been called for update requests. The task-related exit must indicate to the RMI that it is capable of processing single-phase commit requests; otherwise, a two-phase commit is used. Use of single-phase commit improves performance, because CICS does not perform any logging and the task-related user exit is called only once during syncpoint processing.

Single-phase commit for read-only UOWs: To take advantage of single-phase commit for read-only UOWs, the external resource manager must return to the task-related user exit an indicator that the UOW is read-only. This can be done by the resource manager returning a flag indicating the “history” of the UOW so far (that is, whether it is read-only so far), or returning information about the current request. In the latter case, it is the responsibility of the task-related user exit to keep a “history” of the UOW so far. After each request, the task-related user exit must return to CICS with a flag set in the parameter list indicating this history.

At syncpoint time, if CICS detects that the UOW is read-only, it invokes the task-related user exit with an “End-UOW” request instead of the normal prepare and commit requests associated with a two-phase commit. This means that the task-related user exit is invoked only once during syncpoint. The “End-UOW” request is issued during phase 2 syncpoint processing. On receiving an “End-UOW” request, the task-related user exit should invoke the resource manager for single-phase commit. There are no return codes associated with the “End-UOW” request, and CICS does not perform any logging for this type of request.

Single-phase commit for the single updater: To take advantage of single-phase commit for the single-update situation, the task-related user exit must indicate to the RMI that it knows the single-update protocol. It does this by setting a flag in the parameter list at the same time as it expresses an interest in syncpoint.

At syncpoint time, if CICS detects that only resources owned by one external resource manager were updated in the UOW, and if the task-related user exit has indicated that it understands the protocol, CICS invokes the task-related user exit

with an 'Only' request, instead of the normal prepare and commit requests associated with a two-phase commit. This means that the task-related user exit is invoked only once during syncpoint. The 'Only' request is issued during phase 1 syncpoint processing. CICS does not perform any logging for this type of request. When invoked for an 'Only' request, the task-related user exit should invoke the resource manager for single-phase commit.

There are two responses to the 'Only' request: 'OK' and 'Backed-out'. 'OK' means that the UOW was committed; 'Backed-out' means that the single-phase commit failed and the updates were backed out. It is important to note that, unlike the two-phase commit, there is no equivalent 'Remember' response. If a task-related user exit calls a resource manager for single-phase commit and, for example, the resource manager abends while processing this request, the task-related user exit remains in doubt as to the outcome of the request. The task-related user exit cannot return to CICS in this case, but instead must output diagnostic messages as appropriate, and then abend the transaction.

Recovery manager does not keep resynchronization information for UOWs using single phase commit. Because the resource manager is the only updater in the UOW, CICS is ***not*** in doubt about any of its resources. The resource manager has either committed or backed out the updates. The messages output by the task-related user exit, in conjunction with any messages output by the resource manager, can be used to determine the outcome of the UOW.

Modules

Module	Function
DFHERMRS	DFHERMRS is invoked by DFHEISP as a result of a an EXEC CICS RESYNC command. It attaches a CRSY task for each UOW identified in the IDLIST. Calls Recovery manager to delete unwanted resynchronization information.
DFHRMSY	A CRSY task (running program DFHRMSY) is attached for each indoubt UOW appearing in the indoubt list for an EXEC CICS RESYNC command. This program then issues the appropriate 'phase 2 of syncpoint' request, that is, commit or backout, to the external resource manager that issued the EXEC CICS RESYNC.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:

- AP 2540) For trace level RI Level 1
- AP 2541)
- AP 2548) For trace level RI level 2
- AP 2549)
- AP 2560) For trace level RI level 1
- AP 2561)

External interfaces

Calls are made from DFHRMSY, via DFHERM, to the task-related user exit to provide information about the disposition of the UOW, when resynchronization of in-doubts is taking place.

Terminal abnormal condition program

Terminal error processing for BSAM-supported terminals normally routes any error to the terminal abnormal condition program (DFHTACP). Depending on the type of error, DFHTACP issues messages, sets error flags, and places the terminal or line out of service.

Before default actions are taken, CICS passes control to the terminal error program (DFHTEP) for application-dependent action if necessary. On return from the terminal error program, DFHTACP performs the indicated action as previously set by DFHTACP or as altered by the TEP, a sample version of which is supplied by CICS (DFHXTEP in source code form). See "Terminal error program" on page 1836 for further information about the TEP.

Design overview

The terminal abnormal condition program (DFHTACP) is used by terminal control to analyze any abnormal conditions. Appropriate action is taken with regard to terminal statistics, line statistics, terminal status, and line status; the task (transaction) can be terminated. Messages are logged to the transient data master terminal destination (CSMT) or the terminal log destination (CSTL). DFHTACP links to the user-supplied (or sample) terminal error program, passing a parameter list via a COMMAREA that is mapped by the DFHTEPCA DSECT. This allows the user to attempt recovery from transmission errors and to take appropriate action for the task.

Table 88 lists the various TACP message processing routines, which assemble the text of the messages and write them to one of three destinations depending on the type of error.

The matrix shown in Table 89 on page 1812 shows the sequence in which the message routines are called for each error code. For example, for error code X'88', the processing routines are executed in the following order: ME, F, W, X, N, BA, and finally R.

Table 90 on page 1813 gives a generalization of TACP's default error handling upon completion of the message processing. For each error code, it shows the first routine to be called.

Table 88. TACP message routines

Routine	Function
A	Establish DFHTC message number 2501 (Msg too long, please resubmit)
D	Establish DFHTC message number 2502 (TCT search error)
F	Establish DFHTC message number 2507 (Input event rejected)
H	Establish DFHTC message number 2506 (Output event rejected)
I	Establish DFHTC message number 2513 (Output length zero)
J	Establish DFHTC message number 2514 (No output area provided)
K	Establish DFHTC message number 2515 (Output area exceeded)

Table 88. TACP message routines (continued)

Routine	Function
L	Establish DFHTC message number 2517 (Unit check SNS=ss, S.N.O.)
M	Establish DFHTC message number 2519 (Unit exception, S.N.O.)
N	Generate standard message inserts, for example, 'at term tttt'
O	Generate special inserts for message DFHTC2500
Q	Write to terminal causing the error, after retrieving the message text from ME domain using an MEME RETRIEVE_MESSAGE call
R	Write to destination (CSMT or CSTL) using an MEME SEND_MESSAGE call to ME domain
T	Obtain terminal main storage area (message build area)
V	Establish DFHTC message number 2511 (Incorrect write request)
W	Establish 'return code xx' message insert
X	Convert hexadecimal byte into 2 printable characters
AB	Establish DFHTC message number 2534 (Incorrect destination)
AE	Establish DFHTC message number 2500 (Line|CU|Terminal out of service)
AF	Obtain terminal statistics
BA	Obtain line statistics
BB	Establish DFHTC message number 2516 (Unit check SNS=ss)
BC	Establish DFHTC message number 2518 (Unit exception)
BF	Establish DFHTC message number 2521 (Undetermined unit error)
CA	Establish DFHTC message number 2522 (Intercept required)
DB	Establish DFHTC message number 2529 (Unsolicited input)
ME	Initialize parameter list for calling ME domain

Table 89. TACP message construction matrix

Error codes																
X'81'	**X'82'**	**X'84'**	**X'85'**	**X'87'**	**X'88'**	**X'8C'**	**X'8D'**	**X'8E'**	**X'8F'**	**X'94'**	**X'95'**	**X'96'**	**X'97'**	**X'99'**	**X'9A'**	**X'9F'**
ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME	ME
T																
	AE															
		D														
			V													
				DB												
					F											
						H										
							I									
								J								
									K							
										BB						
											L					
												BC				
													M			

Table 89. TACP message construction matrix (continued)

Error codes															
X'81'	**X'82'**	**X'84'**	**X'85'**	**X'87'**	**X'88'**	**X'8C'**	**X'8D'**	**X'8E'**	**X'8F'**	**X'94'**	**X'95'**	**X'96'**	**X'97'**	**X'99'**	**X'9A'**	**X'9F'**
														BF		
															CA	
																AB
A																
	O															
					W	W										
AF																
Q																
					X	X				X	X					
		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
		BA			BA	BA										
	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 90. TACP default error handling

Error code	Default action
X'81'	Abend transaction
X'82'	none
X'84'	Put line in or out of service, as required
X'85'	Abend transaction
X'87'	Unsolicited input message
X'88'	Put line (or terminal) out of service
X'8C'	Put line (or terminal) out of service
X'8D'	Abend transaction
X'8E'	Abend transaction
X'8F'	Abend transaction
X'94'	I/O error test
X'95'	I/O error test
X'96'	I/O error test
X'97'	I/O error test
X'99'	Put line (or terminal) out of service
X'9A'	Test line for next operation
X'9F'	Abend transaction

Modules

DFHTACP

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for the terminal abnormal condition program:

- AP 00E6, for which the trace level is TC 1.

DFHTACP provides trace entries immediately before and after calling DFHTEP.

Terminal control

Terminal control allows communication between terminals and application programs. z/OS Communications Server/NCP is used for most terminal data control and line control services.

Terminal control supports automatic task initiation to process transactions that use a terminal but which are not directly initiated by the terminal operator (for example, printers).

Terminal control can also provide a simulation of terminals, using sequential devices, in order to help test new applications.

Design overview

The user can specify that concurrent terminal support is to be provided by any combination of the following access methods:

- z/OS Communications Server
- Basic sequential access method (BSAM)
- Interregion communication (IRC)
- Console.

The primary function of terminal control is to take an input/output (I/O) request for a terminal and convert it to a format acceptable to the access method (z/OS Communications Server or BSAM).

Terminal control uses data that describes the communication lines and terminals, kept in the terminal control table (TCT). The TCT is generated by the user as part of CICS system definition, or dynamically as needed. The TCT entries contain terminal request indicators, status, statistics, identification, and addresses of I/O and related areas.

When CICS terminal control is used with z/OS Communications Server, z/OS Communications Server itself resides in a separate address space, having a higher priority than CICS. z/OS Communications Server-related control blocks and support programming comprise the CICS terminal control component. The application programs that run under CICS control communicate with terminals through the CICS terminal control interface with z/OS Communications Server.

z/OS Communications Server network functions allow terminals to be connected to any compatible control subsystem that is online. This enables a terminal operator to switch from one CICS system to another, or to another subsystem.

z/OS Communications Server manages the flow of data between devices in the network and z/OS Communications Server application programs such as CICS. z/OS Communications Server is responsible for:

- Connecting, controlling, and terminating communication between the z/OS Communications Server applications and terminal logical units

- Transferring data between z/OS Communications Server applications and logical units
- Allowing z/OS Communications Server applications to share communication lines, communication controllers, and terminals
- Controlling locally attached devices, that is, those not connected through a communication controller
- Providing tools to monitor network operations and make dynamic changes to the network configuration.

In a z/OS Communications Server environment, the functions of CICS terminal control include:

- Establishing communication with terminal logical units (LUs) by issuing logon requests, communicated through the access method
- Handling terminal input and passing user program requests for communication to z/OS Communications Server
- Returning terminal LUs to the access method by accepting logoff requests
- Taking measures to ensure the integrity of messages flowing to and from z/OS Communications Server
- Performing logical error recovery processing for z/OS Communications Server devices.

Terminal control issues z/OS Communications Server macros to receive incoming messages, and routes them to the appropriate CICS application program for processing. Likewise, it sends messages destined for various devices in the network to z/OS Communications Server, which then routes them to the appropriate location.

Terminal control services

The following services are performed by, or in conjunction with, terminal control:

- Service request facilities
- System control services
- Transmission facilities.

Service request facilities:

`Write request`
Sets up and issues or queues access method macros; performs journaling and journal synchronization.

`Read request`
Sets up and issues access method macros; performs journaling if required.

`Wait request`
Causes a dispatcher to suspend.

`Dispatch analysis`
Determines the type of access method and terminal used, and executes the appropriate area of terminal control.

System control services:

`Automatic task initiation`
Services requests for automatic task (transaction) initiation caused by events internal to the processing of CICS.

Task initiation
Requests the initiation of a task to process a transaction from a terminal. When an initial input message is accepted, a task is created to do the processing.

Terminal storage
Performs allocation and deallocation of terminal storage.

Transmission facilities— z/OS Communications Server:

Connection services
Accepts logon requests, requests connection of terminals for automatic task initiation, and returns terminals to the z/OS Communications Server, as specified by the user. If the terminal has not been defined, CICS uses the Communications Server logon information to autoinstall the terminal.

Transmission facilities: z/OS Communications Server/non- z/OS Communications Server:

Access method selection
Passes control to the appropriate access method routine based on the access method specified in the terminal control table.

Wait
Synchronizes the terminal control task with all other tasks in the system. When all possible read and write operations have been initiated, terminal control processing is complete and control is returned to the transaction manager to allow dispatching of other tasks.

Terminal error recovery

The resolution of certain conditions (for example, permanent transmission errors) involves both CICS and additional user coding. CICS cannot arbitrarily take all action with regard to these errors. User application logic is sometimes necessary to resolve the problem.

For the z/OS Communications Server part of the network, terminal error handling is carried out by the node abnormal condition program (NACP) and a sample node error program (NEP), provided by CICS, or a user-written node error program. For further information about these, see "Node abnormal condition program" on page 1750 and "Node error program" on page 1754.

For the portion of the telecommunication network connected to BSAM, these error-handling services are provided by the terminal abnormal condition program (TACP) and by the user-written or sample terminal error program (TEP). For further information about these, see "Terminal abnormal condition program" on page 1811 and "Terminal error program" on page 1836.

The following sequence of events takes place when a permanent error occurs for a terminal:

1. The terminal is "locked" against use.
2. The node or terminal abnormal condition program is attached to the system to run as a separate CICS task.
3. The node or terminal abnormal condition program writes the error data to a destination in transient data control if the user has defined one. This destination is defined by the user and can be intrapartition or extrapartition.
4. The node or terminal abnormal condition program then links to the appropriate node/terminal error program to allow terminal- or transaction-oriented analysis of the error. In the node or terminal error program, the user may decide, for

example, to have the terminal placed out of service, have the line placed in or out of service, or have the transaction in process on the terminal abnormally terminated.

5. The terminal is "unlocked" for use.
6. The node or terminal abnormal condition program is detached from the system if no other terminals are to be processed.

Testing facility—BSAM

To allow the user to test programs, BSAM can be used to control sequential devices, such as card readers, printers, magnetic tape, and direct-access storage devices. These sequential devices can then be used to supply input/output to CICS before actual terminals are available or during testing of new applications.

Terminal control modules (DFHZCP, DFHTCP)

Terminal control consists of two CICS resource managers: ZCP and TCP.

ZCP DFHZCP, DFHZCX, and DFHZCXR provide both the common (z/OS Communications Server and non-z/OS Communications Server) interface, and DFHZCA, DFHZCB, DFHZCC, DFHZCW, DFHZCY, and DFHZCZ provide the z/OS Communications Server-only support.

TCP DFHTCP provides the non-z/OS Communications Server support (not MVS console support).

Terminal control communicates with application programs, CICS system control functions (transaction manager, storage control), CICS application services (basic mapping support and data interchange program), system reliability functions (abnormal condition handling), and operating system access methods (z/OS Communications Server or BSAM).

Requests for terminal control functions made by application programs, BMS, or the transaction manager, are processed through the common interface of DFHZCP. Generally, terminal control requests for other CICS or operating system functions are issued by either ZCP or TCP, depending upon the terminal being serviced.

The ZCP and TCP suites of programs are loaded at CICS system initialization according to specified system initialization parameters, as follows:

- DFHTCP is loaded only if TCP=YES is specified.
- DFHZCP, DFHZCX, and DFHZCXR are always loaded.
- DFHZCA, DFHZCB, DFHZCY, and DFHZCZ are loaded only if VTAM=YES is specified (VTAM is the previous name for z/OS Communications Server).
- DFHZCC and DFHZCW are loaded only if ISC=YES is specified.

Figure 80 on page 1818 shows the relationships between the components of terminal control.

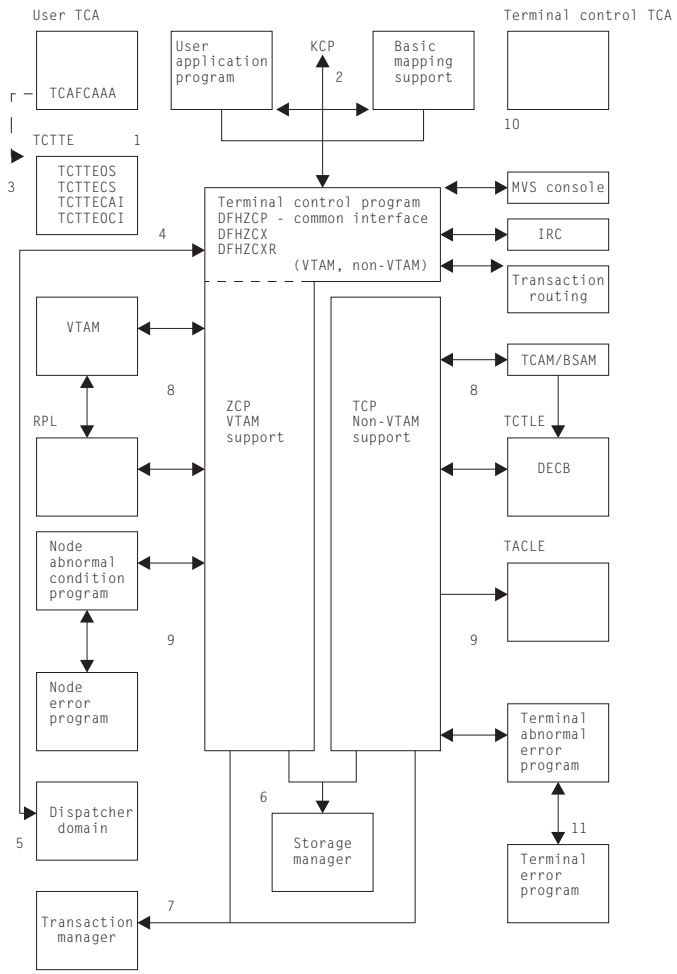


Figure 80. Terminal control interfaces

Notes:

The following notes on Figure 80 relate to the common interface.

1. When a terminal control request is issued by an application program, or internally by the basic mapping support (BMS) routines using the DFHTC macro, request bits are set in the user's task control area (TCA) and control is passed to the common interface (z/OS Communications Server, non-z/OS Communications Server) routines of DFHZCP.
2. If the request includes WAIT and the IMMED option is not in effect, control is passed to the transaction manager to place the requesting program (task) in a suspended state. If WAIT is not included, control is returned to the requesting task.
3. The task's TCA contains the TCTTE address either in a field named TCAFCAAA (facility control area associated address) or in a field named TCATPTA when passing TCATPTA to terminal control.
4. The dispatcher dispatches terminal control through the common interface (DFHZDSP in DFHZCP) for one of the following reasons:
   - The system address space exit time interval (specified by the ICV system initialization parameter) has elapsed since the last terminal control dispatch.

- The specified terminal scan delay (specified by the ICVTSD system initialization parameter) has elapsed.
- There is high-performance option (HPO) work to process.
- The terminal control event has been posted complete (for example, an exit scheduled in the case of z/OS Communications Server, or an event control block (ECB) posted in the case of non-z/OS Communications Server), and CICS is about to go into a wait condition.

5. Terminal control, through its common interface (DFHZDSP) requests the dispatcher to perform a CICS WAIT when the terminal control task has processed through the terminal network and has no further work that it can do.
6. Terminal control communicates with storage manager to obtain and release storage as follows:
   z/OS Communications Server
   ZCP modules issue domain calls for terminal storage (TIOAs), receive-any input area (RAIA) storage, and request parameter list (RPL) storage.
   Non-z/OS Communications Server
   DFHTCP issues DFHSC macros to obtain and release terminal and line storage.
7. Terminal control communicates with the transaction manager by means of the DFHKC macro. The macro can be issued by certain CICS control modules, depending upon the terminal being serviced. Terminal control may request the transaction manager to perform one of the following:
   - Attach a task upon receipt of a transaction identifier from a terminal.
   - Respond to a DFHKC TYPE=AVAIL request (a task control macro documented only for system programming) when a terminal is required by or for a task and that facility is available.
8. Terminal control communicates with operating system access methods in either of the following ways, depending upon the terminal being serviced:
   z/OS Communications Server
   ZCP (referring here to the resource manager) builds z/OS Communications Server request information in the RPL which is then passed to z/OS Communications Server for servicing. z/OS Communications Server notifies terminal control of completion by placing completion information in the RPL. ZCP analyzes the contents of the RPL upon completion to determine the type of completion and the presence of error information. Communication with z/OS Communications Server also occurs by z/OS Communications Server scheduling exits, for example, LOGON or LOSTERM. z/OS Communications Server passes parameter lists and does not always use an RPL.

   When authorized-path z/OS Communications Server has been requested (HPO), communication with z/OS Communications Server also occurs in service request block (SRB) mode (using DFHZHPRX); ZCP uses the RPL with an extension to communicate with its SRB mode code. When an SRB mode RPL request is complete, ZCP calls the relevant exit or posts the ECB, as indicated by the RPL extension.
   Non-z/OS Communications Server
   DFHTCP builds access method requests in the data event control block (DECB), which is part of the terminal control table line entry (TCTLE). The DECB portion of the TCTLE is passed to the access method by terminal control to request a service of that access method.

The access method notifies terminal control of the completion of the service through the DECB. Terminal control analyzes the contents of the DECB upon completion to determine the type of completion and to check for error information.

9. Terminal control communicates with the CICS abnormal condition functions in either of the following ways, depending upon the terminal being serviced:

   z/OS Communications Server

   The activate scan routine (DFHZACT, in the DFHZCA load module) attaches the CSNE transaction to run the node abnormal condition program (DFHZNAC); this is done during CICS initialization. DFHZNAC does some preliminary processing and then passes control to the node error program (DFHZNEP). (The node error program can be either your own version or the default CICS-supplied version.) Upon the completion of the user's error processing, control is returned to DFHZNAC. (For further information about DFHZNAC, see "Node abnormal condition program" on page 1750.)

   Non-z/OS Communications Server

   DFHTCP attaches the CSTE transaction to run the terminal abnormal condition program (TACP) and passes a terminal abnormal condition line entry (TACLE) when an error occurs. The TACLE is a copy of the DECB portion of the TCTLE and contains all information necessary for proper evaluation of the error, together with special action indicators that can be manipulated to alter the error correction procedure. After analyzing the DECB, DFHTACP calls the terminal error program (DFHTEP) with a COMMAREA containing the TACLE address. (The terminal error program can be either your own version or the default CICS-supplied version.) For further information about DFHTACP, see "Terminal abnormal condition program" on page 1811.

10. Terminal control is executed under either the user's TCA or its own TCA as follows:

    User's TCA

    a. During the application program interface
    b. During the interface with basic mapping support
    c. While performing direct z/OS Communications Server terminal SEND requests.

    Terminal control's TCA

    a. When the dispatcher dispatches terminal control
    b. When terminal control issues a request to the transaction manager to attach a task
    c. When terminal control issues a request to storage control
    d. While performing non-z/OS Communications Server terminal I/O or queued z/OS Communications Server terminal I/O
    e. For session-control functions when no task is attached.

Because many devices are supported by CICS terminal control, a large number of modules are required to provide this support.

Figure 81 on page 1821 gives an overview of the relationships between the functions within terminal control and the rest of CICS and Figure 82 on page 1821 through Figure 84 on page 1823 show some of the flows through the terminal control modules.

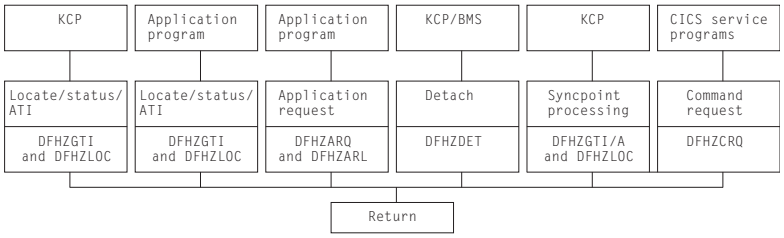


Figure 81. Terminal control functions and modules

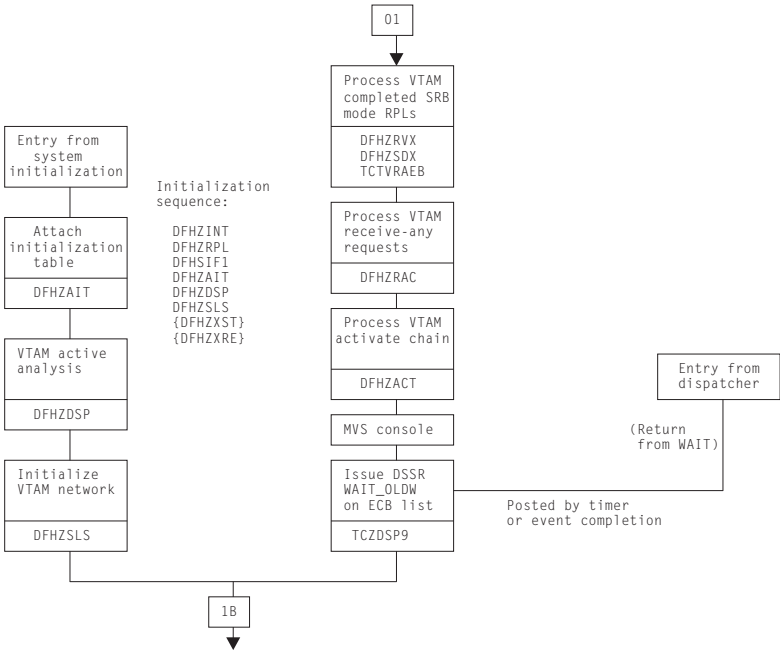


Figure 82. Terminal control ZCP and TCP common control routines

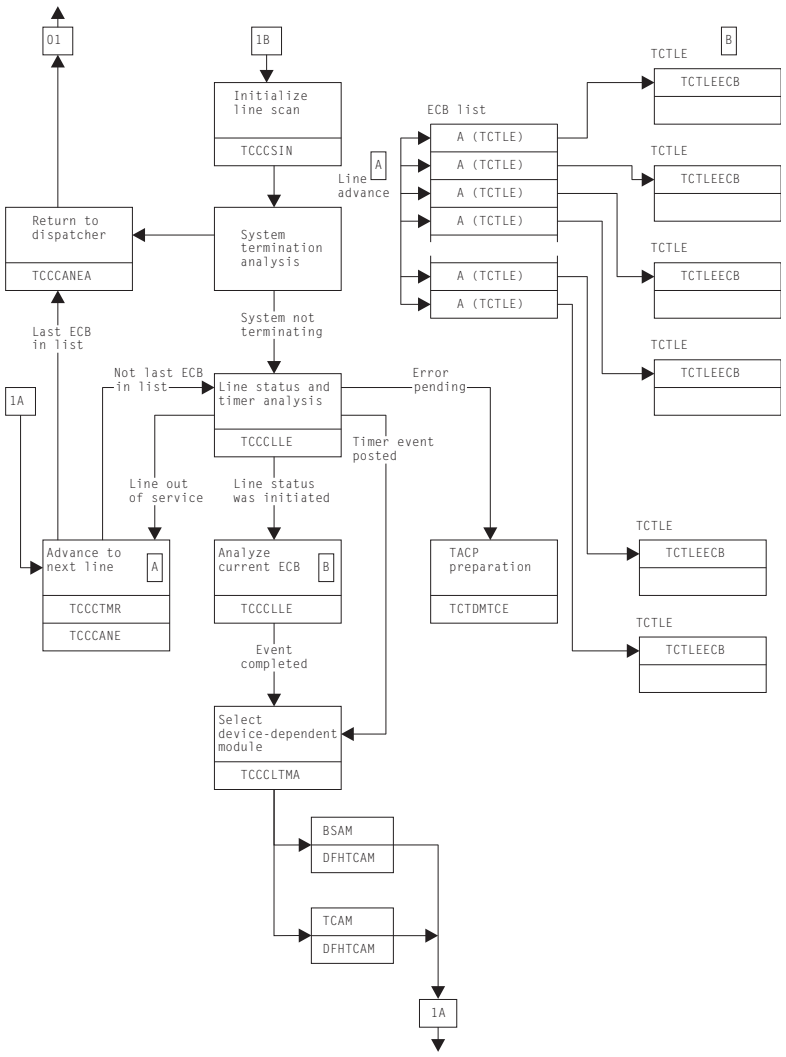


Figure 83. Terminal control TCP control routines (BSAM)

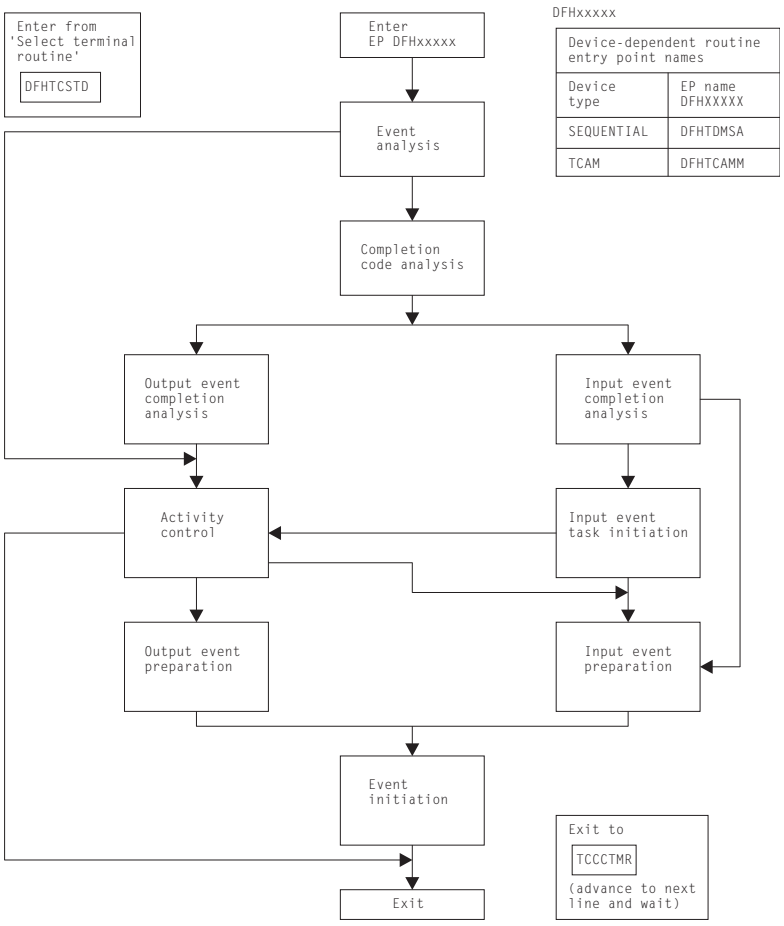


Figure 84. Terminal control general flow through device-dependent modules (TCP only)

High-performance option: When running CICS under MVS, the high-performance option (HPO) can be used. HPO uses z/OS Communications Server with CICS as an authorized program so that the z/OS Communications Server path length is reduced. This is achieved by dispatching SRBs to issue the send and receive requests for data to and from the terminals. The SRB code is executed in the DFHZHPRX module.

System console support: One or more MVS system consoles can be used as CICS terminals. This includes any MVS extended console introduced from MVS/ESA SP 4.1 onward; for example, a TSO user issuing the TSO CONSOLE command.

Each console has a unique number (releases before MVS/ESA SP 4.1) or a unique name (MVS/ESA SP 4.1 onwards). This matches the console number or name defined in the MVS system generation. Consoles are defined to CICS using CEDA DEFINE TERMINAL (see "Resource definition online (RDO)" on page 1761). The console number or name is specified using the CONSOLE or CONSNAME keyword respectively, depending on the level of MVS.

The console operator communicates with CICS using the MVS MODIFY command to start transactions. CICS communicates with the console using either the WTO macro or the WTOR macro.

A system console is modeled by CICS as a TCTTE that has an associated control block, the console control element (CCE). The CCE holds the event control block (ECB) for the console, and both the console ID and the console name.

The interface between a system console and CICS is the command input buffer (CIB), which is created in MVS-protected storage for each MODIFY command. A CIB contains the data for a MODIFY command. CICS addresses the first CIB using the EXTRACT macro and the CIBs are chained together.

The MVS communication ECB is in MVS-protected storage; it is posted complete for each MODIFY command and reset when there are no CIBs to be processed. The CICS system wait list holds pointers to the MVS communication ECB and the ECB for each system console.

When CICS is initialized, an EXTRACT macro is executed to obtain the job name and point to the MVS communication ECB and the first CIB; all these are stored in the TCT prefix.

DFHZCP contains two modules, DFHZCNA and DFHZCNR, which perform system console support.

DFHZCNA is used to:

- Resume a task on completion of a terminal event for the task
- Attach a task to satisfy a request for transaction initiation by a MODIFY command
- Attach a task (AVAIL) requested by automatic transaction initiation (ATI)
- Detach a terminal from a task when the task has completed
- Shut down console support when CICS is quiescing.

DFHZCNR is used to:

- Issue WTO macros for application program WRITE requests
- Issue WTO and WTOR macros for application program CONVERSE or (WRITE,READ) requests
- Issue a WTOR macro with message DFH4200 for application program READ requests.

Console support control modules: DFHZDSP calls DFHZCNA to scan the consoles for any activity.

DFHZCNA checks whether any task is suspended because it is waiting for a terminal event, for example, a READ, and, if the event is completed, resumes that task before starting any new task. This is done by scanning the CCE chain for ECBs that have been posted by MVS.

When a MODIFY command is executed, the communication ECB is posted complete and a CIB for the command is added to the end of the CIB chain. DFHZCNA processes the CIB chain in first-in, first-out order. For each CIB, DFHZCNA searches the CCE chain for the console. With MVS/ESA SP 4.1 (or later), the search is on console name; otherwise, the search is on console ID.

The task is then attached if the 'task pending' flag in the CCE is not set by a preceding CIB in the chain. In the course of scanning the CIB chain, DFHZCNA may find a MODIFY command that requires a task to be attached, but cannot attach the task immediately because there is already a task active, or there is an

outstanding error condition to clear. DFHZCNA therefore sets the 'task pending' flag in the CCE to remember the existence of the CIB. During the CIB chain scan, the condition preventing the task attach might clear, and a subsequent CIB might be selected for attach. However, the 'task pending' flag prevents this, and ensures that CIBs are processed in order. All 'task pending' flags are reset before each CIB chain scan.

If the task is to be attached, DFHZCNA obtains a TIOA and moves the data from the CIB to the TIOA. DFHZATT is then called to attach the task. If the attach fails, the TIOA is freed. A QEDIT macro frees the CIB if the attach is successful, and the scan continues.

When a transaction is automatically initiated and DFHKCP schedules the transaction for a terminal which is a console, a flag is set in the CCE by DFHZLOC. After DFHZCNA has completed scanning the CIB chain, it checks that the console does not have a task already attached and there is not a CIB on the chain for the console; if both these conditions are satisfied, the task is attached.

DFHZCNA issues a QEDIT macro to prevent any more MODIFY commands being accepted when CICS is shutting down. Any MODIFY commands on the CIB chain after shutdown has been started are processed. When other access methods have been quiesced, and there are no tasks attached for a console, console support is shut down.

If a console not defined to CICS is used to enter a MODIFY command, DFHZCNA sets up an error code and links to DFHACP to issue the error message. This is done using the TCTTE for the error console, CERR.

DFHZCNR sends terminal control requests from an application program to a specific system console by issuing WTO and WTOR macros. It is called by DFHZARQ.

For a WRITE request, DFHZCNR executes either a single WTO macro, or one or more multiline WTO macros, depending on the amount of data specified for the request.

For a READ request, DFHZCNR acquires a TIOA for the reply area and executes a WTOR macro with a CICS-supplied message, DFH4200. This message requests the operator to reply, and the transaction waits for this reply.

For a CONVERSE or (WRITE,READ) request, DFHZCNR acquires a TIOA for the reply area and executes a WTOR macro with the data specified for the WRITE. If there is any data remaining, DFHZCNR then executes either a single WTO macro, or one or more multiline WTO macros, depending on the amount of data. The transaction then waits until the operator replies to this request.

How terminals are defined to CICS

Terminal definitions are created as resource definitions or DFHTCT macros, and then added to the terminal control table (TCT) as TCT terminal entries (TCTTEs). DFHTCT macros can only be used for z/OS Communications Server logical units supporting logical device codes and sequential devices attached by BSAM. All z/OS Communications Server connected terminals, intersystem communication links, and multiregion operation links are defined using resource definition online (RDO).

When a cold start is performed, CICS obtains its TCT entries from DFHTCT macros or from groups of resource definitions in the CSD file, which are named in the GRPLIST system initialization parameter. These are recorded in the CICS catalog.

When a warm start is performed, CICS obtains the definitions from the DFHTCT macros and from the CICS catalog; the GRPLIST is ignored.

On emergency restart, CICS obtains the definitions from the DFHTCT macros and from the CICS catalog; the GRPLIST is ignored. Then CICS re-applies any in-flight TCT updates using information from the system log.

During CICS execution, TCT entries can be added as follows:

- By using the CEDA INSTALL command
- By the autoinstall process when an unknown terminal logs on
- By the transaction routing component when a TCT entry is shipped from a terminal-owning to an application-owning region.
- By using the EXEC CICS CREATE command

During CICS execution, TCT entries can be deleted as follows:

- By using the EXEC CICS DISCARD command
- By the autoinstall process when an autoinstalled terminal logs off or has been logged for a period.
- By the transaction routing component when a TCT entry has been unused for a period.
- Using the CEDA INSTALL, EXEC CICS CREATE, transaction routing, or autoinstall processes to replace the old entry.

Figure 85 on page 1827 shows the terminal control table (TCT).

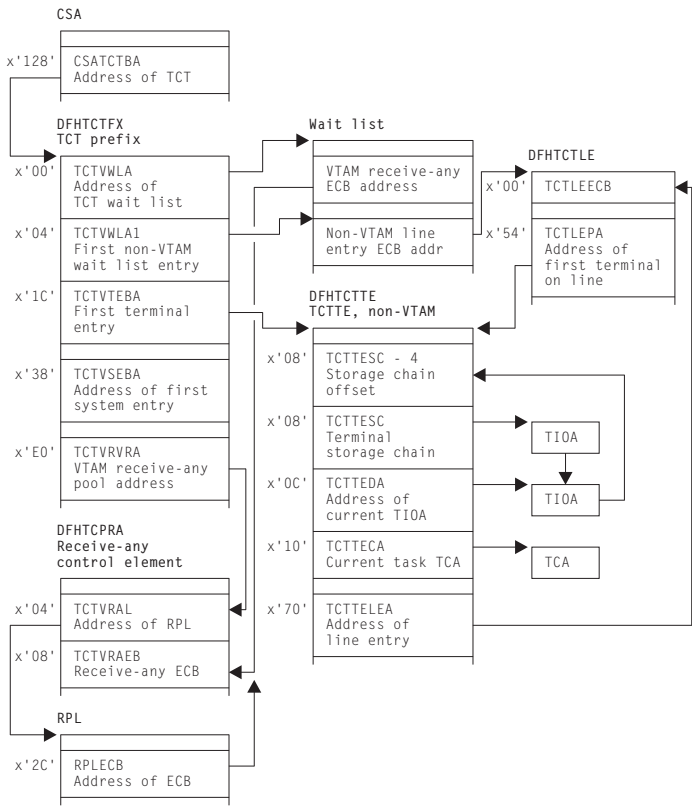


Figure 85. Terminal control table (TCT)

DFHZCQ: DFHZCQ installs, deletes, catalogs, uncatalogs, recovers, and inquires on terminals. Entries are installed in and deleted from the terminal control table by DFHZCQ. DFHZCQ is called by the following modules:

DFHAMTP
For the CEDA transaction and EXEC CICS CREATE, to install TCT entries

DFHEIQSC
For EXEC CICS DISCARD CONNECTION, to discard a connection.

DFHEIQST
For EXEC CICS DISCARD TERMINAL, to discard a terminal.

DFHTBSS
During CICS initialization, to restore terminal definitions at warm or emergency restart

DFHZATA
The autoinstall program

DFHZATD
The autoinstall delete program

DFHZATS
When a TCT entry is shipped, installed, or deleted for transaction routing

DFHZTSP
When a transaction route request is received to recatalog the connection if certain characteristics have changed.

DFHQRY
When the QUERY function is used to discover the actual characteristics of a device, complete the TCT entry, and recatalog the resulting TCTTE

DFHWKP
The warm keypoint program, to record information for RDO-eligible terminals in the CICS catalog, and to uncatalog autoinstalled entries.

DFHZCQ calls the table builder services (TBS) modules which in turn, call the appropriate DFHBSxxx modules to build the TCTTE for the input parameters. DFHZCQ is heavily dependent on the module that calls it to supply the complete set of parameters to be used to create the TCTTE; DFHZCQ itself is not responsible for determining parameters for the TCTTE.

DFHBS* builder programs: DFHZCQ calls the builder programs, whose names all begin DFHBS. These **builders** are responsible for creating TCTTEs. The parameters given to DFHZCQ are passed on to the builders, which extract the parameters and set the relevant fields in the TCTTE.

For further information about builders, see "Builders" on page 1493.

Contents of the TCT: The TCT describes the logical units (LUs) known to CICS. Each active LU is represented by a terminal control table terminal entry (TCTTE). The TCT does not describe the network configuration; it describes the CICS logical viewpoint of the network.

The TCT contains pointers to these z/OS Communications Server-related control blocks:

- Access method control block (ACB)— Link an application program, such as CICS, to z/OS Communications Server
- Receive-any control blocks (RA-RPL, RA-ECB, RACE)— Process initial transaction input
- Node initialization block (NIB) descriptors and bind-area models— Used during logon processing
- TCTTEs— Describe the logical units known to CICS
- ACB and RPL exit lists— Point to the z/OS Communications Server exit routines.

TCT indexing(DFHZGTI and DFHZLOC): There are two types of requests that can be used in CICS to locate terminal entries:

1. DFHZGTI calls
2. and DFHTC CTYPE=LOCATE calls

Both these modules use DFHTM calls to a variety of indexes and chains to locate terminal entries in the TCT with efficiency.

The DFHZGTI module has the following call types:
Locate Find a TCT entry in the given 'domain' which matches the name
GetStart
Obtain a browse token for Getnexts.
GetFirst
Find the first entry that matches the name in the given domain.
GetNext
Find the next entry that matches the name in the given domain.

GetEnd
Release the browse token
Release
Unlock an entry

Callers can decide to have an entry returned as locked or unlocked.

In DFHZGTI the total TCT is carved up into 'domains' A TCT entry can reside in several domains depending on its type. Callers to DFHZGTI specify one domain on a call and are returned one entry that fits the name (or partial name) that is supplied. DFHZGTI calls can be for the following domains:
Terminal by termid
All terminals (local, remote, non-vtam) by the terminal id (4-char).
Session by termid
All sessions (z/OS Communications Server, MRO, remote) by the terminal id (4-char).
Global by termid
All terminal and all sessions by the terminal id (4-char).
System by sysid
All connections (local, remote) by the sysid (4-char)
MRO system by sysid
MRO connections by sysid (4-char).
LU61 system by sysid
LU61 connections by sysid(4-char).
REMDEL system by sysid
Systems that need REMDEL sent to them (because they do not support timeout) when a local entry is deleted by sysid (4-char).
Terminal by netname
z/OS Communications Server local terminals by the netname (8-char).
System by netname
All connections (local, remote) by the netname (8-char).
Remote terminal by netname
Remote terminals by the netname (8-char).
Global by netname
Terminals, remote terminals and sessions by the netname (8-char).
Remote by Unique
All remote terminals and remote connections by the unique name that is Terminal-Owning-Region (TOR) netname, followed by a period, followed by the termid or sysid in the TOR. (13-char).
Remote terminal by Rsysid
Remote terminals by the value of REMOTESYSTEM (4-char).
Remote system by Rsysid
Remote connections by the value of REMOTESYSTEM (4-char).
Indirect system by Rsysid
Indirect connections by the value of REMOTESYSTEM (4-char).
Generic system by mbrname
Generic connections by the member-name of the connection in the generic z/OS Communications Server resource (8-char).

DFHTC CTYPE=LOCATE calls are processed by DFHZLOC. DFHZLOC does not have access to as wide a range of domains as DFHZGTI, but it provides extra facilities such as finding particular types of sessions for a connection. Both DFHZGTI and DFHZLOC can lock TCT entries.

Locks: The table manager program (DFHTMP) is used to locate TCT entries by both DFHZGTI and DFHZLOC. When DFHTMP gives the address of an entry, it

notes the address of the calling task, and this has the effect of a shared lock unless the caller asked for the entry not to be locked. All locks are released implicitly at the end of the task.

When a TCT entry is deleted, it must not be in use by another task. This is achieved by issuing the DFHTM QUIESCE macro. Other tasks that issue DFHTM LOCATE for that entry are suspended when they acquire a shared lock. These tasks are resumed when the original task issues a delete (if the commit option is used), or at syncpoint if not.

In addition to TMP read locks, DFHZLOC and DFHZGTI, use update locks which are obtained and released by DFHZGTA. DFHZGTA's involvement in TCT updates is discussed in "Builders" on page 1493. For efficiency, two flags in each TCT entry (one for delete and one for update) are examined before a TCT entry is returned. If either is set, and the request does not ask to see all updates, DFHZGTA is called to determine if the inquiring task holds the lock on the termid or sysid name. If it does, the entry is returned, otherwise the entry is ignored. This hides entries that are being installed or replaced from other parts of CICS until they are ready to be used, without requiring a lock search for each inquiry. The Builders, see "Builders" on page 1493, are responsible for setting and resetting the flags in the TCT entry.

The following sections describe some of the callers of DFHZCQ.

System initialization (DFHTCRP, DFHAPRDR and DFHTBSS): The DFHTCRP program is responsible for reestablishing TCTTEs that were in existence in the previous CICS run. There are three stages of processing in DFHTCRP:

1. Initialize DFHZCQ and DFHAPRDR, then exit if START=COLD
2. Reestablish TCTTEs recorded in the CICS catalog calling DFHZCQ for each one.
3. Call DFHAPRDR to allow it to proceed and forward-recover in-flight updates to TCTTEs recorded in the system log at emergency restart.

The DFHAPRDR program is called by DFHTCRP in two phases:

1. To initialize its control blocks.
2. To wait until Recovery Manager has delivered any inflight log records and DFHAPRDR (running on another task) has called DFHTBSS to recover them.

DFHAPRDR is called by Recovery Manager (RM) for each log record that are for UOWs that did not write a Forget record to the system log when CICS failed. It is then called again to denote the end of any such records. On this call DFHAPRDR waits until DFHTCRP has rebuilt the TCT from the catalog, and then calls DFHTBSS to recover each log record (which will update the TCT and catalog). Then it posts DFHTCRP to show that the TCT has recovered and returns to Recovery Manager.

The DFHTBSS program is called by DFHAPRDR with log records for TCT updates that were being written to the catalog when CICS failed. It then calls DFHZCQ to reinstall or delete the entries that the log records represent.

CEDA INSTALL and EXEC CICS CREATE (DFHAMTP): When the CEDA INSTALL command is used to install a group of TERMINAL definitions, the flow of control is as follows:

1. DFHAMP processes CEDA and EXEC CICS CREATE commands.
2. DFHAMPIL processes the INSTALL and CREATE commands.
3. DFHAMTP calls DFHTOR and then DFHZCQ.

4. DFHTOR receives as input a partial definition (TERMINAL, TYPETERM, CONNECTION, or SESSIONS), calling one of the DFHTOAxx modules, depending on the type of resource definition:
   - DFHTOAxx adds a partial definition to a BPS. For a terminal device, a complete BPS is built from information from one TYPETERM and one TERMINAL definition; for an ISC or MRO link, a complete BPSes are built from information from one CONNECTION and one (or more) SESSIONS definition(s).
   - DFHTOBPS builds the BPS, calling one of the DFHTRZxP modules to translate the parameter list into BPS format.
5. When DFHTOR has built a complete BPS, it returns it to DFHAMTP, ready to be passed to DFHZCQ.

For additional information about this process, see "Resource definition online (RDO)" on page 1761.

Autoinstall

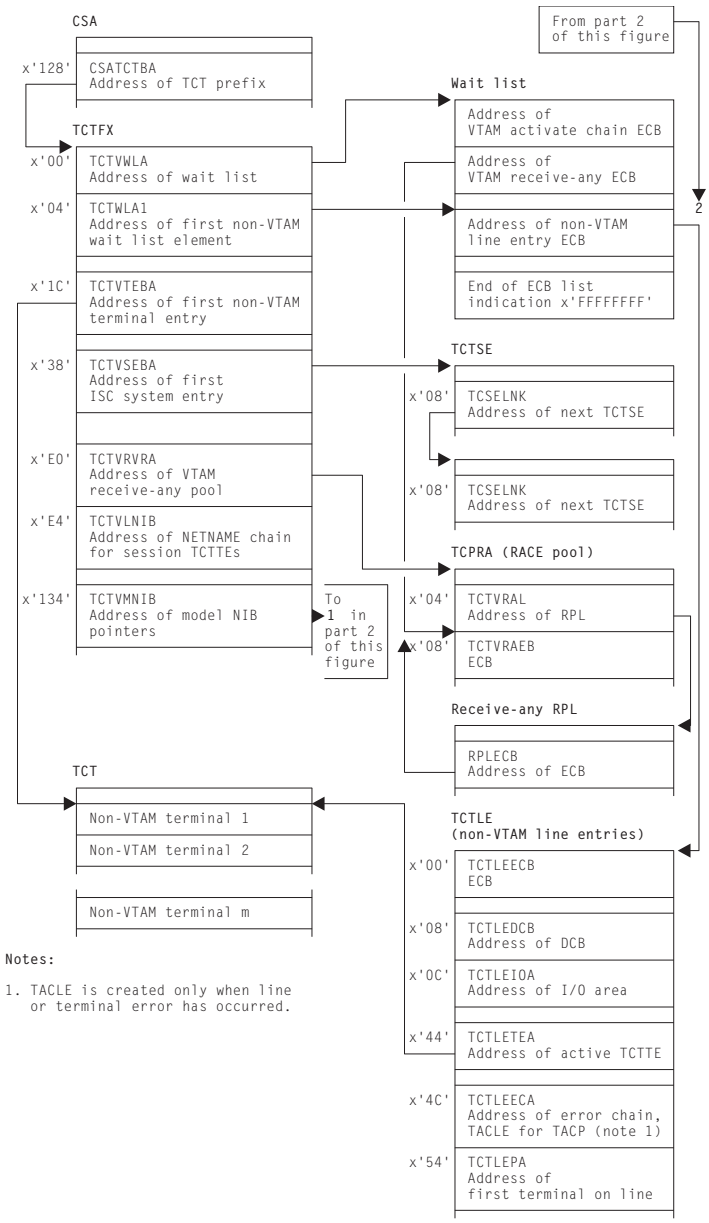
For information about this process, see "Autoinstall for terminals, consoles and APPC connections" on page 1453.

QUERY function (DFHQRY)

The QUERY function (DFHQRY) is used to determine the characteristics of IBM 3270 Information Display System devices, and complete the information about a device in the TCTTE. DFHQRY sends a read partition query structured field to the device, and analyzes the response. The TCTTE fields mainly affected are those used by basic mapping support (BMS), such as extended attributes. If QUERY(ALL) or QUERY(COLD) is specified in the terminal definition, DFHQRY is executed before any other transaction is initiated at a terminal. If QUERY(ALL) is specified, this is done after each logon. If QUERY(COLD) is specified, it is only done following the first logon after a cold start. After completing the TCTTE fields, DFHQRY calls DFHZCQ to recatalog the TCTTE.

Control blocks

Figure 86 on page 1832 shows the control blocks associated with terminal control.


CSA
x'128'
CSATCTBA
Address of TCT prefix
TCTFX
x'00'
TCTVWLA
Address of wait list
x'04'
TCTWLA1
Address of first non-VTAM
wait list element
x'1C'
TCTVTEBA
Address of first non-VTAM
terminal entry
x'38'
TCTVSEBA
Address of first
ISC system entry
x'E0'
TCTVRVRA
Address of VTAM
receive-any pool
x'E4'
TCTVLNIB
Address of NETNAME chain
for session TCTTEs
x'134'
TCTVMNIB
Address of model NIB
pointers
To
1 in
part 2
of this
figure
From part 2
of this figure
Wait list
Address of
VTAM activate chain ECB
Address of
VTAM receive-any ECB
Address of non-VTAM
line entry ECB
End of ECB list
indication x'FFFFFFFF'
2
TCTSE
x'08'
TCSELNK
Address of next TCTSE
x'08'
TCSELNK
Address of next TCTSE
TCPRA (RACE pool)
x'04'
TCTVRAL
Address of RPL
x'08'
TCTVRAEB
ECB
Receive-any RPL
RPLECB
Address of ECB
TCT
Non-VTAM terminal 1
Non-VTAM terminal 2
Non-VTAM terminal m
TCTLE
(non-VTAM line entries)
x'00'
TCTLEECB
ECB
x'08'
TCTLEDCB
Address of DCB
x'0C'
TCTLEIOA
Address of I/O area
x'44'
TCTLETEA
Address of active TCTTE
x'4C'
TCTLEECA
Address of error chain,
TACLE for TACP (note 1)
x'54'
TCTLEPA
Address of
first terminal on line
Notes:
1. TACLE is created only when line
or terminal error has occurred.

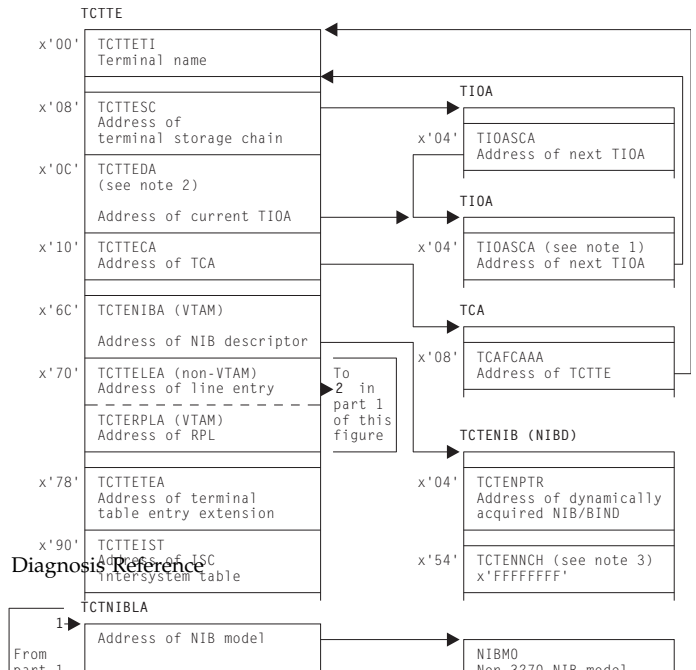

TCTTE
x'00'
TCTTETI
Terminal name
x'08'
TCTTESC
Address of
terminal storage chain
x'0C'
TCTTEDA
(see note 2)
Address of current TIOA
x'10'
TCTTECA
Address of TCA
x'6C'
TCTENIBA (VTAM)
Address of NIB descriptor
x'70'
TCTTELEA (non-VTAM)
Address of line entry
TCTERPLA (VTAM)
Address of RPL
To
2 in
part 1
of this
figure
x'78'
TCTTETEA
Address of terminal
table entry extension
x'90'
TCTTEIST
Address of ISC
intersystem table
TIOA
x'04'
TIOASCA
Address of next TIOA
TIOA
x'04'
TIOASCA (see note 1)
Address of next TIOA
TCA
x'08'
TCAFCAAA
Address of TCTTE
TCTENIB (NIBD)
x'04'
TCTENPTR
Address of dynamically
acquired NIB/BIND
x'54'
TCTENNCH (see note 3)
x'FFFFFFFF'
TCTNIBLA
1
From
Address of NIB model
NIBMO

Figure 87 shows the TCTLE and Figure 88 shows the TACLE.

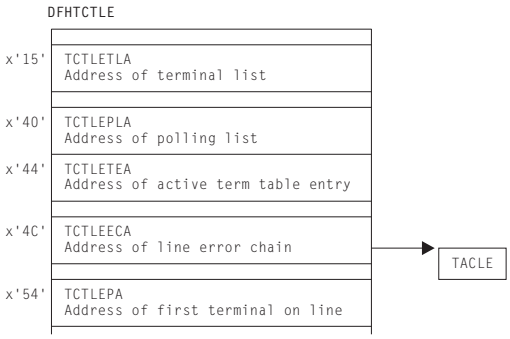


Figure 87. Terminal control table line entry (TCTLE)

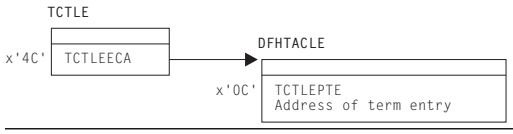


Figure 88. Terminal abnormal condition line entry (TACLE)

Terminal input/output areas (TIOAs) are set up by storage control and chained to the terminal control table terminal entry (TCTTE) as needed for terminal input/output operations. The TCTTE contains the address of the first terminal-type storage area obtained for a task (the beginning of the chain), and the address of the active TIOA.

Modules

The DFHZCx modules contain CSECTs that issue z/OS Communications Server macros to perform specific communication functions, and exit routines that are driven by z/OS Communications Server when network events occur that are related to CICS.

The following is a list of the DFHZCx load modules concerned with terminal control and z/OS Communications Server management in CICS, together with brief descriptions of their component object modules (CSECTs):

Module	CSECT	Description
DFHZCA	DFHZACT	Activate scan
	DFHZFRE	Freemain
	DFHZGET	Getmain
	DFHZQUE	Queue manager
	DFHZRST	RESETSR request
DFHZCB	DFHZATI	Automatic task initiation
	DFHZDET	Task detach
	DFHZHPSR	Authorized path SRB requests
	DFHZLRP	Logical record presentation
	DFHZRAC	Receive-any completion
	DFHZRAS	Receive-any slowdown processing
	DFHZRVS	Receive specific
	DFHZRVX	Receive specific exit
	DFHZSDR	Send response
	DFHZSDS	Send DFSYN
	DFHZSDX	Send synchronous data exit
	DFHZSSX	Send DFSYN command exit
	DFHZUIX	User input exit
DFHZCC	DFHZARER	Protocol error and exception handler
	DFHZARL	APPC application request logic
	DFHZARM	APPC migration logic
	DFHZARR	Application receive request logic
	DFHZARRA	Application receive buffer support
	DFHZARRC	Classify what next to receive
	DFHZARRF	Receive FMH7 and ER1
	DFHZBKT	Bracket state machine
	DFHZCHS	Chain state machine
	DFHZCNT	Contention state machine
	DFHZCRT	RPL_B state machine
	DFHZRLP	GDS post-z/OS Communications Server receive logic
	DFHZRLX	GDS receive exit logic
	DFHZRVL	GDS pre-z/OS Communications Server receive logic
	DFHZSDL	GDS send logic
	DFHZSLX	GDS send exit logic
	DFHZSTAP	Conversation state determination
	DFHZUSR	Conversation state machine
DFHZCP	DFHZARQ	Application request handler
	DFHZATT	Attach routine
	DFHZCNA	MVS console
	DFHZDSP	Dispatcher
	DFHZISP	Allocate/free/point
	DFHZSUP	Startup task
	DFHZUCT	3270 uppercase translate
DFHZCW	DFHZERH	APPC ERP logic
	DFHZEV1	APPC bind security (part 1)
	DFHZEV2	APPC bind security (part 2)
DFHZCX	DFHSNAS	Create signon/sign-off ATI sessions
	DFHSNPU	Preset userid signon/sign-off
	DFHSNSU	Session userid signon/sign-off
	DFHSNTU	Terminal userid signon/sign-off
	DFHSNUS	US domain - local and remote signon
	DFHSNXR	XRF reflecting signon state
	DFHZABD	Abend routine for incorrect requests
	DFHZAND	Build TACB before issuing PC abends
	DFHZCNR	MVS console request
	DFHZIS1	ISC/IRC syncpoint
	DFHZIS2	IRC internal requests
	DFHZLOC	Locate TCTTE and ATI requests
	DFHZSTU	Status changing TCTTEs/LCDs and TCTSEs

Module	CSECT	Description
DFHZCXR	DFHBSXGS	APPC session name generation
	DFHZTSP	Terminal sharing functions
	DFHZXRL	APPC command routing
	DFHZXRT	Routed APPC command handling
DFHZCY	DFHZASX	DFASY exit
	DFHZDST	SNA-ASCII translation
	DFHZLEX	LERAD exit
	DFHZLGX	LOGON exit
	DFHZLTX	LOSTERM exit
	DFHZNSP	Network services exit
	DFHZOPA	Open z/OS Communications Server ACB
	DFHZRRX	Release request exit
	DFHZRSY1	Resynchronization part 1
	DFHZRSY2	Resynchronization part 2
	DFHZRSY3	Resynchronization part 3
	DFHZRSY4	Resynchronization part 4
	DFHZRSY5	Resynchronization part 5
	DFHZRSY6	Resynchronization part 6
	DFHZSAX	Send command exit
	DFHZSCX	SESSION control input exit
	DFHZSDA	Send command
	DFHZSES	SESSIONC
	DFHZSEX	SESSIONC exit
	DFHZSHU	Shutdown z/OS Communications Server
	DFHZSIM	SIMLOGON
	DFHZSIX	SIMLOGON exit
	DFHZSKR	Send response to command
	DFHZSLS	SETLOGON start
	DFHZSYN	Handle CTYPE=syncpoint/recover request
	DFHZSYX	SYNAD exit
	DFHZTPX	TPEND exit
	DFHZTRA	Create ZCP/VIO trace requests
	DFHZXPS	APPC persistent session recovery
	DFHZXRC	XRF and persistent sessions state data analysis
DFHZCZ	DFHZCLS	CLSDST
	DFHZCLX	CLSDST exit
	DFHZCRQ	CTYPE command request
	DFHZEMW	Error message writer
	DFHZOPN	OPNDST
	DFHZOPX	OPNDST exit
	DFHZRAQ	Read ahead queuing
	DFHZRAR	Read ahead retrieval
	DFHZTAX	Turnaround exit

Exits

DFHZCB has three global user exit points: XZCIN, XZCOUT, and XZCOUT1. DFHZCP has one global user exit point: XZCATT. DFHTCP has the following global user exit points: XTCIN, XTCOUT, XTCATT, XTCTIN, and XTCTOUT.

Trace

The following point IDs are provided for terminal control:

- AP 00E6 (DFHTCP), for which the trace level is TC 2
- AP 00FC (DFHZCP), for which the trace level is TC 1
- AP FBxx, for which the trace levels are TC 1, TC 2 and Exc
- AP FCxx, for which the trace levels are TC 1, TC 2, and Exc
- AP FDxx, for which the trace level is TC 1

- AP FExx (APPC application receive requests), for which the trace levels are TC 2 and Exc.

Terminal error program

The terminal error program (DFHTEP) is invoked by the terminal abnormal condition program (DFHTACP) when an abnormal condition associated with a terminal or line occurs. The terminal error program (TEP) can be either of the following:

- The CICS-supplied sample TEP (DFHXTEP in source code form)
- A user-supplied TEP.

Design overview

The TEP analyzes the cause of the terminal or line error that has been detected by the terminal control program. The CICS-supplied version is designed to attempt basic and generalized recovery actions.

A user-supplied TEP can be used to enable processing to be performed whenever a communication system error is reported to CICS; for example, to analyze the error and accept or override the default actions set by DFHTACP.

When TEP processing is complete, control goes back to DFHTACP.

Note: Communication system errors (non-z/OS Communications Server) are passed only to DFHTEP—not to the application programs.

Guidance information about TEP coding is given in the Troubleshooting for recovery processing.

Modules

DFHTEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided specifically for this function; however, DFHTACP provides trace entries immediately before and after calling the terminal error program (see "Terminal abnormal condition program" on page 1811 for further details).

Trace control macro-compatibility interface

DFHTRP is responsible for handling all requests for trace services that are made by using the routine addressed by CSATRNAC in the CICS common system area (CSA).

Some parts of the CICS AP domain invoke DFHTRP to record trace information. This is achieved by use of the DFHTR, DFHTRACE, or DFHLFM macro.

DFHTRP converts all requests for recording trace entries into TRACE_PUT calls to the trace domain. All requests for changing the various trace flags that control tracing are converted into KEDD format calls to the kernel domain.

Design overview

The input to DFHTRP, set up by the macro used for the invocation or by the calling program directly, consists of the following TCA fields:

TCATRTR
The trace request byte. The bottom half byte has one of the following values:
- **2** User trace entry
- **3** An entry requested via DFHLFM on entry to a LIFO module
- **4** A system entry requested via DFHTR or DFHTRACE
- **5** An entry requested via DFHLFM on exit from a LIFO module.

TCATRID
The trace ID of the entry to be made. This is one byte X'nn'. The resulting trace point ID is AP 00nn.

TCATRF1/TCATRF2
Two 4-byte fields to appear as FIELD A and FIELD B in the trace entry.

TCATRRSN
An 8-character field used by some entries to specify a resource name.

The following flags in the TCA and CSA are tested by DFHTRP before making the call to the trace domain (TRACE_PUT function):

CSATRMAS (X'80' bit in CSATRMF1)
The trace master flag. This is off unless at least one of internal, auxiliary, or GTF trace is active.

TCANOTRC (X'40' bit in TCAFLAGS)
This is set according to the TRACE (YES|NO) specification on the TRANSACTION definition for the transaction ID used to start this task. It allows suppression of all trace activity for specified transaction IDs.

X'80' bit in TCATRMF
This is the user entry 'single' flag. It allows suppression of user trace entries for the associated task.

The process flow is as follows:

1. Test appropriate flags and exit if trace not required.
2. Execute data collection routine specific to trace ID in TCATRID to set up fields in trace entry.
3. Call TR domain with TRACE_PUT call to write the entry to the active destinations.
4. Invoke the storage violation trap (if this has been activated) by using the CSFE DEBUG transaction, or by using the CHKSTSK or CHKSTRM startup override. See Dealing with storage violations for information.

Modules

DFHTRP

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for trace entries recording "trace on" and "trace off" calls to DFHTRP:

- AP 00FE, for trace turned on
- AP 00FF, for trace turned off.

There are no corresponding trace levels for these point IDs; that is, the trace entries are always produced.

Trace formatting

There are three possible destinations for CICS trace entries:

Internal
To main storage in the CICS region

Auxiliary
To a BSAM data set managed by CICS

GTF To the MVS-defined destination for generalized trace facility (GTF) records.

This section describes the code used to interpret and format CICS trace entries from all of these destinations when they are processed offline.

In this context, "formatting" is used to mean the overall process of producing a report, suitable for viewing or printing, from trace data in a dump or trace data set. "Interpretation" is the process of taking just the point ID and the data fields from a trace entry and producing a character string describing what the entry represents.

There are four environments for trace formatting:

- Internal trace in transaction dump
- Internal trace in system dump
- Printing auxiliary trace data set
- Printing GTF trace data set or processing GTF records in an SDUMP.

Table 91. CICS trace formatting summary

	Transaction dump printout	**System dump printout**	**Auxiliary trace printout**	**GTF trace printout**
CICS trace type	Internal	Internal	Auxiliary	GTF Â
Data set	DFHDMPx	SYS1.DUMPnn	DFHxUXT	SYS1.DUMPnn or SYS1.TRACE Â
Controlling program	DFHDU710	DFHTRDUF	DFHTRPRA	DFHTRPRG Â
Load module name	DFHDU710	DFHPD710	DFHTU710	AMDUSREF (alias DFHTR710)

Design overview

The controlling program (DFHDU710, DFHTRDUF, DFHTRPRA, or DFHTRPRG) is responsible for acquiring the trace formatting control area (TRFCA), which is used for communication between the different routines.

As far as possible, the necessary code is constructed of routines that can run in all four environments. Subroutines required by the common code that cannot themselves be common (such as the line print subroutine) have their addresses placed in the TRFCA by the controlling program.

The controlling routines are:

DFHDU710
The dump utility program used to print transaction dumps. Invokes DFHTRFPB for each internal table block.

DFHTRDUF
The system dump formatting routine for the trace domain. Invokes DFHTRFPB for each internal table block.

DFHTRPRA
The main routine of the trace utility program DFHTU710 used to print an auxiliary trace data set. Invokes DFHTRFPP to encode selective print parameters. Invokes DFHTRFPB for each auxiliary trace block.

DFHTRPRG
The main routine of the GTF format appendage for CICS entries (format ID X'EF') AMDUSREF (alias DFHTR710). Invokes DFHTRFPP to encode selective print parameters. Invokes DFHTRFFE for each trace entry.

A noncommon subroutine required in all four environments is:

TRFPRL
Print a specified character buffer. This is contained in the controlling program.

The common routines required in more than one environment are:

DFHTRFPP
Process parameters. Passed a character string, encodes the string as selective print parameters into the TRFCA (for DFHTRPRA and DFHTRPRG only). See Dump utilities (DFHDU680 and DFHPD680) for details of the selective print parameters.

DFHTRFPB
Process block. Processes a trace block from a dump or auxiliary trace data set, calling DFHTRFFE for each entry in the block.

DFHTRFFE
Format entry. Passed a trace entry, it calls DFHxxTRI, TRFPRL, and DFHTRFFD to produce the formatted entry.

DFHTRFFD
Format data. To format and print the trace data fields of a particular entry in hex and character form. Calls TRFPRL to print each line.

DFHxxTRI
The interpretation routine for the *xx* domain. Builds the interpretation string for a particular entry given the trace point ID and the data fields from the entry. The AP domain routine DFHAPTRI calls one of the interpretation routines DFHAPTRx. Each of these is responsible for a functional component of the AP domain.

DFHTRIB
The interpretation build program. Adds printable data to the interpretation buffer in the TRFCA as requested by the interpretation routine.

DFHCDCON
The interpretation of some trace entries requires analysis of domain call parameter lists. Converts a hexadecimal parameter list into a printable list of keywords. If the resulting interpretation string would have been more than 1024 bytes long if all keywords were included, the warning '<<INTERPRETATION OVERFLOWED>>' is printed with the string.

DFHxxyyT
The data file for an *xxyy* format parameter list that is used by DFHCDCON to translate the hexadecimal parameter list into a printable list of keywords.

The components of the trace formatting function are shown in Figure 89.

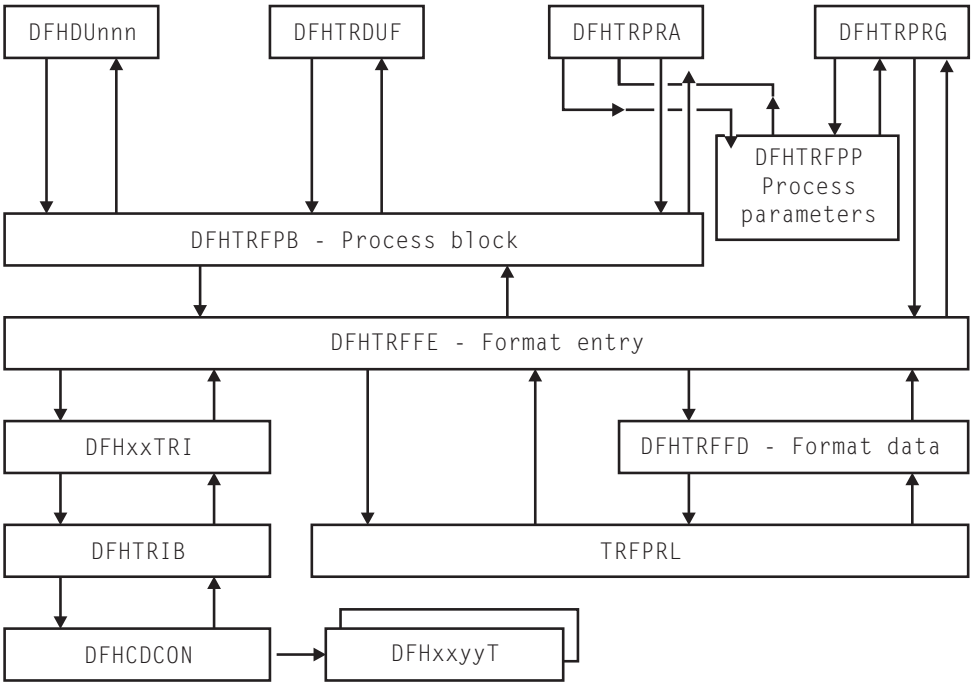


Figure 89. Trace formatting components

Segmented entries on GTF

CICS trace entries can be split into several entries in the GTF data set, because there is a 256-byte restriction on GTF entry length. However, in a formatted GTF trace, CICS reconstructs the segmented entries to ensure that all the entries for any region are formatted completely and correctly. This makes the segmentation of the entries transparent.

In the GTF data set, the segmented entries might be interleaved in the following ways:

- Header and continuation entries of different types of trace might be interleaved.
- Records from different CICS regions that write to the same GTF data set might be interleaved.

To format a GTF trace, CICS uses 4 KB buffers to reconstruct the segmented entries. For each type of trace there can be up to five 4 KB buffers for reconstructing the entries. The reconstructed trace entries appear in order of completion, and so they might be out of time sequence.

GTF entries with the CICS format ID X'EF' are written from parts of CICS that run asynchronously with the mainline code, as well as from the trace domain itself. The source of the entry is identified by the type byte in TREN_TYPE in the entry header. See DFHTREN in Data areas for a full description of the trace entry header.

```
Type      Source of entry
 00         TR domain
 01         not used
 02         DFHMNSVC
 03         normal CICS z/OS Communications Server exit
 04         CICS z/OS Communications Server LERAD/SYNAD exit
 05         CICS z/OS Communications Server TPEND exit
 06         CICS z/OS Communications Server HPO exit
 07         CICS z/OS Communications Server HPO LERAD/SYNAD exit
```

Control blocks

The trace formatting control area (TRFCA) is used as a communication area between the routines that go to make up each of the four trace formatting load modules. See Data areas for details of DFHTRFCA.

Modules

Module	Function
Controlling programs	
DFHDU710	Internal trace in transaction dump
DFHTRDUF	Internal trace in system dump
DFHTRPRA	Auxiliary trace
DFHTRPRG	GTF trace
Common routines	
DFHTRFPB	Process trace block
DFHTRFPP	Process selective print parameters
DFHTRFFE	Format trace entry
DFHTRFFD	Format data from entry
DFHTRIB	Interpretation build routine
DFHCDCON	Parameter list decode routine
Trace interpretation routines	
DFHAPTRA	MRO entries
DFHAPTRB	XRF entries
DFHAPTRC	User exit management entries
DFHAPTRD	DFHAPDM/DFHAPAP entries
DFHAPTRE	Data tables entries
DFHAPTRF	SAA communications and resource recovery entries
DFHAPTRG	ZC exception and z/OS Communications Server exit entries
DFHAPTRI	Application domain entries (router)
DFHAPTRJ	ZC z/OS Communications Server interface entries
DFHAPTRL	CICS OS/2 LU2 mirror entries. CICS OS/2 is no longer supported.
DFHAPTRN	Autoinstall terminal model manager entries
DFHAPTRO	LU6.2 application request logic entries
DFHAPTRP	Program control entries
DFHAPTRR	Partner resource manager entries
DFHAPTRS	DFHEISR trace entries
DFHAPTRV	DFHSRP trace entires
DFHAPTRW	Front End Programming Interface feature entries
DFHAPTR0	Old-style entries
DFHAPTR2	Statistics entries
DFHAPTR4	Transaction manager entries
DFHAPTR5	File control entries
DFHAPTR6	DBCTL entries
DFHAPTR7	Transaction routing entries
DFHAPTR8	Security entries
DFHAPTR9	Interval control entries

Module	Function
DFHCCTRI	Local and global catalog domain entries
DFHDDTRI	Directory manager entries
DFHDMTRI	Domain manager domain entries
DFHDSTRI	Dispatcher domain entries
DFHDUTRI	Dump domain entries
DFHKETRI	Kernel domain entries
DFHLDTRI	Loader domain entries
DFHLGTRI	Log Manager domain entries
DFHL2TRI	Log Manager domain entries
DFHLMTRI	Lock manager domain entries
DFHMETRI	Message domain entries
DFHMNTRI	Monitoring domain entries
DFHNQTRI	Enqueue domain entries
DFHPATRI	Parameter manager domain entries
DFHPGTRI	Program manager domain entries
DFHRMTRI	Recovery Manager domain entries
DFHSMTRI	Storage manager domain entries
DFHSNTRI	Signon entries
DFHSTTRI	Statistics domain entries
DFHTITRI	Timer domain entries
DFHTRTRI	Trace domain entries
DFHTSITR	Temporary Storage domain entries
DFHUSTRI	User domain entries
DFHXMTRI	Transaction manager domain entries
DFHXSTRI	Security domain entries

Exits

Global user exit points are not applicable to offline utilities.

Transaction Failure program

The abnormal condition program has been divided into two new programs according to function.

1. **DFHTFP** which is a new program that is invoked after transaction initialization on abnormal termination.
2. **DFHACP** which is invoked by transaction manager whenever an incorrect transaction is detected.

The transaction failure program (DFHTFP) is invoked during transaction abend processing. Its purpose is to reset the status of a terminal attached to the transaction, and to send a message informing the terminal operator that the transaction has abended. It also calls the user-written (or default) program error program (DFHPEP), and writes a message to the CSMT transient data destination.

DFHTFP resolves any abnormal conditions other than those associated with a terminal, or those handled directly by the operating system.

Design overview

Errors can be classified as belonging in either of two broad categories:

1. **DFHTFP**. Task abnormal conditions, which are detected by CICS control programs and are often due to an application program destroying system

control information. When this happens, the task is terminated, the program error program (DFHPEP) is called, the terminal operator is, if possible, informed of the error, and the error is logged at destination CSMT. If the transaction has entered syncpoint processing, then DFHPEP is **NOT** called.

2. **DFHACP**. Operator errors, such as incorrect transaction identifiers, security key violations, or failure of an operator to sign on to the system before attempting to communicate with CICS. When any of these happens, the program error program is **NOT** called, the terminal operator is notified, and the error is logged at destination CSMT.

Figure 90 and Figure 91 on page 1844 show the interfaces between the abnormal condition programs, DFHTFP and DFHACP, and other components when an error has been detected.

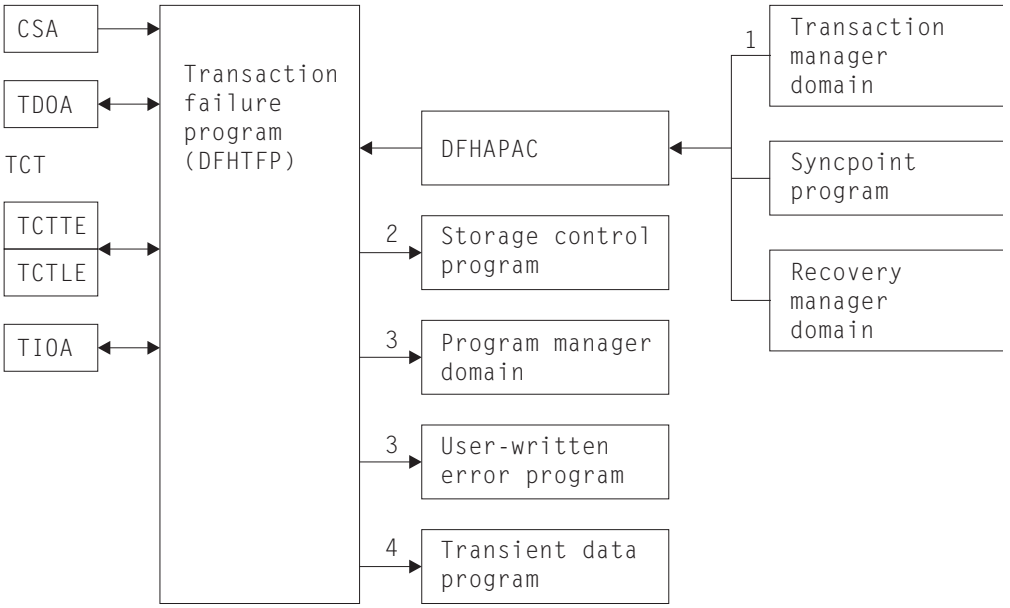


Figure 90. DFHTFP abnormal condition program interfaces

Note:

1. DFHTFP is invoked by transaction manager whenever a task is abnormally terminated. The operator ID for error messages is in the terminal control table terminal entry (TCTTE) at TCTTEOI. DFHTFP returns to transaction manager after the error message has been issued. When a task is abnormally terminated because of a stall purge condition, the stall purge count is increased by one and the transaction identifier (from the installed resource definition) is included in the error message.
2. DFHTFP communicates with storage control to obtain and release terminal input/output areas (TIOAs).
3. DFHTFP links to the user-supplied (or default) program error program by issuing a DFHPGLU LINK_URM domain call, which passes a parameter list via a COMMAREA (mapped in this case by DFHPCOM TYPE=DSECT). Any abend within a DFHPEP program results in control returning to DFHTFP unless there is an active HANDLE ABEND for this program. See “Program error program” on page 1758 for further information about the DFHPEP program.
4. DFHTFP and DFHACP both write error messages to the transient data destination, CSMT, by calling the message domain.

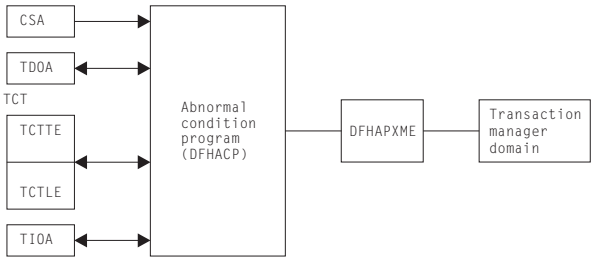


Figure 91. DFHACP abnormal condition program interfaces

Note:

1. DFHACP is invoked by transaction manager whenever an incorrect transaction code is detected.
2. DFHTFP and DFHACP both write error messages to the transient data destination, CSMT, by calling the message domain.

Modules

DFHTFP, DFHACP, DFHAPAC, and DFHAPXME

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for the abnormal condition program:

- AP 00DC, for which the trace level is AP 1.

Transaction restart program

The transaction restart program, DFHREST, is a user-replaceable program that helps you to determine whether or not a transaction is restarted. The default version of DFHREST requests a transaction restart under certain conditions; for example, if a program isolation deadlock occurs (that is, when two tasks each wait for the other to release a particular DL/I database segment), one of the tasks is backed out and automatically restarted, and the other is allowed to complete its update.

For further information about the transaction restart program, see the Troubleshooting for recovery processing. For information about how to provide your own code for DFHREST, see Writing a transaction restart program.

Design overview

During the execution of any transaction, certain temporary-storage data, intrapartition destinations, and files are protected for dynamic backout. In addition, for a transaction that is defined as restartable, the following actions take place:

- Any terminal input/output area (TIOA), command-level communication area, or terminal user area existing at task initiation is copied to the dynamic log.
- Interval control automatic initiate descriptors (AIDs) used in the task are preserved by means of deferred work elements (DWEs) until the next syncpoint.

- Data is maintained to show:
  - What terminal traffic has occurred during the task
  - Whether a syncpoint has been passed
  - Whether or not the current activation of the task is the result of a restart.

If a transaction abends, but before backout has been attempted, DFHREST may be invoked to decide whether or not the task is to be restarted. Even if DFHREST decides that the transaction can be restarted, CICS may overrule the restart, for example because of a transaction backout failure.

DFHREST is invoked by DFHXMTA passing a parameter list via a COMMAREA that is mapped by the DFHXMRSD DSECT. DFHREST should return to DFHXMTA, indicating whether or not the transaction should be restarted. If DFHREST requests a restart, and CICS does not overrule this decision, the principal facility is not released and the principal facility owner reattaches a new task to restart the transaction.

Note:

1. DFHREST can invoke CICS facilities such as file control and transient data, using the command-level interface.
2. If an error occurs while linking to, or in, the transaction restart program, the restart is not attempted for this task.
3. DFHREST runs before backout.

Control blocks

CICS supplies a description of the transaction restart program commarea, in Assembler-language, COBOL, PL/I, and C, which maps the layout of the parameter list passed between DFHXMTA and DFHREST. The parameter list contains information that helps you code your own version of DFHREST to determine whether a restart should be requested for a task.

Modules

DFHREST is a skeleton user-replaceable program that you can modify.

Exits

Global user exit points are not relevant for this function.

Trace

Trace point IDs are not relevant for this function.

Transaction Restart Statistics

CICS keeps a count of the number of times that each transaction has been restarted.

Transaction routing

Transaction routing allows one CICS system to run a transaction in another CICS system. The transaction routing facility enables a terminal operator to enter a CICS transaction code into a terminal attached to one CICS system, and start a transaction on another CICS system in a different address space in the same processing system or in another system. This topic does not contain information about transaction routing using IP interconnectivity (IPIC).

For information about IPIC support for transaction routing see Overview of transaction routing.

There are two cases of transaction routing:

- Advanced program-to-program communications (APPC); that is, LU6.2
- Non-APPC (for example, LU2).

APPC transaction routing uses much of the non-APPC function, and there is often considerable overlap between the function provided by modules for each of the two cases.

CICS transaction routing gives a detailed description of transaction routing.

Design overview

Figure 92 shows the overall design of this component.
CICS executes the CICS relay program DFHAPRT (which invokes the

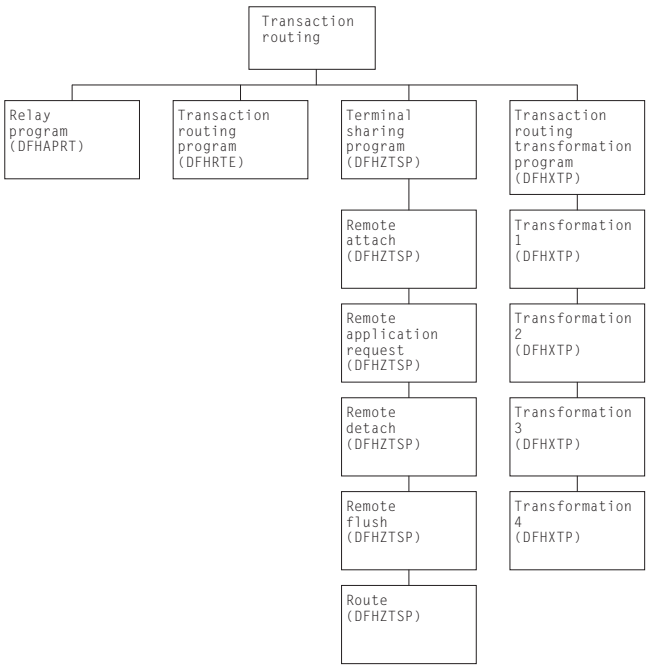


Figure 92. Transaction routing

user-replaceable dynamic transaction routing program) as follows:

- When a transaction defined with the value DYNAMIC(YES) is initiated.
- When a transaction definition is not found and CICS uses the special transaction defined on the DTRTRAN system initialization parameter.
- Before routing a remote, terminal-oriented, transaction initiated by ATI.
- If an error occurs in route selection.
- At the end of a routed transaction, if the initial invocation requests re-invocation at termination.

If CICS has been generated with the appropriate options for intercommunication, the initialization of CICS with the ISC=YES system initialization parameter specified causes the following modules to be loaded:

- DFHXTP (transaction routing data transformation program)

- DFHZCXR (which includes the DFHZTSP CSECT, the terminal sharing program).

The entry point addresses of these modules are contained in the optional features list that is addressed by CSAOPFLA in the CSA.

The rest of this section is mainly concerned with APPC transaction routing, which occurs when an APPC device is linked through an LU6.2 session to a transaction that is defined as remote.

Overview of operation in the application-owning region for APPC transaction routing

Figure 93 shows the modules in the application-owning region for transaction routing for APPC devices.

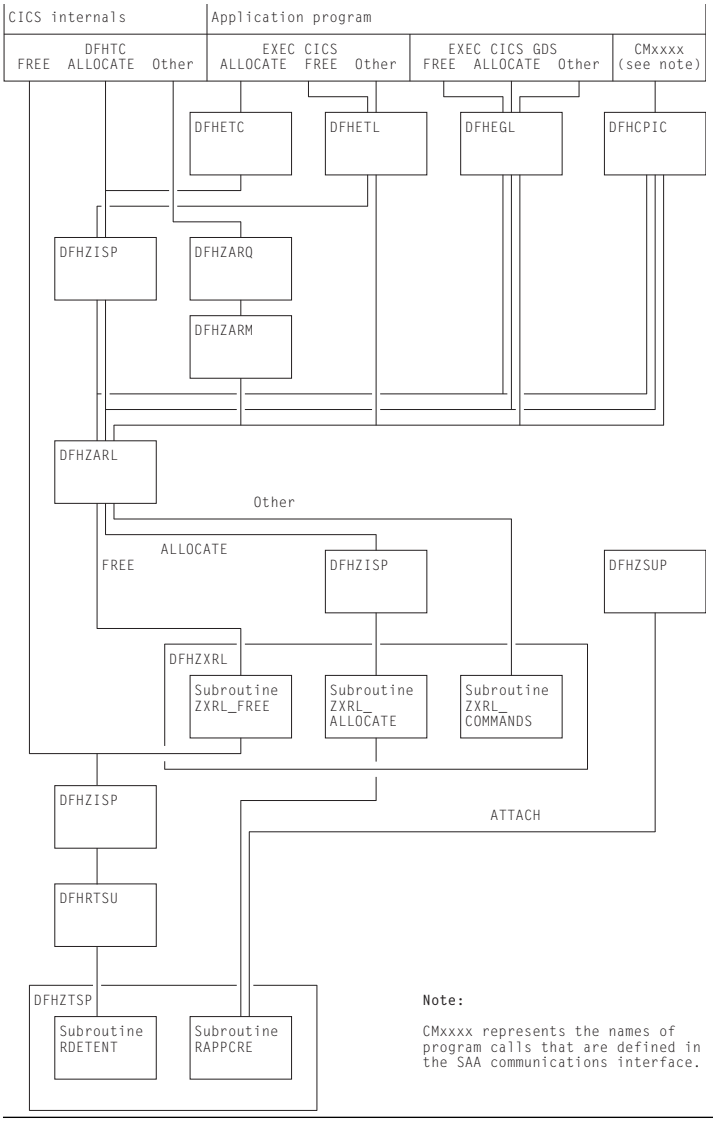


Figure 93. Transaction routing for APPC devices: modules in the application-owning region

APPC control blocks: A remote APPC device is defined in the application-owning region with a remote terminal control table system entry (or remote system entry). There are no TCT mode entries or session TCTTE entries associated with the remote system entry when it is defined.

A session with the remote APPC device is represented by a surrogate session TCTTE (or surrogate session entry). The surrogate is built dynamically when the conversation between the systems is initiated, and is deleted when the conversation terminates.

Figure 94 shows the way in which the TCT entries are related.

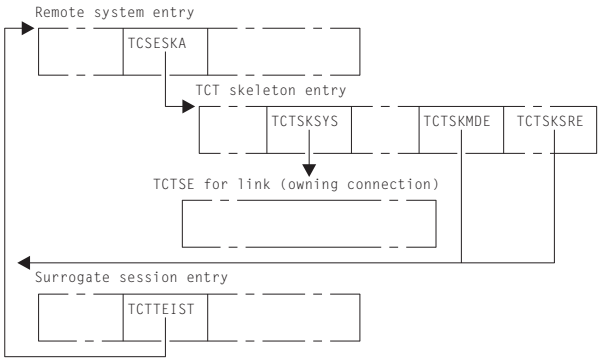


Figure 94. Transaction routing for APPC devices: TCT control-block structure in the application-owning region

Remote system entry: The remote system entry is similar to a normal system entry and, together with the TCT skeleton entry, also includes the following information:

- SYSIDNT of the terminal-owning region (TCTSKSYS)
- SYSIDNT of remote APPC device (local name) (TCTSKID)
- REMOTENAME of APPC device (SYSIDNT on terminal-owning region) (TCTSKHID)
- NETNAME of remote APPC device (TCSESID).

The remote system entry may be defined explicitly with CEDA DEFINE and INSTALL commands.

Alternatively, it is installed dynamically when the first transaction is routed from the remote APPC device. In this case, all data required to build the system entry is included in the initial ATTACH data stream from the application-owning region. No INQUIRE or INSTALL data is sent.

The remote system entry is recorded on the catalog and recovered after warm start and restart. It is located by TMP in the REMOTE domain and SYSTEM domain.

Surrogate session entry: The session between the terminal-owning region and the APPC device is represented in the application-owning region by a surrogate session entry.

The surrogate session entry is used to support the routing of commands to the APPC device, and to record security and status information for the conversation.

A surrogate session entry cannot be defined by the user; instead it is created when the conversation is initiated (by an ATTACH request from the APPC device, or an ALLOCATE request from the application-owning region), and is deleted when the conversation ends.

The surrogate session entry is not recorded on the catalog, is not accessible via TC LOCATE, and does not have an entry in the TMP index. It is not recovered after warm start or restart.

CEMT and EXEC CICS INQUIRE or SET commands cannot be used to modify a remote system entry.

DFHZXRL: This module forms a principal part of the transaction routing component for APPC devices. It passes DFHLUC macro requests issued in an application-owning region to the terminal-owning region.

All DFHLUC macro requests cause DFHZARL to be invoked. DFHZARL passes a request to DFHZXRL if the TCTTE address passed is for a surrogate session, and the request is one that DFHZXRL is known to handle (apart from ALLOCATE). ALLOCATE requests are always routed from DFHZARL to DFHZISP. DFHZISP is then responsible for calling DFHZXRL if the system from which a session is to be allocated is found to be remote. Table 92 summarizes this and shows which of the three main routines in DFHZXRL is called. ZXRL_ALLOCATE, ZXRL_COMMANDS, and ZXRL_FREE are described in "ALLOCATE processing in the application-owning region" on page 1851, "Other LU6.2 command processing in the application-owning region" on page 1853, and "FREE processing in the application-owning region" on page 1853 respectively.

Table 92. DFHZXRL's processing of DFHLUC requests

DFHLUC request	DFHZXRL's caller	DFHZXRL routine called
ALLOCATE	DFHZISP	ZXRL_ALLOCATE
ISSUE-ABEND ISSUE-ATTACH ISSUE-CONFIRMATION ISSUE-ERROR ISSUE-SIGNAL RECEIVE SEND WAIT EXTRACT-PROCESS	DFHZARL	ZXRL_COMMANDS
FREE	DFHZARL	ZXRL_FREE

The input and output for DFHZXRL is provided by means of the LUC parameter list, that is, the parameter list which is built by the DFHLUC macro. DFHZARL passes the LUC parameter list to DFHZXRL unaltered. If the LUC parameter list previously contained only the SYSID name, DFHZISP adds the address of the remote system entry to the LUC parameter list before passing it to DFHZXRL.

DFHZXRL calls routine RAPPCRE of DFHZTSP to build the surrogate TCTTE representing the session with the APPC device, and DFHZISP calls routine RDETENT to free it.

ATTACH processing in the application-owning region: The following describes how a transaction is attached in the application-owning region when the attach request has been routed from the terminal-owning region.

DFHZSUP module:

1. Issues DFHSEC TYPE=CHECK,RESTYPE=TRAN to validate transaction security against the security values associated with the intersystem link at bind time.
2. Processes the incoming attach FMH5.

   For an LU6.2 ISC connection:

   - Sets the TCTTE to indicate a mapped or unmapped conversation.
   - Validates synclevel requested in FMH5 against the value negotiated at bind time.
   - Moves the TPN from the FMH5 to the TCA extension.
   - Performs attach-time security processing, as defined by the ATTACHSEC parameter in the resource definition for the LUC CONNECTION to the terminal-owning region. This may change the security values associated with the link from the bind-time established values that were checked in step 1) to user-level values, obtained from the SNT for a userid specified in the FMH5.

   For an MRO connection:

   - Issues DFHZIRCT FN=ZSUP to extract the USERID and UOW-ID from the LU6.2 style FMH5.
   - Performs attach-time security processing, as defined by the ATTACHSEC parameter in the resource definition for the LUC CONNECTION to the terminal-owning region. This can change the security values associated with the link from the bind-time established values that were checked in step 1) to user-level values, obtained from the SNT for a userid specified in the FMH5.
   - Deletes the LU6.2-style FMH5 from the front of the data stream.
3. Issues DFHZUSRM TYPE=SET,REQUEST=ATTACH_INBOUND and DFHLUC TYPE=INIT-CALL macros to move input data into a buffer bypassing the FMH5 ATTACH header.
4. PIP processing is bypassed because PIP is never present on an attach from a terminal-owning region when transaction routing.
5. Puts the remaining data into a TIOA with a DFHTC TYPE=(READ,WAIT),NOATNI=YES.
6. Issues a DFHIS TYPE=RATT, to call DFHZTSP to build a surrogate session entry to represent the session TCTTE in the terminal-owning region.
7. Assign the security values established for the link to the surrogate, as preset security values are shipped from the terminal-owning region, and cannot be defined on the application-owning region.

   ATTACH security processing in DFHZSUP has established two SNTTEs associated with the link session:

   a. The SNTTE pointed to by TCTELSNT in the LU6.2 extension or TCTEIRSN for MRO represents link-level security values established at bind time.
   b. The SNTTE pointed to by TCTTESNT represents user-level security values established during ATTACH security processing.

   TCTTESNT is copied to the surrogate TCTTE. No provision is made for preset user security values to override the TCTTESNT value.

   Preset security values defined for the terminal session on the terminal-owning region are processed only on that system, during local attach processing. The SNTTE then associated with the local TCTTE is used to build the routed attach FMH5.

At transaction end, no SNTTEs addressed by the surrogate are deleted when the surrogate is deleted. This is done, if necessary, as part of the termination of the LINK SESSION.

Each system in a "daisy chain" imposes its own link security requirements. An intermediate system with a lower level of security would route the ATTACH with lower security (that is, no USERID or verified bit) which could cause it to be rejected by the next system in the chain.

8. Passes control to the requested application program.

DFHZTSP module:

1. Performs initialization housekeeping, checks the link TCTTE and TIOA.
2. Locates remote system entry from the TMP REMOTE domain. If not found, attaches the CITS transaction (DFHZATS) to install it.
3. Builds surrogate session TCTTE.
4. Gets a TIOA and chains it to the surrogate.
5. Issues DFHIS TYPE=XTP,XFNUM=2 to call DFHXTP.
6. Chains surrogate to TCA and Link TCTTE.
7. Copies link operator dispatching priority from the link and establishes dispatching priority for the surrogate.

DETACH processing in the application-owning region: At transaction end, routine RDETENT of DFHZTSP is called to delete the surrogate session entry. The remote system entry is not deleted, and can be used by a subsequent transaction routing request, by an ATI request, or by an ALLOCATE request issued in the application-owning region.

ALLOCATE processing in the application-owning region: A session can be allocated as a result of either of the following macro calls:

- DFHLUC TYPE=ALLOCATE
- DFHTC TYPE=ALLOCATE

The DFHLUC call invokes DFHZARL, which passes control to DFHZISP, the module that handles allocation and freeing of sessions. The DFHTC call invokes DFHZISP directly.

DFHZISP locates the TCTSE for the system identified on the ALLOCATE request.

The request is routed to DFHZXRL if the following conditions hold:

- The system is LU6.2
- The system is remote
- DFHZISP was called as a result of a DFHTC TYPE=ALLOCATE request (which is the case when DFHZISP is called from DFHZARL).

The address of the remote TCTSE is inserted in the parameter list passed to DFHZXRL.

If a Privileged Allocate request is made, the transaction abends, because the request is not permitted for a remote system.

DFHZXRL module: For an ALLOCATE request, control passes to subroutine ZXRL_ALLOCATE which establishes a session between the application-owning region and the alternate facility, and builds a surrogate session TCTTE.

Subroutine ZXRL_ALLOCATE:

1. Checks that the parameter list contains the TCTSE address for the remote LU6.2 system.
2. Obtains the address of the TCTSE of the system to which the LU6.2 commands are to be routed.
3. Allocates a session to the terminal-owning region.

   The connection between the terminal-owning region and application-owning region which supports remote alternate facilities may be an LU6.2 ISC connection or an MRO connection. Subroutine ZXRL_ALLOCATE allocates the session using a DFHTC TYPE=ALLOCATE macro call that can allocate a session on either type of connection.

   The default profile DFHCICSR is used; this may specify the modename for an LU6.2 connection. The modename specified on the EXEC CICS ALLOCATE is not used here, but is shipped to the terminal-owning region where it is used to allocate an LU6.2 session between the terminal-owning region and the APPC device.

   The queuing option (NOQUEUE|NOSUSPEND) specified on the ALLOCATE request by the caller is used when the DFHTC TYPE=ALLOCATE macro call is issued for the connection. If NOQUEUE is not specified, the request may also be queued when it is issued in the terminal-owning region. If a session failure occurs during this period, the transaction in the application-owning region and the relay transaction in the terminal-owning region abend.

   If a session between the application-owning region and terminal-owning region cannot be allocated:

   - When the failure is due to CICS logic, corruption of CICS storage, or incorrect resource definition by the user, the transaction abends.
   - When the failure is due to other conditions (such as session failure or 'SYSBUSY'), an appropriate return code is passed to the caller.

     The return code is handled so as to minimize the differences between local and remote APPC devices as seen by the user of the DFHLUC interface. The actions available are:

     – Where the condition could be encountered with a local terminal, reflect the return code to the caller in LUCRCOD2 and LUCRCOD3 with LUCESYSI (X'01') in LUCRCOD1.
     – Where the condition would not occur with a local terminal, reflect a different return code to the caller.
4. Issues a DFHIS TYPE=XTP,XFNUM=3 macro call that invokes a stream that is passed to the terminal-owning region.
5. Issues a DFHTC TYPE=(WRITE,WAIT,READ),FMH=YES macro call to send the request to the terminal-owning region and receive the response.
6. Issues a DFHIS TYPE=RALL that invokes DFHZTSP to build a surrogate session TCTTE, then chains the link session TCTTE and the surrogate session TCTTE together.
7. Issues a DFHIS TYPE=XTP,XFNUM=2 macro call that invokes DFHXTP to unwrap the response from the terminal-owning region and update the surrogate session TCTTE and the parameter list created by the DFHLUC macro.
8. Examines the return codes in the response:
   - If the request has been successful, returns the surrogate session TCTTE address to the caller.
   - If the request has not been successful, issues a DFHIS TYPE=RDET macro call to free the surrogate session TCTTE.

FREE processing in the application-owning region: One of the following macro calls is made in the application-owning region to request that a surrogate session TCTTE should be freed:

- DFHLUC TYPE=FREE
- DFHTC TYPE=FREE

The DFHLUC TYPE=FREE call invokes DFHZARL, which passes control to DFHZXRL; and subroutine ZXRL_FREE in DFHZXRL is then called to issue a DFHTC TYPE=FREE request against the surrogate. The DFHTC TYPE=FREE call invokes DFHZISP.

DFHZISP:

1. Bypasses security processing (sign-off) for a surrogate session entry, because the sign-off is performed for the link.
2. Issues the DFHIS TYPE=RDET macro that calls DFHZTSP to free the surrogate and link TCTTEs.

Other LU6.2 command processing in the application-owning region: Most SAA communications calls, EXEC CICS GDS commands, and EXEC CICS commands relating to LU6.2 sessions cause a call to DFHZARL using the DFHLUC macro.

The EXEC CICS SYNCPOINT, EXEC CICS SYNCPOINT ROLLBACK, and EXEC CICS (GDS) ISSUE PREPARE commands are handled under the control of the syncpoint program, which uses DFHLUC macro requests to send syncpoint flows on LU6.2 sessions, and DFHTC macro calls to end any dangling conversations.

DFHTC macro requests: DFHTC macro requests may be issued against surrogate session TCTTEs. Unlike requests for other surrogate TCTTEs, which are passed to DFHZTSP, DFHZARQ handles these requests in the same way as other requests against LU6.2 sessions: they are passed to DFHZARM which in turn calls DFHZARL. Within DFHZARL, requests are handled in a similar way to those initiated by the DFHLUC macro.

DFHLUC requests: DFHLUC requests are passed to DFHZARL: when the session is a surrogate, the request is passed to DFHZXRL (routine ZXRL_COMMANDS).

DFHZXRL module: Input to routine ZXRL_COMMANDS in DFHZXRL is the application command in the form of a DFHLUC macro call parameter list.

1. ZXRL_COMMANDS normally wraps up the command to be shipped and relevant TCTTE fields by calling a transformer routine in DFHXTP.

   However, if the first syncpoint flow has been received, then:

   - Application requests ISSUE-ERROR and ISSUE-ABEND are sent unwrapped on the link session.
   - All other requests are rejected with a state error.
2. ZXRL_COMMANDS tests the state of its link with the terminal-owning region (this may not be the same as the state of the application):

   If it finds that it is in 'RECEIVE' state, it issues a DFHTC TYPE=(READ,WAIT) in order to receive the change direction (CD) indicator from the terminal-owning region. Except during syncpoint processing, however, the session is normally in 'SEND' state when a command is issued.
3. ZXRL_COMMANDS then sends the wrapped-up request to the remote system using the DFHTC macro. To reduce the number of flows when the command may result in the termination of the conversation, the following rules are applied for both MRO and ISC links:

- If the application command is SEND LAST WAIT and the application program is in 'SEND' state, the command is sent using a DFHTC TYPE=(WRITE,LAST) macro.
- If the application command is WAIT and the application program is in 'FREE PENDING AFTER SEND LAST' state, the command is sent using a DFHTC TYPE=(WRITE,LAST) macro.
- If the end bracket (EB) indicator has been sent to the terminal-owning region all other commands result in a state error return code.

In other cases and when the link between the terminal-owning region and application-owning region is MRO, ZXRL_COMMANDS issues a DFHTC TYPE=(WRITE,WAIT,READ).

However, when the link is LU6.2, the following additional rules are applied in order to exploit the buffering provided by LU6.2:

- When the application's command is a SEND and the application is in 'SEND' state ZXRL_COMMANDS, issues a DFHTC TYPE=(WRITE,WAIT) macro to send the request without waiting for a response.
- When the application's command is a SEND and the application is not in 'SEND' state ZXRL_COMMANDS, issues a DFHTC TYPE=(WRITE,WAIT,READ) so that it can get the state error back from the remote system immediately.
- For all other commands, including SEND INVITE and so on, ZXRL_COMMANDS issues a DFHTC TYPE=(WRITE,WAIT,READ).

4. ZXRL_COMMANDS receives the response to its DFHTC macro call. This may be:
   - An ATNI or ATND abend. ZXRL_COMMANDS frees the link session and returns 'TERMERR' to the application.
   - 'SIGNAL', which is used by the terminal-owning region when it is in 'RECEIVE' state to indicate to the application-owning region that there is an abnormal response pending.

     ZXRL_COMMANDS issues a DFHTC TYPE=(WRITE,WAIT,READ) to send the change direction indicator and get the abnormal response from the terminal-owning region.
5. When the DFHTC macro included a READ, and the request was succesfully processed, ZXRL_COMMANDS checks for a wrapped reply from the terminal-owning region, and calls DFHXTP to unwrap the reply. When the resulting DFHLUC parameter list indicates SYNCPOINT or SYNCPOINT ROLLBACK, and the link is an MRO connection, ZXRL_COMMANDS issues a DFHTC TYPE=READ, because there is a SYNCPOINT or ROLLBACK flow pending.

   When there is no wrapped reply, ZXRL_COMMANDS checks for SYNCPOINT ROLLBACK received (the only possibility under these circumstances).

LU6.2 daisy-chaining considerations:

There is no special-case code to distinguish between the terminal-owning region and an intermediate system.

When DFHZXRT has interpreted a request received from the application-owning region, it issues the LU6.2 service request (DFHLUC) macro call with the parameter list that was created in the application-owning region. The macro generates a call to DFHZARL. If the TCTTE is a surrogate, which is the case in an intermediate system, control passes to DFHZXRL as described in "DFHZXRL module" on page 1853.

Overview of operation in the terminal-owning region for APPC transaction routing

Figure 95 shows the modules in the terminal-owning region for transaction routing for APPC devices.

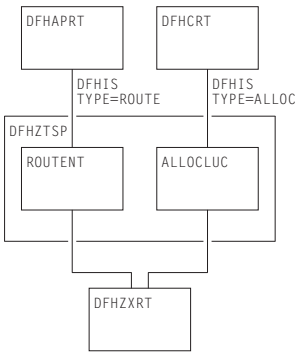


Figure 95. Transaction routing for APPC devices: Modules in the terminal-owning region

In the terminal-owning region, operation is under the control of a relay program. When transaction routing is initiated from the APPC device, the relay program is DFHAPRT (which is also used for non-APPC devices). When transaction routing is initiated by an ALLOCATE request in the application-owning region, the relay program is DFHCRT. Both relay programs call DFHZTSP, which calls DFHZXRT.

When an APPC device initiates a conversation with an application in the application-owning region, relay program DFHAPRT is started in the terminal-owning region. It calls the ROUTENT routine of DFHZTSP, which allocates a session to the application-owning region and starts the requested transaction there (see "ATTACH processing in the terminal-owning region").

When an application running in the application-owning region initiates a conversation with a remote APPC device by issuing an ALLOCATE request, the DFHCRT relay program is started in the terminal-owning region. It calls the ALLOCLUC routine of DFHZTSP which allocates a session to the APPC device (see "Program error program" on page 1758).

After a conversation has been started by either method, the LU6.2 commands passed from the application-owning region are processed by DFHZXRT, which issues the LU6.2 service request (DFHLUC) macro with an appropriate parameter list against the APPC device.

ATTACH processing in the terminal-owning region: The following flow describes the steps involved in routing a transaction from an APPC device across an LU6.2 intersystem link.

DFHZSUP module:

1. Processes the incoming FMH5 from the terminal, as follows:
   - It sets TCTTE to indicate mapped or unmapped conversation.
   - It validates synclevel requested in FMH5 against the value negotiated at bind time.
   - It moves the TPN from the FMH5 to the TCA extension.
   - It performs attach-time security processing, as defined by the ATTACHSEC parameter in the resource definition for the APPC device (or

CONNECTION). This may change the security values associated with the terminal from the default link-level values to user-level values, obtained from the SNT for a user who is signed on.

2. Checks transaction security code against new security levels developed during ATTACH security processing in the previous step.
3. Issues DFHSEC TYPE=CHECK,RESTYPE=TRAN to validate transaction security against the security values associated with the terminal (and with the user, if signed on).
4. Issues DFHZUSRM TYPE=SET,REQUEST=ATTACH_INBOUND and DFHLUC TYPE=INIT-CALL macros to move input data into a buffer bypassing the FMH5 ATTACH header.
5. If PIP is present, builds a new TCA extension and moves the PIP data into it by issuing a DFHLUC TYPE=RECEIVE (which also causes the PIP data to be deleted from the buffer).
6. Puts remaining mapped data into a TIOA with a DFHTC TYPE=(READ,WAIT),NOATNI=YES.
7. Issues DFHPC TYPE=XCTL to the relay program DFHAPRT.

DFHAPRT module:

1. Drives the dynamic routing exit if the transaction has been defined as dynamic.
2. Sets up the DFHISCRQ parameter list with remote sysid and tranid.
3. Recognizes that the principal facility is an APPC device.
4. Issues DFHIS macro to invoke DFHZTSP.

DFHZTSP module:

1. If the transaction has been defined with an associated TRPROF, the profile named is located with a DFHKC CTYPE=PROFLOC; otherwise the default DFHCICSS profile is used.
2. Issues DFHTC TYPE=ALLOCATE,REQUID=CSRR to allocate a session to the remote system using the profile identified in step 1.
3. Flags the returned TCTTE as a relay link and puts the remote sysid into TCTESYID in the terminal TCTTE. If the LINK TCTTE status is 'COLD', issues DFHTC CTYPE=CATALOG.
4. Sets up the transformer parameter list (DFHXTSTG) to indicate ATTACH FMH5 required, COLD or not COLD, and transaction routing for an APPC device, passing the tranid, user TCTTE, and link TCTTE.
5. Issues DFHIS TYPE=XTP,XFNUM=1 to call the transformer program, DFHXTP, to build the data. (See "Transformer program (DFHXTP)" on page 1859.)
6. Issues DFHTC TYPE=(WRITE,WAIT,READ) against the link to route the ATTACH request to the application-owning region. This causes DFHZARM (when the link is ISC) or DFHZIS2 (when the link is MRO) to add an LU6.2 FMH5 preceding the LU6.1 FHM5 built by XTP. This contains security data required to validate the request at the application-owning region.

ALLOCATE processing in the terminal-owning region:

DFHCRT module: Transaction CXRT (program DFHCRT) is started in the terminal-owning region when the attach FMH5 is received from the application-owning region

Program DFHCRT:

1. Checks that the principal facility of the task is an ISC or MRO session.

If not, and if it is a terminal, a message is written to the facility, and the transaction terminates.

2. Issues DFHIS TYPE=ALLOC macro which calls DFHZTSP.

DFHZTSP module: The ALLOCLUC routine of DFHZTSP is invoked when the DFHIS TYPE=ALLOC macro is issued. This routine is called with input from the application-owning region in a TIOA.

Routine ALLOCLUC:

1. Issues DFHIS TYPE=XTP,XFNUM=4 which updates the TCTTE and builds a parameter list of the type created by the DFHLUC macro.
2. Verifies that the parameter list contains an ALLOCATE request (the only valid request at this stage). If it does not, the transaction abends.
3. Issues a DFHLUC MF=E macro with the supplied parameter list.
4. If the request is successful, DFHZTSP:
   a. Issues DFHIS TYPE=XTP,XFNUM=1 which wraps the updated TCTTE and DFHLUC parameter list ready for transmission to the application-owning region.
   b. Issues a DFHTC TYPE=(WRITE,WAIT,READ) against the session with the application-owning region.
   c. Passes control to DFHZXRT. The TIOA received with the preceding DFHTC request should contain data for one of the requests that DFHZXRT handles.
5. If the request is unsuccessful, DFHZTSP:
   - Issues DFHIS TYPE=XTP,XFNUM=1 which wraps the updated TCTTE and DFHLUC parameter list ready for transmission to the application-owning region.
   - Issues DFHTC TYPE=(WRITE,LAST) to send the response to the application-owning region.
   - Frees the session with the application-owning region.

FREE processing in the terminal-owning region: When an end-bracket has flowed from the application-owning region to the terminal-owning region as a result of an application command (for example, EXEC CICS SEND LAST), and the corresponding command has been issued in the terminal-owning region against the terminal, DFHZXRT issues a DFHLUC TYPE=FREE macro against the terminal, and a DFHTC TYPE=FREE macro against the link to the application-owning region.

Other LU6.2 command processing in the terminal-owning region: DFHZXRT is called by DFHZTSP following a DFHTC TYPE=(WRITE,WAIT,READ) macro. The reply received from the application-owning region is processed as follows:

1. If an application request has been received, DFHZXRT:
   - Calls DFHXTP to unwrap the application program's request
   - Issues the DFHLUC macro call with the parameter list created in the application-owning region
   - Calls DFHXTP to wrap the response to the DFHLUC macro
   - Sends the response to the application-owning region.

     Normally the wrapped terminal response is sent to the application-owning region with a DFHTC TYPE=(WRITE,WAIT,READ) macro. However, there are exceptions:

- If the response to the DFHLUC macro call is a request for SYNCPOINT ROLLBACK, DFHZXRT sends the wrapped terminal response with a DFHTC TYPE=WRITE macro and then issues a DFHSP TYPE=ROLLBACK command.
- If the response to the DFHLUC macro call is a request for SYNCPOINT, DFHZXRT sends the wrapped terminal response with a DFHTC TYPE=WRITE macro and then issues a DFHSP TYPE=PREPARE against the link.

  The response to the macro is processed in the same way as when a SYNCPOINT request is received from the application, and issued to the terminal, except that the roles of the terminal and link are reversed.
- If the session to the terminal has been freed by an application command, DFHZXRT sends the wrapped terminal response with a DFHTC TYPE=(WRITE,LAST) macro.
- When the session to the application-owning region is in 'RECEIVE' state, normally DFHZXRT issues a DFHTC TYPE=READ to get the next request from the application.

  However, if the link between the terminal-owning and application-owning regions is LU6.2, and the response to the DFHLUC macro issued to the terminal indicates that the terminal has issued one of ISSUE_SIGNAL, ISSUE_ERROR, ISSUE_ABEND, or SYNCPOINT_ROLLBACK, DFHZXRT issues an ISSUE_SIGNAL against the link with the application-owning region to notify the application-owning region that the terminal-owning region wants to send. It then issues a series of DFHTC TYPE=READ macros until it receives the change of direction indicator.

  The data is processed in the normal way when 'SIGNAL' is received from the terminal. In the other cases, that is, if a negative response is received from the terminal, the data from the application-owning region is purged.

  After the change direction indicator is received, DFHZXRT sends the response to the application-owning region, ISSUE_SIGNAL and ISSUE_ERROR are sent using a DFHTC TYPE=(WRITE,WAIT,READ) macro, ISSUE_ABEND is sent using a DFHTC TYPE=(WRITE,LAST) macro, and SYNCPOINT_ROLLBACK is sent using a DFHTC TYPE=WRITE macro.
- If the response from the terminal was 'ROLLBACK', a DFHSP TYPE=ROLLBACK macro is issued.

2. If a syncpoint request has been received, DFHZXRT:
   - Issues a DFHLUC TYPE=ISSUE-PREPARE macro against the terminal TCTTE.
   - Checks the terminal's response:

     If the terminal response indicates that a SYNCPOINT or BACKOUT request was issued, DFHSPP is called.

     If the terminal response indicates that the terminal issued a SEND_ERROR request, DFHZXRT issues a DFHTC CTYPE=ISSUE_ERROR macro followed by a DFHTC TYPE=(WRITE,WAIT,READ) macro against the link session.

     If the terminal response indicates that the terminal issued DEALLOCATE(ABEND), DFHZXRT issues a DFHTC CTYPE=ISSUE_ABEND macro against the link session. It then frees the link with the application-owning region and returns.
3. If a syncpoint rollback request has been received, DFHZXRT issues a SYNCPOINT ROLLBACK request.

When DFHZXRT detects that EB has flowed on both the session with the terminal and the session with the application-owning region, it issues DFHTC TYPE=FREE on both and returns.

Transformer program (DFHXTP)

The terminal-sharing data-transformation program, DFHXTP, constructs and interprets the data streams flowing between terminal-owning and application-owning regions, for both APPC and non-APPC transaction routing environments.

It does this by using four transformers. These either wrap this data from the surrogate TCTTE (in the AOR) or the real TCTTE (in the TOR) into the link TCTTE's TIOA, or they unwrap this data from the link TCTTE's TIOA into the surrogate or real TCTTE.

The transformers work in matching wrap and unwrap pairs. Transformer 1 wraps any data to be sent from a TOR to an AOR, which is then unwrapped in the AOR by transformer 2. Transformer 3 wraps any data to be sent from an AOR to a TOR, which is then unwrapped in the TOR by transformer 4. Figure 96 shows this process.

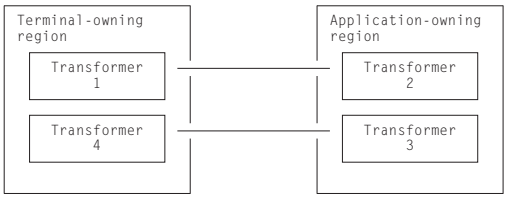


Figure 96. DFHXTP transformer operations

The transformer program is capable of shipping data from the TCTTE and the following control blocks that are chained off the TCTTE:

- The TCTTE extension, chained off TCTTETEA in the TCTTE.
- The terminal partition extension, chained off TCTTETPA in the TCTTE BMS extension.
- The TCTTE user extension, chained off TCTTECIA in the TCTTE.
- The SNTTE, chained off TCTTESNT in the TCTTE.
- The DFHLUC parameter list, and fields chained off it.

  Note that because this field is not chained off the TCTTE but is in LIFO, its address is passed as a parameter to the transformer program.
- The TCA extension for LU6.2 communication.
- Fields from the terminal control table system entry (TCTSE), chained off TCTTEIST in the TCTTE.
- Fields from the terminal control table mode entry (TCTME), chained off TCTTEMOD in the TCTTE.
- The data interchange block (DIB), chained off TCTEDIBA in the TCTTE.

The fields to be shipped are defined in tables in the transformer program.

There is special-case code to deal with fields that cannot be processed by the table-driven code.

For the transaction routing of LU6.2 commands, DFHXTP must ensure that the data stream built for transmission contains all the information relevant to support the issuing of a DFHLUC macro request on the remote system. This information consists primarily of:

- The DFHLUC parameter list
- Any data addressed by the parameter list
- The conversation state machine (TCTEUSRS in DFHTCTZE) in the TCTTE
- TCTTE fields required to build the surrogate TCTTE, in particular:
  - The synclevel supported by the terminal
  - The information returned to the application by the EXTRACT PROCESS command.

Data streams for transaction routing: Figure 97 shows the types of transaction-routing data streams.

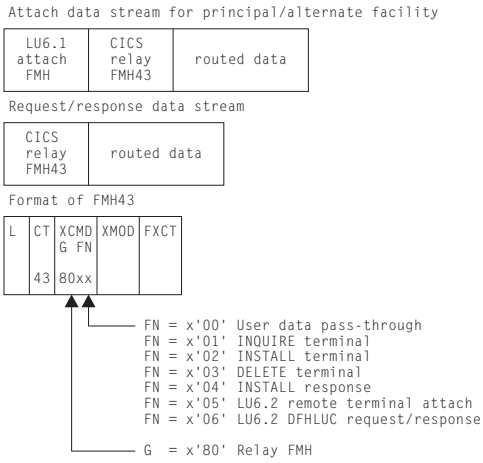


Figure 97. Transaction-routing data streams

The transformer builds four types of data stream for transaction routing:

1. Attach data stream for principal facility
   - Built by transformer 1
   - Shipped from TOR to AOR
   - Unwrapped by transformer 2
   - Contains an LU6.1 attach FMH (FMH5)
   - For LU6.2, the routed data does not contain a DFHLUC parameter list.
2. Attach data stream for alternate facility
   - Built by transformer 3
   - Shipped from AOR to TOR
   - Unwrapped by transformer 4
   - Contains an LU6.1 attach FMH (FMH5)
   - For LU6.2, the routed data contains a DFHLUC parameter list.
3. DFHLUC request data stream
   - Built by transformer 3
   - Shipped from AOR to TOR
   - Unwrapped by transformer 4

- For LU6.2, the routed data contains a DFHLUC parameter list.

4. DFHLUC response data stream
   - Built by transformer 1
   - Shipped from TOR to AOR
   - Unwrapped by transformer 2
   - For LU6.2, the routed data contains a DFHLUC parameter list.

Note: The first transformer request for remote alternate facilities is to transformer 3, and not to transformer 1. This is because the same transformers are used whether transaction routing is initiated in the terminal-owning region or in the application-owning region.

An LU6.1 attach FMH5 is used when a transaction is to be started in the system to which the request is sent. CSRR is specified as the return process to indicate the use of transaction routing. In the case of routing to the application-owning region, the transaction is the user transaction; in the case of routing to the terminal-owning region, the transaction is the CXRT relay transaction.

Transaction-routed data format: Figure 98 shows the format of the data stream passed between a TOR and an AOR to provide transaction routing from any supported device.

The fields that are shipped depend principally on the type of terminal and on other parameters, as follows:

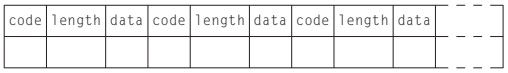


Figure 98. Routed data format

The length field in Figure 98 depends upon whether the field type is described in the table that follows as being V (Variable), F (Fixed), or U (Undefined). A V field is 2 bytes in length, an F field is 1 byte, and U indicates a variable that is no longer wrapped or unwrapped if it is encountered.

Table 93 shows the various data fields that may appear in a transaction routing data stream, together with their codes and field types.

Table 93. Transaction routing data stream. Built by the terminal sharing transformer (DFHXTP).

Code	Hex	Type	DSECT	Field	Description
1	01	V		XTPCDTC1	TC request bytes or attach start code
2	02	V		XTPCDOPC	Operator class
3	03	V		XTPCDTUA	TCTTE user area
4	04	V		XTPCDTIA	Terminal I/O area
5	05	V		XTPCDCMA	COMMAREA
6	06	V		XTPCDLPS	Terminal partition set
7	07	V		XTPCDPLM	Page LDC mnemonic
8	08	V		XTPCDPGD	Page data
9	09	V		XTPCDRQI	Request ID
10	0A	V		XTPCDETI	Error terminal ID
11	0B	V		XTPCDETL	Error terminal LDC
12	0C	V		XTPCDMCF	Message control flags
13	0D	V		XTPCDTTL	Message title

Table 93. Transaction routing data stream (continued). Built by the terminal sharing transformer (DFHXTP).

Code	Hex	Type	DSECT	Field	Description
14	0E	V		XTPCDRTT	Route target ID: netname.termid.ldc.opid
15	0F	V		XTPCDCPS	Application partition set
16	10	F	DFHTCTTE	TCTTEAID	Automatic initiate descriptor
17	11	F	DFHTCTTE	TCTTECAD	Cursor address
18	12	F	DFHTCTTE	TCTESIDO	Outbound signal data
19	13	F	DFHTCTTE	TCTESIDI	Inbound signal data
20	14	F	DFHTCTTE	TCTE32SF	Screen size attributes
21	15	F	DFHTCTTE	TCTTEFX	Transparency attributes
22	16	F	DFHTCTTE	TCTTEBMN	Map set name
23	17	F	DFHTCTTE	TCTTECRE	Request completion extension
24	18	F	DFHTCTTE	TCTTECR	Request completion analysis
25	19	F	DFHTCTTE	TCTTEDES	TCAM destination name
26	1A	F	DFHTCTTE	TCTTETM	Terminal model number
27	1B	F	DFHTCTTE	TCTTETID	Teller identification for 2980
28	1C	F	DFHTCTTE	TCTTEOI	Operator identification
29	1D	F	DFHTCTTE	TCTTEEDF	EDF mode
30	1E	F	DFHTCTTE	TCTTETC	Nominated transaction
31	1F	F	DFHTCTTE	TCTTETS	Terminal status
32	20	U	DFHSNTTE	SNTESSF	Userid
33	21	F	DFHTCTTE	TCTEASCZ TCTEASCL TCTEASCC	Alternate screen size attributes
34	22	F	DFHTCTTE	TCTE32EF TCTE32E2	3270 extended feature flags
35	23	F	DFHTCTTE	TCTETXTF	3270 text feature flag
36	24	F	TCTTETTE	TCTEAPGL TCTEAPGC	Alternate page size
37	25	F	DFHTCTTE	TCTECSG1 TCTECSG2	Coded graphic character set identifiers
38	26	F	DFHTCTTE	TCTEUSRS	LU6.2 conversation state machine
39	27	F	TCTTELUC	TCTECVT	LU6.2 conversation type (mapped or unmapped)
40	28	F	TCTTELUC	TCTESPL	LU6.2 syncpoint level
41	29	F	DFHTCTTE	TCTESPSA	Additional syncpoint flags
42	2A	F	TCTTELUC	TCTEIAHB	Attach FMH indicator
43	2B	F	DFHTCTSE	TCSESID	NETNAME of APPC device
44	2C	U	DFHSNTTE	SNTENLS	User's national language
45	2D	F	DFHTCTTE	TCTENLS	National Language Support Code
46	2E	F	DFHTCTTE	TCTESCFL	Security flag
47	2F	F	DFHTCTTE	TCTEITRS	Trace flags
48	30	F	DFHTCTME	TCMEMODE	Mode group name
49	31	F	DFHTCTTE	TCTTENLI	National language in use
50	32	F	TCTTELUC	TCTELUC1	LUC flag byte 1
51	33	F	DFHTCTTE	TCTESSPL	Synclevel of link
53	35	F	DFHTCTTE	TCTEVTP	Send mode/receive mode
54	36	F	DFHTCTTE	TCTTEIO	Task to be initiated
55	37	F	DFHLFS	PRESETC	Preset userid
56	38	F	TCTTETTE	TCTTEFMB	Outbound formatting status
57	39	F	DFHTCTTE	TCTEUCTB	UCTRAN = YES
58	3A	F	DFHTCTTE	TCTETSU3	UCTRAN = TRANID
63	3F	F	DFHTCTTE	TCTTETT	Terminal type code

Table 93. Transaction routing data stream (continued). Built by the terminal sharing transformer (DFHXTP).

Code	Hex	Type	DSECT	Field	Description
64	40	F	DFHLUCDS	LUCOPN0 LUCOPN1 LUCOPN2 LUCOPN3	LUC request codes
65	41	F	DFHLUCDS	LUCRCODE	LUC request error feedback
66	42	F	DFHLUCDS	LUCSDBLK	LUC conversation feedback
67	43	F	DFHLUCDS	LUCNSYS	System name for LUC Allocate
68	44	F	DFHLUCDS	LUCMODNM	Modename for LUC Allocate
69	45	F	DFHLUCDS	LUCMSGNO	Message number for LUC Abend and Error
70	46	F	DFHLUCDS	LUCSENSE	Sense code for LUC Abend and Error
71	47	F	DFHLUCDS	LUCRQCON	Conversation type for LUC Issue Attach
72	48	F	DFHLUCDS	LUCRQSYN	Syncpoint level for LUC Issue Attach
73	49	F	DFHLUCDS	LUCFTPNL LUCFTPN	TPN for LUC Issue Attach
74	4A	F	DFHLUCDS	LUCPIP	PIP indicator for LUC Issue Attach
75	4B	F	DFHLUCDS	LUCTAREL	Maximum receivable length for LUC Receive
76	4C	F	DFHLUCDS	LUCMGAL	Mode group name of allocated session
90	5A	F	DFHDIBDS	DIBSENSE	DIB system/user sense data
128	80	V		XTPCDZIR	ZC install response
129	81	V		XTPCDZBP	ZC builder parameter set
130	82	V		XTPCDZIM	ZC install message set
131	83	V		XTPCOPCL	Opclass in routed message
132	84	V		XTPCDPNM	Program name for ISSUE LOAD
133	85	V		XTPLUCSD	Message text for LUC Send
134	86	V		XTPLUCRD	Message text for LUC Receive
135	87	V		XTPLUTCX	TCA extension for LU6.2
136	88	V		XTPLUMSG	Message text for LUC Issue Abend or Issue Error
137	89	V		XTPIPASS	Issue Pass
138	8A	V		XTPLDATA	Logon Data
139	8B	V		XTPRETC	Issue Pass Return Code
140	8C	V		XTPLMOD	Issue Pass Logmode

Control blocks

Relay transaction control blocks

To support transaction routing, the relay transaction owns two TCTTEs; see Figure 99 on page 1864. One TCTTE is for the terminal, the other is for the link to the user transaction. The link TCTTE has bit TCTERLT in field TCTETSU set on, to indicate that it is being used by the relay transaction.

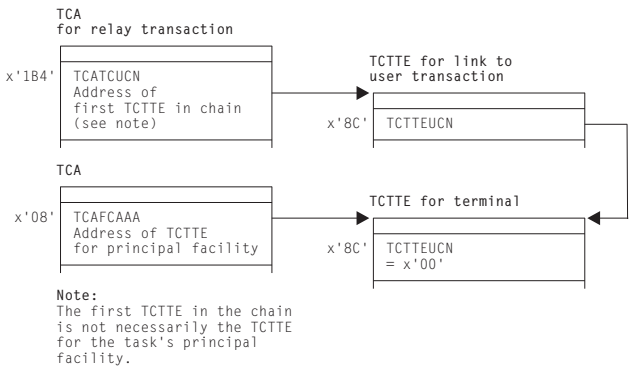


Figure 99. Control blocks associated with the relay transaction

User transaction control blocks

The user transaction owns two or more TCTTEs; see Figure 100 on page 1865. One TCTTE is always present for the link to the relay transaction, and another TCTTE, called the surrogate TCTTE, represents the terminal TCTTE in the relay transaction address space. Field TCTTERLA in the surrogate TCTTE contains the address of the TCTTE for the link to the relay transaction. Bit TCTESUR (in field TCTETSU) set on indicates that the TCTTE is for a surrogate terminal. The link TCTTE has bit TCTERLX in field TCTETSU set on, to indicate that it is being used as a relay link.

If the user transaction executes CICS functions that are shipped to another address space or processing system, one TCTTE is chained off from the TCA for each different address space or processing system.

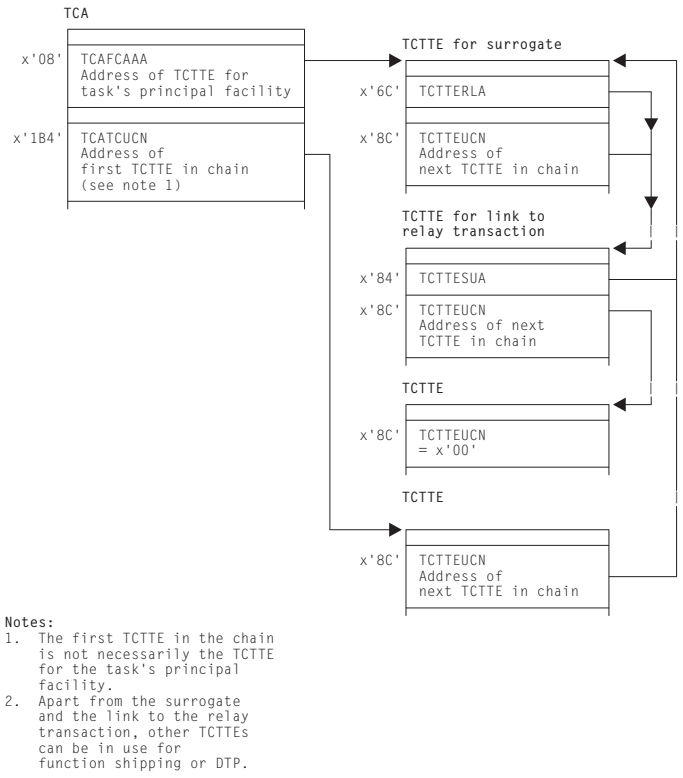


Figure 100. Control blocks for the user transaction (non-APPC device)

Modules

The principal modules associated with transaction routing are as follows:

DFHAPRT
is the relay program for non-APPC devices, and for APPC devices when the device initiates a transaction by sending an attach FMH5 to CICS.

DFHCRT
is the relay program for APPC devices when CICS sends an attach FMH5 to the device.

DFHRTSU
is the program which maintains the state of a surrogate APPC session during syncpoint

DFHXTP
is the data transformation program for terminal sharing. It constructs and interprets data streams flowing between terminal-owning and application-owning regions, for both APPC and non-APPC transaction routing environments.

DFHZTSP
is the terminal sharing program. It is used by transaction routing for devices of all types, exclusively so for non-APPC devices.

DFHZXRL
runs in the application-owning region to route APPC requests to the terminal-owning region.

DFHZXRT
runs in the terminal-owning region to receive APPC requests from the application-owning region, and issue them to the APPC device.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:

- AP DBxx (DFHXTP), for which the trace level is IS 1.
- AP 08xx (DFHCRT, DFHZXRL, and DFHZXRT), for which the trace levels are IS 1, IS 2, and Exc.

Transient data control

Transient data control provides an optional queuing facility for managing data being transmitted between user-defined destinations (I/O devices or CICS tasks). This function facilitates data collection.

Design overview

The transient data program provides a generalized queuing facility enabling data to be queued (stored) for subsequent internal or offline processing. Selected units of information can be routed to or from predefined symbolic queues. The queues are classified as either **intrapartition** or **extrapartition**.

Intrapartition queues

Intrapartition queues are queues of data, held in a direct-access data set, for eventual input to one or more CICS transactions. Intrapartition queues are accessible only by CICS transactions within the CICS address space. Data directed to or from these internal queues is called intrapartition data. It can consist of variable-length records only.

An intrapartition queue is mapped onto one or more control intervals in the intrapartition data set. The control intervals are allocated to a queue as records are written and freed automatically as they are read or as the queue is deleted.

Examples of the data queued for intrapartition processing are:

- Transactions that require processes to be performed serially, not concurrently. An example of this type of process is one in which pending order numbers are to be assigned.
- Data to be used in a data set (file) update that could pass through the queue to allow the data to be applied in sequence.

Recovery of intrapartition transient data queues: Following abnormal system termination, intrapartition queues defined as recoverable by the user can be restored. Recovery is accomplished by reconstructing the queues from catalog data and from log records written automatically by CICS during normal execution. Two types of recovery are possible: **physical** and **logical**.

Physical recovery of intrapartition transient data queues: Physically recoverable transient data queues are restored to the state they were in when the system terminated abnormally. A physically recoverable transient data queue is not backed out if it has been updated by a unit of work (UOW) that has subsequently failed. Data written to such a queue is always committed and is restored during warm and emergency restarts.

When a UOW reads, writes, or deletes a physically recoverable queue, a log record is written to the system log. When the system is brought up after an abnormal

termination, CICS can re-create a queue by retrieving definition information associated with the queue from the catalog, and state data from the log. .

Note: There is an exception to the rule that states that a physically recoverable queue is restored to the state it was in when CICS abnormally terminated. If a UOW reads a physically recoverable queue and CICS then terminates abnormally, the read operation will be backed out when CICS is subsequently brought back up.

Logical recovery of intrapartition transient data queues: Logically recoverable transient data queues are restored to the state they were in at the time they were last syncpointed. All inflight UOWs are backed out. If a UOW updates a logically recoverable queue and subsequently fails, all updates to the queue are backed out. Logically recoverable queues are restored during warm and emergency restarts.

Logically recoverable queues are logged as part of the first phase of syncpoint processing. When CICS is brought up after an abnormal termination, it can re-create logically recoverable queues by retrieving definition information associated with the queue from the catalog, and state data from the log.

Logically recoverable transient data queues can suffer from indoubt failures. If a UOW is indoubt and CICS abnormally terminates, the indoubt UOW environment is re-created when CICS is next brought up. When the indoubt failure is resolved, the UOW is committed or backed out.

Extrapartition queues

Extrapartition queues are sequential data sets on tape or direct-access devices. Data directed to or from these external queues is called extrapartition data and can consist of sequential records that are fixed- or variable-length, blocked or unblocked.

Data can be placed on an extrapartition data set by CICS for subsequent input to CICS or for offline processing. Sequentially organized data created by other than CICS programs can be entered into CICS as an extrapartition data set. Examples of data that might be placed on extrapartition data sets are:

- System statistics
- Transaction error messages
- Customer data, such as cash payments that can be applied offline.

Indirect queues

Intrapartition and extrapartition queues can be referenced through indirect destinations. This provides flexibility in program maintenance. Queue definitions can be changed, using the CEDA transaction, without having to recompile existing programs.

Automatic transaction initiation

When data is sent to an intrapartition queue and the number of entries (WRITEQs from one or more programs) in the queue reaches a predefined level (trigger level), the user can optionally specify that a transaction be automatically initiated to process the data in that queue.

The automatic transaction initiation (ATI) facility allows a user transaction to be initiated either immediately, or, if a terminal is required, when that terminal has no task associated with it. The terminal processing status must be such that messages can be sent to it automatically. Through the trigger level and automatic transaction initiation facility, an application program can switch messages to terminals. After a

task has been initiated, a command in the application program is executed to retrieve the queued data. All data in the queue is retrieved sequentially for the application program.

Trigger transactions may only execute sequentially against their associated queue. When a trigger transaction has been attached, another transaction will not be attached until the first transaction has completed. If a trigger transaction suffers an indoubt failure, (the transaction must be associated with a logically recoverable queue) another trigger transaction cannot be attached until the indoubt failure has been resolved.

Transient data services

The following services are performed by the transient data program in response to transient data commands issued in application programs:

Intrapartition data disposition
Controls and queues data for serially reusable or re-enterable facilities (programs, terminals) related to this partition or region.

Intrapartition data acquisition
Retrieves data that has been placed in a queue for subsequent internal processing.

Extrapartition data acquisition
Enters a sequentially organized data set into the system.

Extrapartition data disposition
Writes fixed- or variable-length data in a blocked or unblocked format on sequential devices, usually for subsequent offline processing.

Automatic transaction initiation
Initiates a transaction to process previously queued transient data when a predefined trigger level is reached.

Dynamic open/close
Logically opens or closes specified extrapartition data sets (queues) during the real-time execution of CICS.

Dynamic allocation and deallocation of extrapartition queues
Extrapartition transient data queues do not have to be predefined in your JCL. They can be created dynamically.

Transient data

This section describes transient data's interfaces.

Intrapartition queues: Figure 101 on page 1869 shows transient data's interfaces for intrapartition queues.

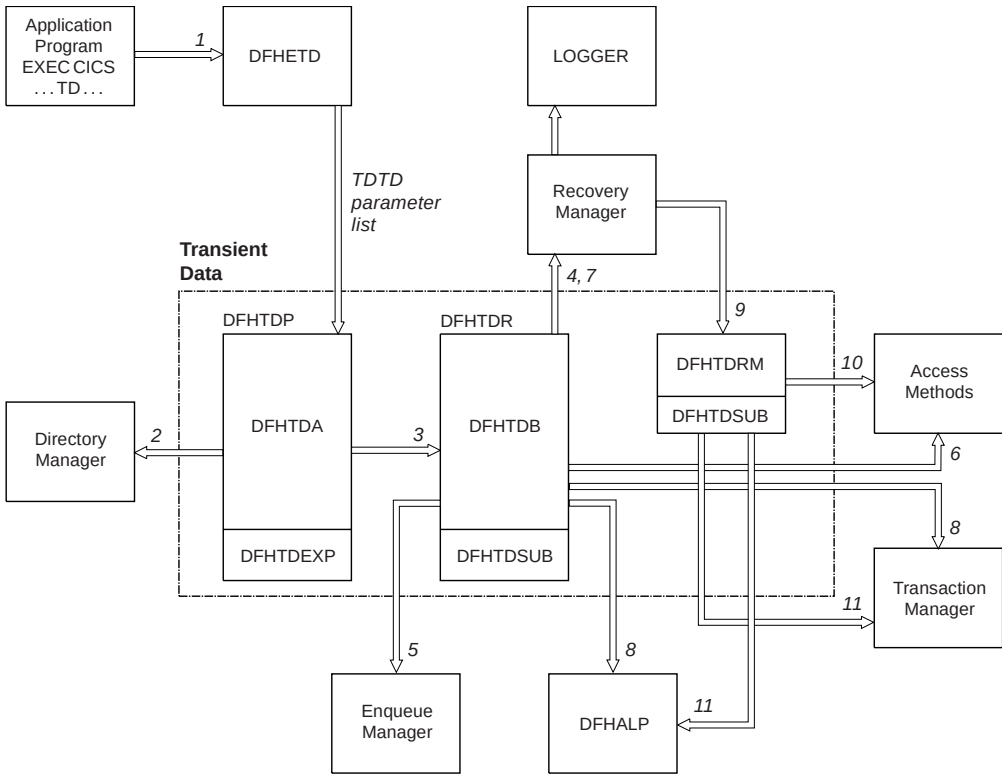


Figure 101. Transient data interfaces for intrapartition queues

Note:

1. An application program invokes a Transient Data request (WRITEQ TD, READQ TD, or DELETEQ TD). The EXEC interface module, DFHETD is invoked and calls Transient Data using the TDTD CDURUN parameter list.
2. Transient Data locates the target queue using a Directory Manager locate.
3. Assuming that the required queue has been found, the call is passed to the module that handles intrapartition queue requests, DFHTDQ.
4. If the target queue is logically recoverable, Transient Data must tell Recovery Manager it is interested in this UOW by setting its work token in the Recovery Manager's table.
5. If the target queue is logically recoverable, Transient Data must obtain an enqueue on the appropriate end of the queue by invoking the Enqueue Manager.
6. Data is read from (or written to) the target queue using the appropriate access method. In the case of physically recoverable queues only, the buffers are always flushed and the data set hardened.
7. After the request has completed, Transient Data must log the state of the queue, if the queue is physically recoverable.
8. If the request was a WRITEQ TD request and the target queue was physically recoverable or non-recoverable, the trigger level may have been exceeded. If the trigger transaction is to be associated with a terminal DFHALP is invoked so that the required AID can be scheduled. If the trigger transaction is to be associated with a file, Transaction Manager is invoked to attach the trigger transaction.

9. If a UOW has updated a logically recoverable queue, Recovery Manager invokes Transient Data when the UOW begins syncpoint processing DFHTDRM.
10. Transient Data invokes the appropriate access methods to harden the data set. Finally, Recovery Manager invokes Transient Data once more, detailing whether Transient Data should commit or back out its updates.
11. If the UOW commits the updates. Transient Data attaches a trigger transaction or schedules an AID if the trigger level has been exceeded. DFHALP is invoked if the trigger transaction is associated with a terminal. Transaction Manager is invoked if the trigger transaction is associated with a file.

Extrapartition queues: Figure 102 shows the transient data interfaces for extrapartition queues.

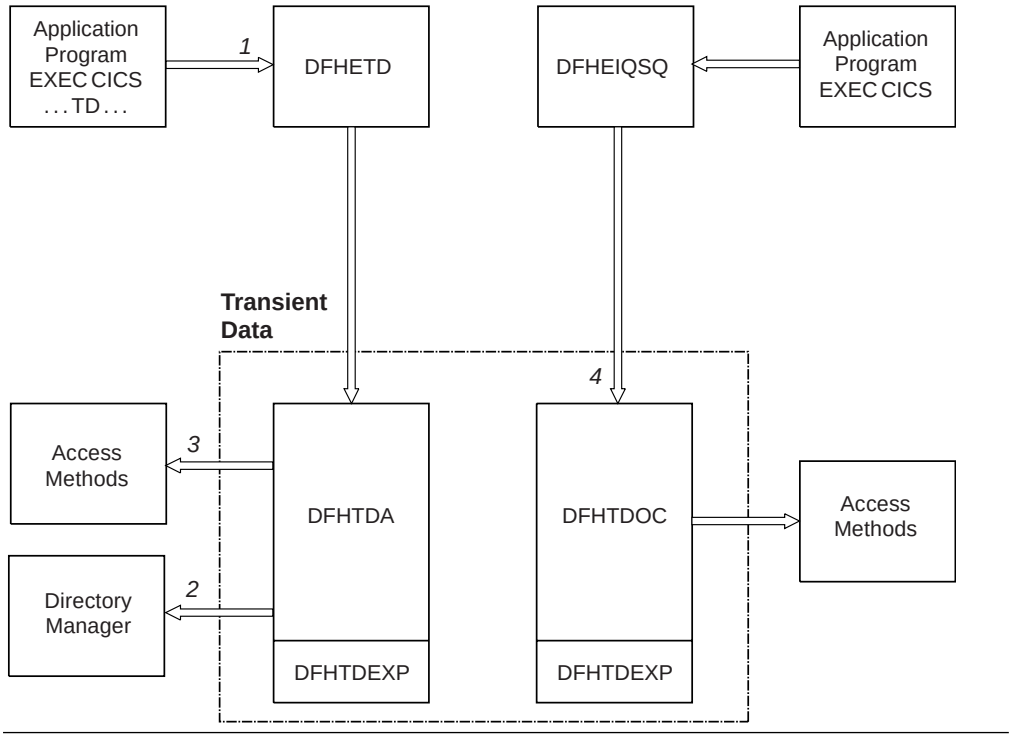


Figure 102. Transient data interfaces for extrapartition queues

Note:

1. An application program invokes CICS transient data services (WRITEQ TD, READQ TD, or DELETEQ TD). The EXEC interface module, DFHETD is invoked. DFHETD invokes CICS transient data services using the TDTD CDURUN parameter list.
2. CICS locates the target queue using Directory Manager.
3. The request is passed to the appropriate QSAM routine for processing. QSAM PUT with LOCATE mode is used.
4. If an application program requests that an intrapartition queue be opened or closed, module DFHTDOC is invoked using the TDOC CDURUN parameter list.

Modules

Module	Function
DFHTDP	Provides request analysis and extrapartition processing.
DFHTDA	Included in load module DFHTDP. Provides request analysis and processing for extrapartition queues.
DFHTDEXC	Included in load module DFHTDP. Contains subroutines associated with the processing of extrapartition queues.
DFHTDOC	Included in load module DFHTDP. Manages the opening and closing of extrapartition queues.
DFHETD	Processes EXEC CICS commands and maps them to the TDTD CDURUN parameter list.
DFHTDB	Included in load module DFHTDQ. Processes intrapartition queue requests.
DFHTDSUC	Included in load module DFHTDQ. Contains subroutines associated with the processing of intrapartition transient data queues.
DFHTDRM	Undertakes sync point processing for transient data queues.
DFHTDTM	Manages requests to install, discard, set, and inquire on transient data queues

Exits

The following global user exit points are provided for this function: XTDREQ, XTDEREQ, XTDEREQC, XTDIN, and XTDOUT.

Trace

The following point ID is provided for transient data control:

- AP F6xx, for which the trace levels are TD 1 and Exc.

User exit control

User exit control enables the user to run exit programs at selected points in CICS modules in the application domain and in other domains. The exit program can be enabled or disabled dynamically, and useful information can be transferred to a user work area.

This function:

- Controls which exit programs are to run at which exit points. This is generally specified using EXEC CICS commands and can be changed during a CICS run.
- Invokes the specified exit programs when control reaches an exit point in a CICS module, and handles any change in flow indicated by a return code from the user exit program.

Design overview

User exit control provides an interface that allows the user to run exit programs at selected points (known as exit points) in CICS control modules. The exit programs are separate from the control modules and are associated with them dynamically by means of the EXEC CICS ENABLE command. (See for a description of how to use exit programs.)

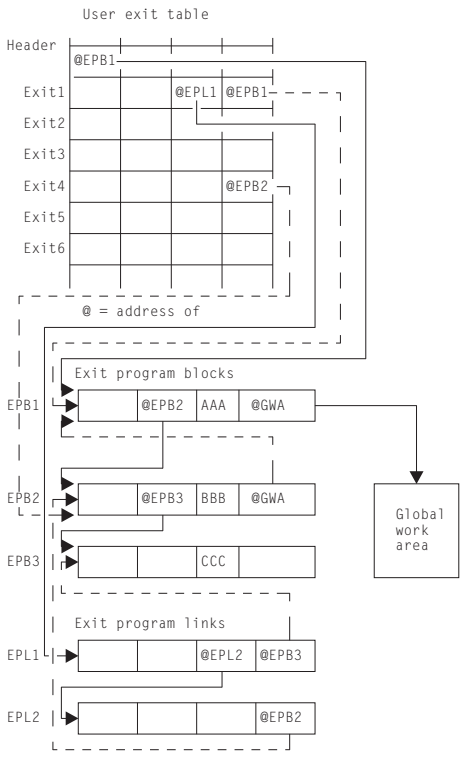
An exit point can have more than one exit program, and an exit program can be shared by more than one exit point. Work areas can be set up for the exit programs, and several exit programs can share a work area. For some exit points, the continuation of the control module can be controlled by a return code.

Each exit point is identified internally by an exit number. The user exit table (UET) contains a UET header and an entry for each exit point, in exit-number order. The UET is addressed from CSAUETBA in the CSA and exists throughout the life of CICS.

Each enabled exit program is represented by an exit program block (EPB). This exists only while an exit program is enabled or while any other exit program is using the work area owned by this exit program. The EPBs are chained together in order of enablement. The UET header points to the first EPB.

Each activation of an exit program for a particular exit point is represented by an exit program link (EPL) which points to the EPB for the exit program. The first EPL for each exit point is contained in the UET entry. If an exit point has more than one exit program, additional EPLs are obtained to represent each subsequent activation. These additional EPLs are chained off the UET entry in order of activation. Thus, for each exit, its EPL chain defines the exit programs that are to be executed at that exit point, and the order of execution.

The user exit interface (UEI) control blocks are illustrated in Figure 103 on page 1873.



Note:

1. There are three enabled programs: AAA, BBB, and CCC.
2. Program AAA owns a global work area, which is shared by program BBB. The global work area pointer (@GWA) in BBB's EPB points to the EPB of the program owning the shared area, namely AAA's EPB.
3. Exits 1 and 4 are associated with these exit programs.
4. For Exit 1, exit programs AAA, CCC, and BBB have been activated, in that order, as indicated by the EPL chain.
5. Exit program BBB has been activated for exit 4.

Figure 103. UEI control blocks

All user exit programs are executed in the AP domain. When exit programs are activated for exit points in other domains, control is passed from the domain to the AP domain's user exit service module, which creates the necessary environment to invoke the exit programs via the user exit subroutine.

User exit control modules

This section describes the function of the user exit control modules.

DFHUEM (user exit manager):

The user exit manager (DFHUEM) processes EXEC commands that are entered by an application program or the command interpreter, to control user exit activity.

DFHUEM contains three routines, corresponding to the three commands, as follows:

ENABLE

Checks whether a user exit block (EPB) already exists for the exit program specified in the PROGRAM operand.

- If an EPB is not found and the ENTRY operand is not specified, the exit program is loaded, and the following actions take place:
  1. A new EPB is obtained and added to the chain.
  2. The name and entry address of the exit program are placed in the EPB.
  3. If the GALENGTH operand is specified, a work area is obtained, and its address and length are placed in the EPB. The work area is obtained from 24-bit storage or 31-bit storage, depending on the value specified in the GALOCATION operand. If the GALOCATION operand is not specified, 24-bit storage is used.
  4. If the GAPROGRAM operand is specified, the address of the EPB for the exit program specified in the GAPROGRAM operand is placed in the new EPB, thus allowing exit programs to share a global work area.
- If the EXIT operand is specified, the exit program link (EPL) chain for the specified exit point is found.
  1. A new EPL is obtained, if necessary, and added to the chain.
  2. The address of the EPB for the exit program specified in the PROGRAM operand is placed in the EPB.
  3. The activation count in the EPB is increased by 1.
  4. If the exit point is not in the AP domain, the domain is notified that the exit point is active.
- If the START operand is specified, the start flag in the EPB is set on.

DISABLE

Finds the EPB for the exit program specified in the PROGRAM operand.

- If the STOP or EXITALL operand is specified, the start-flag in the EPB is set off.
- If the EXIT operand is specified, the EPL chain for the specified exit point is found. The EPL that is pointing to the EPB for the exit program specified in the PROGRAM operand is removed from the chain and the activation count is reduced by 1.
- If the EXITALL operand is specified:
  1. All EPL chains are scanned.
  2. All EPLs pointing to the EPB for the exit program specified in the PROGRAM operand are removed from its chain.
  3. If the ENTRY operand was not specified when the exit program was enabled, the exit program is deleted.
  4. The EPB is removed from the chain.
  5. If a work area used by the exit program is not still being used by another exit program, it is released.
  6. Any EPB or EPL that is no longer required is moved to a free-chain anchored in the user exit table header (UETH).
- When EXIT or EXITALL is specified for exit points that are not in the AP domain, the domain is notified when there are no exit programs active.

EXTRACT-EXIT

Finds the EPB for the exit program specified in the PROGRAM operand. The work area's address and length are extracted from this EPB (or from the EPB that owns the work area) and placed in the user's fields specified in the GASET and GALENGTH operands.

DFHUEH (user exit handler): The user exit handler module, DFHUEH, is used to process exit points in the AP domain.

At each exit in a control module, there is a branch to the DFHUEH program. This module scans the EPL chain for that exit and invokes each started exit program in the chain, passing it a parameter list and a register save area. On return from each

exit program, the return code is checked and a current return code (maintained by DFHUEH for return to the control module) is set as appropriate.

DFHAPEX (user exit service module): The user exit service module, DFHAPEX, is used to process exit points in domains other than the AP domain.

When an exit point is reached in a non-AP domain, control is passed to the user exit service module (DFHAPEX) in the AP domain, if the domain has previously been notified that there is an exit program activated for the exit point.

The user exit service module constructs the user exit parameter list, using special parameters from the domain, and invokes the user exit subroutine (DFHSUEX).

The return code from DFHSUEX is passed back to the calling domain.

DFHSUEX (user exit subroutine): The DFHSUEX module invokes all started user exit programs for an exit point in a domain (other than the AP domain) by scanning the EPL chain, using the same processing as the user exit handler (DFHUEH). The parameter list defined by DFHAPEX is passed to the exit programs. Return codes from the exit programs are checked and returned to DFHAPEX.

Control blocks

The control blocks associated with the user exit interface are listed and are illustrated in Figure 104 on page 1876. Further information about the control blocks is given in the "Design overview" on page 1871 and in Figure 103 on page 1873.

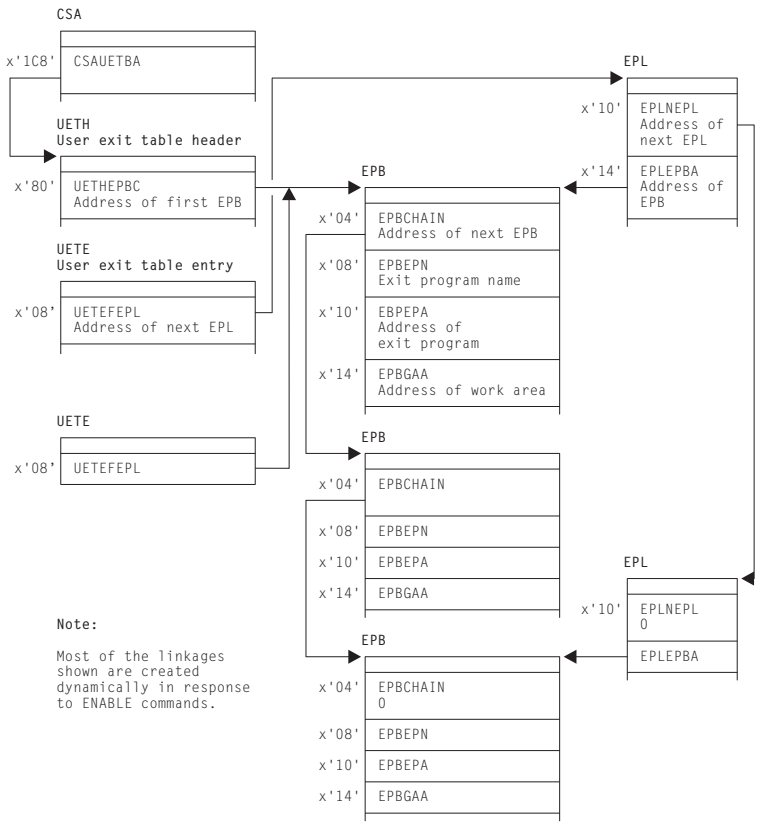


Figure 104. Control blocks associated with the user exit interface

The main control blocks are as follows:

UETH User exit table header

UETE User exit table entry: one for every exit point

EPB Exit program block: one for every enabled user exit program, containing information about the location and activity of the program, and any global work area owned or shared by the program

EPL. Exit program link. Each EPL indicates one exit program to be invoked at an exit point and which EPL, if any, contains information about the next program to be invoked at that exit point.

Modules

Module	Function
DFHAPEX	The interface between an exit point in a domain (other than the AP domain) and the AP domain.
DFHSUEX	Handles the invocation of user exit programs at exit points in CICS domains (other than the AP domain). Processing is similar to DFHUEH, passing a parameter list defined in DFHAPEX.
DFHUEH	Links an exit point in a CICS management module in the AP domain and the user code. DFHUEH invokes in turn each started exit program for that exit point, passing a parameter list defined in the CICS management module.
DFHUEM	The EXEC interface processor for the ENABLE, DISABLE, and EXTRACT user exit commands.

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:

- AP D5xx, for which the trace levels are UE 1, AP 1, AP 2, and Exc.

For user exit programs running at an exit point within the AP domain, UE level-1 trace entries are produced.

For user exit programs running at an exit point in a CICS domain other than the AP domain, the UE level-1 trace entries are not produced. Instead, the D5xx trace entries for AP level 1 and AP level 2 are available, providing more information than the UE trace. For AP level 1, the DFHUEPAR parameter list is traced, containing the addresses of fields special to that exit point. For AP level-2 tracing, the contents of the fields are printed, each field being truncated to 200 bytes if necessary.

WTO and WTOR

Design overview

The DFHSUWT module provides the following support for executing MVS WTO and WTOR SVCs:

SEND supports Write To Operator (WTO):

- A single-line message up to 113 characters, or a multiline message consisting of a control line and up to nine lines of 69 characters
- Route code specification (route code list of 1 through 28 numbers, each in the range 1 through 28)
- Descriptor code specification (descriptor code list of 1 through 16 numbers, each in the range 1 through 16).

CONVERSE

supports Write To Operator With Reply (WTOR):

- A single-line message up to 121 characters
- Route code specification (route code list of 1 through 28 numbers, each in the range 1 through 28)
- Descriptor code specification (The descriptor code lists one number through each in the range 1-28. It is made up of 16 numbers, each in the range 1 through 16.)
- A reply with maximum length of 119 characters.

The DFHWTO macro may be used to send a message, normally to the system operator, when neither the CICS message domain nor the old message program (DFHMGP) can be used. The message domain cannot be used during certain phases of initialization, because it requires a kernel stack environment. DFHMGP cannot be used during initialization, nor during any sort of abend or dump processing, because it uses task LIFO storage and may therefore invoke the storage control program.

The DFHWTO macro may also be used to terminate CICS abnormally or to request a reply from the operator.

Any WTO or WTOR macros that are issued by CICS might be intercepted by the console message handling facility described under “Console message handling” on page 1774

page 1774. This service optionally inserts the CICS region's applid into CICS messages before they are displayed on the console.

Modules

DFHSUWT and DFHWTO

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:

- AP FF0x, for which the trace levels are AP 1 and Exc.

z/OS Communications Server generic resource

This section describes how the generic resource support provided by the z/OS Communications Server is used by CICS.

A CICS system may register as a Communications Server generic resource. It may then be known either by its unique applid or by the generic resource name which is shared by a number of CICS systems, all of which are registered to the same generic resource.

For more information about CICS support for z/OS Communications Server generic resource consult the Configuring z/OS Communications Server generic resources in Configuring. Consult *z/OS Communications Server: SNA Programming* for information about generic resource from the z/OS Communications Server point of view.

Design Overview

If CICS is to register as a generic resource member, the GRNAME system initialization parameter must be specified.

If GRNAME is specified CICS attempts to register immediately after the ACB is open by issuing the z/OS Communications Server SETLOGON OPTCD=GNAMEADD command.

If registration succeeds, CICS is then a member of the generic resource specified by the SIT GRNAME parameter and may be addressed either by its generic resource name or (subject to certain restrictions) by its unique applid. Use of the generic resource name allows z/OS Communications Server to balance the workload by selecting whichever generic resource member is most lightly loaded.

If registration fails, CICS initialization continues but CICS will not be a generic resource member.

The registration status may be examined by means of the CEMT INQUIRE VTAM command.

CICS de-registers as a generic resource by means of the VTAM SETLOGON OPTCD=GNAMEDEL command immediately before the ACB is closed.

Note: VTAM is now z/OS Communications Server.

Generic resource and LU6.1/LU6.2

Although terminals may log on freely using either the generic resource name or the member name this is not the case with LU6.1 and LU6.2 connections which are more restricted in their use of member names.

LU6.2 GR to GR connections

For LU6.2 connections between generic resources the design makes use of LU6.2 autoinstall. Only connections which are intended to issue an ACQUIRE need be defined and these must all have the generic resource name specified as the NETNAME.

Two types of connection are possible.

- Generic resource name connections. These are connections which have the generic resource name as the NETNAME. NETNAMEs must be unique and so there can only be one of these per partner generic resource.
- Member name connections. These are connections which have the unique applid (member name) as the NETNAME.

Since there can only be one generic resource name connection for each partner generic resource it follows that most connections will be member name connections.

EXEC CICS INQUIRE CONNECTION or CEMT INQUIRE CONNECTION may be used to determine which is the generic resource name and which the member name.

When the first BIND from a different generic resource comes into the SCIP exit (DFHZBLX), a generic resource name connection will be established. If no predefined generic resource name connection exists one will be autoinstalled. Subsequent BINDs coming into DFHZBLX from different members of the same generic resource will cause member name connections to be autoinstalled. A member name connection should never be defined for a member of a different generic resource because this creates the possibility of having two definitions (TCSE's) for the same connected system.

Communications between members of the same generic resource must be by member names only.

Two new bits TCSE_GR and TCSE_GRNAME_CONN have been introduced to indicate the different connection types. They are only valid for LU6.2 connections between generic resources.

The table shows different values of TCTENNAM, TCSESID and TCSEX62N for LU6.2 connections between generic resources, depending on the settings of TCSE_GR and TCSE_GRNAME_CONN:

TCSE_GR TCSE_GRNAME_CONN	ON ON	ON OFF
TCTENNAM TCSESID TCSEX62N	GRname GRname membername	membername membername GRname

LU6.2 GR to non-GR connections

If a single (non-generic resource) system has an LU6.2 connection to a generic resource member it may use either the generic resource name or the member name as the NETNAME.

If the member name is used the initial acquire of the connection must be done by the non-generic resource partner. This means that the generic resource side must not have autoconnect set on. This is because the generic resource partner relies on z/OS Communications Server to tell it if it is to known by its member name. z/OS Communications Server does this by setting a bit which is valid for the first BIND only. Sessions can be acquired by either partner once the SNASVCMG sessions have bound.

For these connections TCSE_GR is always set off and TCSE_GRNAME_CONN has no meaning on both systems. The rule here is that TCSESID always contains the NETNAME (as defined in the RDO connection definition) and TCSEX62N always contains the member name (unique applid). The table illustrates this:

TCSE_GR TCSE_GRNAME_CONN RDO_HOSTNAME	OFF not applicable GRname	OFF not applicable membername
TCTENNAM TCSESID TCSEX62N	GRname GRname membername	membername membername membername

If the generic resource name is to be used, the single system may itself be made into a generic resource allowing it to exploit the design for communications between generic resources. If this is not possible the solution is to use a "hub" or code a generic resource resolution exit to ensure that not more than one member of a generic resource communicates with the single system at any one time using the generic resource name. (The use of "hubs" is described in the CICS Intercommunications Guide).

LU6.1

There is no autoinstall for LU6.1, and so less flexibility is allowed for LU6.1 connections between generic resources. CICS-CICS LU6.1 connections can only communicate by generic resource names and must use a "hub" or a generic resource resolution exit.

TCSE_GR and TCSE_GRNAME_CONN do not apply to LU6.1. For LU6.1 connections with a generic resource the generic resource name is in TCTENNAM and TCSESID and the member name is in TCSEX61N.

Ending affinities

Affinities are records held by z/OS Communications Server to show it where to direct data flows within a generic resource. Some of these affinities are "owned" by CICS. These are affinities for LU6.2 synclevel 2, LU6.2 limited resources and LU6.1 connections. They may be ended by means of the SET CONNECTION ENDAFFINITY and PERFORM ENDAFFINITY commands.

Generic resource and ATI

This section applies only to those terminals which are logged on using the generic resource name.

When an ATI request is issued in an AOR for a terminal that is logged on to a TOR, CICS uses the terminal definition in the AOR to determine the identity of the TOR to which the request should be shipped. If there is no terminal definition in the AOR, the "terminal-not-known" global user exits (XICTENF and XALTENF) may be used to supply the name of the TOR.

However, if the TOR in question is a member of a generic resource and the user has logged on using the generic resource name, z/OS Communications Server will have connected the terminal to the generic resource member which was most lightly loaded at the time. If the user then logs off and on again the terminal may be connected to a different generic resource member. If this happens, the TOR which is to receive the ATI request cannot be determined from the terminal definition in the AOR or the "terminal-not-known" user exit.

CICS solves the problem in the following manner:

1. The ATI request is first shipped to the TOR specified in the terminal definition in the AOR (or by the "terminal-not-known" exit). If the terminal is logged on to this TOR (the "first-choice" TOR) the ATI request completes as normal.
2. If the terminal is not logged on to the first-choice TOR, the TOR issues a VTAM INQUIRE OPTCODE=SESSNAME to find which generic resource member, if any, the terminal is now logged on to. This information is passed back to the AOR and the request is then shipped to the correct TOR.
3. If the first-choice TOR is not available, the AOR issues a VTAM INQUIRE OPTCODE=SESSNAME to find where the terminal is now logged on. The INQUIRE is not attempted in the following situations:
   - The z/OS Communications Server in the AOR is a pre-4.2 version and does not support generic resource.
   - The AOR was started with the z/OS Communications Server system initialization parameter set to NO.

   The INQUIRE will not succeed if the TORs and the AOR are in different networks.

   If the INQUIRE is successful the ATL request is shipped to the TOR where the terminal is logged on.

Note: VTAM is now the z/OS Communications Server.

Modules

DFHZBLX

DFHZBLX is a new module which has been created to deal with LU6.2 BIND processing. Part of its function was formerly part of DFHZSCX. It is link-edited with DFHZSCX and is still logically part of it, but it returns directly to z/OS Communications Server, not via DFHZSCX.

There is a new part of the module, apart from that which was once contained in DFHZSCX, which deals with generic resource BIND processing. If CICS is registered as a generic resource and the partner is also a generic resource, DFHZBLX has to decide on the appropriate type of connection. This may be either

a generic resource name connection, in which the NETNAME is the partner's generic resource name, or a member name connection, in which the NETNAME is the partner's member name.

DFHZBLX is also responsible for setting the bits in the connection entry which are specific to generic resource.

If CICS is not registered as a generic resource, the generic resource code is not invoked.

DFHZGCH

DFHZGCH is a domain subroutine which is called by DFHEIQSC after one of the following commands.

- EXEC CICS SET CONNECTION ENDAFFINITY
- CEMT SET CONNECTION ENDAFFINITY
- EXEC CICS PERFORM ENDAFFINITY
- CEMT PERFORM ENDAFFINITY

Its function is to issue the z/OS Communications Server CHANGE OPTCD=ENDAFFINITY command.

If the affinity is ended successfully,

- the connection is deleted if it is autoinstalled.
- If the connection is defined,
  - the generic resource specific information in the connection entry is reset,
  - the catalog entry is updated,
  - the connection is deleted from the TCSM index.

The z/OS Communications Server return codes are reflected back to DFHEIQSC.

DFHZGIN

DFHZGIN is a domain subroutine.

In a TOR it is called by DFHCRS when a request has been shipped from a remote system, if a terminal cannot be located.

In an AOR it is called by DFHALP when the schedule of an AID fails because the TOR has gone away.

It has two functions:

1. INQUIRE_NQN

   A VTAM INQUIRE OPTCD=NQN is issued to find the fully qualified NETNAME of a terminal given the NETNAME as input. The fully qualified NETNAME is required for INQUIRE OPTCD=SESSNAME.
2. INQUIRE_SESSNAME

   A VTAM INQUIRE OPTCD=SESSNAME is issued to find which member of a generic resource a terminal is logged on to given a fully qualified NETNAME as input.

The following responses are returned to the caller:

- OK - z/OS Communications Server return code was X'00' fdb2 X'00'
- NOT FOUND - z/OS Communications Server return code X'14' fdb2 X'88'

- EXCEPTION - The call was rejected for some other reason than not found.

For the exception case an exception trace is written and a message in the range DFHZC0182 - DFHZC0185 is output to the CSNE log giving the z/OS Communications Server return codes.

Note: VTAM is now z/OS Communications Server.

Problem solving for generic resource

Trace TC level 1, 2 & exception in the ranges AP FA50-FA59, FAB0-FABA and FB87-FB8F.

Messages DFHZC0170 to DFHZC0185 are written to the console and CSNE logs.

Information output by DFHZNAC following BIND failures.

If a dump is produced examine the generic resource status and generic resource flag bytes.

The following symptoms may indicate that an affinity should be ended and has not been.

- Sessions failing to acquire with message DFHZC2405 "Node not activated". This may also indicate a setup error.
- Sessions failing to acquire with various instances of DFHZC2411. This may also indicate that a rule has been violated.
- CICS fails to register as a generic resource when it has previously been a member of a different generic resource. Message DFHZC0171 is written to the console with z/OS Communications Server rtncd X'14' fdb2 X'86'.
- Connections autoinstalling unexpectedly. If a non-generic resource is addressing a generic resource member by its member name this may also indicate that the first ACQUIRE was issued from the generic resource side.

Generic resource status byte (TCTV_GRSTATUS)

TCTV_GR_REGD (X'80')
This CICS is registered as a member of a generic resource.

TCTV_GR_REGERR (X'40')
This CICS attempted to register as a generic resource member (SIT GRNAME parameter specified) but the attempt was rejected by z/OS Communications Server.

TCTV_GR_NOTAVAIL (X'20')
This CICS attempted to register as a generic resource member (SIT GRNAME parameter specified) but the level of z/OS Communications Server was not 4.2 or later.

TCTV_GR_DREGD (X'08')
This CICS was previously a member of a generic resource but has successfully de-registered.

TCTV_GR_DREGERR (X'04')
This CICS attempted to de-register as a member of a generic resource by issuing SETLOGON OPTCD=GNAMEDEL but the attempt was rejected by z/OS Communications Server.

TCTV_GR_NOTAPPL (X'02')
The GRNAME system initialization parameter was not specified.

TCTV_GR_NOTREG (X'00')
CICS is not registered as a generic resource and has not attempted to register. (Holds this value before registration is attempted, if required.)

Generic resource flag byte (TCSEI_GR)

TCSE_GR (X'80)
Both partners are registered as generic resources. Valid from initial acquire to ENDAFFINITY.

TCSE_GR_NAME_CONN (X'40')
Set on for a generic resource name connection in which TCSESID contains the generic resource name and TCSEX62N contains the member name.

Set off for a member name connection in which TCSESID contains the member name and TCSEX62N contains the generic resource name.

This bit is only meaningful if TCSE_GR is set on.

TCSE_USE_OUR_MEMBER_NAME (X'20')
The partner is using our member name. (An indication that the member name, not the generic resource name must be passed in the BIND).

TCSE_MSG179_ISSUED (X'10')
Message DFHZC0179 has been issued. This message is issued when the secondary SNASVCMG session binds if TCSE_GR is set. It makes clear which is the generic resource name and which the member name of the partner session.

TCSE_CATLG_DONE (X'08')
A defined connection with an affinity has been catalogued.

TCSE_MSG177_ISSUED (X'04')
Message DFHZC0177 has been issued. This message is output whenever an LU6.2 limited resources, LU6.2 synclevel 2 or LU6.1 connection is acquired. It is output when the secondary SNASVCMG session binds. It is intended to alert the user to the fact that acquiring the connection has caused an affinity to be created and gives the NETNAME and NETID of the partner.

Trace

Trace point ids

- FA50 - FA59

  are provided for problem determination during ENDAFFINITY processing. (Module DFHZGCH)
- FAB0 - FABA

  are provided for problem determination during INQUIRE SESSNAME processing. (Module DFHZGIN)
- FB87 - FB8F

  are provided for problem determination during generic resource registration and de-registration. (Module DFHZGSL)

Waits

Module	Type	Resource Name	Resource Type	ECB	Function
DFHZGCH	MVS	CHANGECB	ZC_ZGCH	CHANGECB	Wait for completion of INQUIRE SESSNAME

Module	Type	Resource Name	Resource Type	ECB	Function
DFHZGIN	MVS	INQ_ECB	ZC_ZGIN	INQ_ECB	Wait for ENDAFFINITY to complete

z/OS Communications Server LU6.2

This section describes the layer of CICS that manages the interface to the z/OS Communications Server for LU6.2 communication. The Communications Server LU6.2 provides advanced program-to-program communication (APPC) between transaction-processing systems, and enables device-level products (APPC terminals) to communicate with host-level products and with each other. APPC sessions can therefore be used for CICS-to-CICS communication, and for communication between CICS and other APPC systems (for example, AS/400) or terminals.

For information about the CICS functions that you can use to exploit LU6.2 communication, see "Distributed program link" on page 1547, "Distributed transaction processing" on page 1549, "Function shipping" on page 1706, "Intersystem communication (ISC)" on page 1732, "Transaction routing" on page 1845.

Design overview

The main feature that distinguishes LU6.2 from other LU types is the support for parallel sessions i.e. many sessions (and conversations) between the two LUs at the same time. These sessions are further grouped by use of the class of service facility in z/OS Communications Server. The TCT structure for LU6.2 reflects this. Under the system entry (TCTSE) are a series of mode group entries (TCTMEs). Within a mode group there are a number of sessions represented by terminal entries (TCTTEs).

All the sessions within a mode group have the same transmission characteristics, that is, the same class of service. When a request to ALLOCATE a session is made, a MODENAME can be specified, indicating which class of service is required.

When a session has been allocated and a conversation started, data can be received and sent between the connected LUs. This is more or less directly under the control of the CICS application in the case of DTP, or indirectly under the control of the user for the other ISC facilities.

CICS also supports LU6.2 single session connections. These are represented by a TCTSE, a single TCTME and a single TCTTE. They support the same functions as parallel session connections.

Detailed information about z/OS Communications Server LU6.2 commands and macros is given in the relevant z/OS Communications Server manuals.

Session management

Systems Network Architecture (SNA) defines several processes to be used in managing LU6.2 sessions. The CICS implementation provides transaction code for the following Transaction Program Names (TPNs) defined by LU6.2.

- X'06F1' = CHANGE_NUMBER_OF_SESSIONS (CNOS)
- X'06F2' = EXCHANGE_LOG_NAME (XLN)

The required transaction definitions are:

TRANSACTION	XTRANID	PROGRAM
CLS1	X'06F10000'	DFHZLS1
CLS2	X'06F20000'	DFHCLS3

These resource definitions are provided in the DFHISC group.

So that the SNA service transaction programs can always communicate with each other, even when all the sessions between two systems are busy, two extra sessions are always created whenever parallel sessions exist between two systems. CICS generates these two extra sessions (with a reserved MODENAME of SNASVCMG) unless SINGLESESS(YES) is specified for the connection. Only SNA service transaction programs are allowed to use these two sessions.

Change Number Of Sessions (CNOS): When there are parallel sessions between two LU6.2 systems, it is possible to vary the number of sessions available using CEMT or EXEC CICS commands, either for the entire connection, or by modegroup. The number of available sessions for a modegroup is called the SESSION LIMIT. It corresponds to the number of in-service sessions in that modegroup. The two systems must agree on the session limit for a modegroup at any given time. To achieve this, the LU6.2 architecture defines a CNOS service transaction program which runs in each system, communicating with its counterpart using architected CNOS commands and replies. They negotiate the session limit and the numbers of contention winners and losers at each end. For CICS, the CNOS service transaction program is DFHZLS1.

CNOS commands are not required for the SNASVCMG modegroup on parallel session connections, or for single session connections, because the session limits are fixed.

Figure 105 shows the flow of control for CNOS operations.

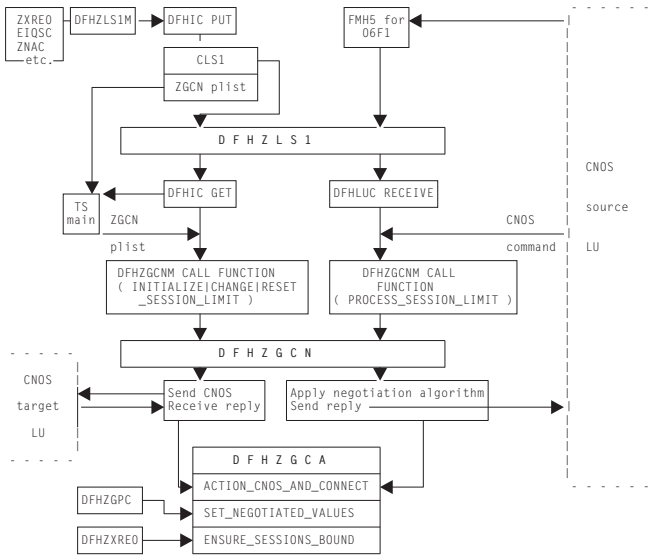


Figure 105. Flow of control for CNOS

Exchange Log Name (XLN): When DFHZNAC determines that it is necessary to exchange log names with a remote system, it starts the syncpoint resynchronization transaction, using the DFHCRERI macro specifying FUNCTION(XLN). The main program for this transaction is DFHCRRSY (in load module DFHLUP). When DFHCRRSY determines that resynchronization is required it will schedule other instances of itself to perform the resynchronization.

When TPN X'06F2' is received from a remote system, DFHCRRSY is called to handle the inbound Exchange Log Names and resynchronization.

LU6.2 session states

The following CICS modules maintain specific states of LU6.2 sessions.

Module	State	Macro
DFHZBKT	SNA bracket state	DFHZBSM
DFHZCNT	Contention state	DFHZCNM
DFHZCHS	Chain state	DFHZCHM
DFHZCRT	RPL_B state	DFHZCRM

These modules are invoked via the macros shown in the last column. Any query or change to the states is performed using these macros.

The LU6.2 states for each session are stored in the TCTTE for that session. The modules and associated TCTTE field are usually referred to as **state machines**. When a module, such as DFHZARL, wants to check that the session is in a suitable state to perform a given operation, it uses the appropriate state machine to perform the check by invoking the CHECK function of the relevant macro. If the operation subsequently causes a change in the state of the session, the SET function of the relevant macro is invoked to record the new state.

LU6.2 SEND and RECEIVE processing

LU6.2 SEND processing is done by DFHZSDL, using POST=SCHED to drive the z/OS Communications Server exit DFHZSLX asynchronously when the request has been passed to z/OS Communications Server.

DFHZRVL does LU6.2 RECEIVE processing, issuing the request to z/OS Communications Server for asynchronous processing which drives the z/OS Communications Server exit DFHZRLX on completion. DFHZRLX queues completed RPLs for further processing by DFHZRLP to a chain anchored off TCTVRPLQ in the TCT prefix. Entries are removed from the queue by DFHZDSP, and passed to the program designated to process the completed RPL. When the z/OS Communications Server authorized path feature is used, the SEND and RECEIVE requests use the CICS high performance option (HPO) routines.

SEND and RECEIVE processing for LU6.2 use different RPLs:

- RECEIVE uses the receive RPL (also known as RPL_B, and addressed by TCTERPLB in the TCTTE LUC extension).
- SEND uses the send RPL (addressed by TCTERPLA in the TCTTE).

There are two exceptions when a SEND uses the receive RPL instead of the send RPL:

1. DFHZSDL sending a response
2. DFHZRLP sending DR1 response via synchronous SEND.

The processing state of the receive RPL is maintained in the LU6.2 RPL_B state machine field (TCTERPBS in the TCTTE LUC extension) by the DFHZCRT module and DFHZCRM macro combination, thus allowing rapid identification of the stage and type of RECEIVE being processed.

LU6.2 state machine transitions for contention, bracket, and chain states are performed via the DFHZCNM, DFHZBSM, and DFHZCHM macros as part of SEND and RECEIVE processing for LU6.2 sessions.

Limited resources

For efficient use of some network resources (for example, switched lines), SNA allows for such resources to be defined in the network as **limited resources**. Whenever a session is bound, z/OS Communications Server indicates to CICS whether the bind is over a limited resource. Both single and parallel sessions may use limited resources.

The limited resources (LR) function is part of the LU6.2 base option set. When communicating over switched lines, it may be important to stop using this expensive resource as soon as possible. LR provides this facility. A bit in the BIND image is copied into the TCTTE to indicate LR usage. This bit (TCTE_LR) is used to determine whether CICS should UNBIND the link when the TCTTE is freed and no outstanding tasks are using the link.

SNASVCMG (parallel) sessions are not scheduled to be unbound until the initial CNOS exchange has been performed for all mode groups in the connection. They are then treated in the same way as user sessions.

Two bits in the terminal control table are used to reflect LR: TCTE_LR in the terminal entry (TCTTE) and TCSE_LR in the system entry (TCTSE). The following table shows the meanings of the TCTE_LR bit (ON or OFF) in combination with the TCTENIS 'node now in session' bits (YES or NO).

TCTE_LR	TCTENIS	Meaning
ON	YES	Current session over LR
ON	NO	Previous session over LR
OFF	YES	Current session not LR
OFF	NO	Never bound, or previous session not LR

TCSE_LR (in the system entry) is set ON when the first LR session is bound, and OFF as a result of CNOS negotiation to release the connection. If TCSE_LR is ON and there are no bound sessions, the connection state is then 'available'.

Modules

The modules in the following list handle the z/OS Communications Server LU6.2 support in CICS.

Session management state machines

- DFHZBKT
- DFHZCHS
- DFHZCNT
- DFHZCRT

Send and Receive processing

- DFHZRLP

- DFHZRLX
- DFHZRVL
- DFHZSDL
- DFHZSLX

CNOS

- DFHZLS1
- DFHZGCN
- DFHZGCA

Persistent Verification

- DFHCLS3

XLN and Resynchronization

- DFHCRRSY

DFHZRVL

DFHZRVL is invoked to issue an LU6.2 receive specific request to receive:

- Data
- Commands
- Responses
- Purge to end-chain (used by DFHZERH to clear incoming data)
- A single RU.

Two broad categories of RECEIVE data are recognized by CICS; both are processed as RECEIVE_WAIT requests to z/OS Communications Server:

1. RECEIVE_WAIT, where CICS waits until input is received from z/OS Communications Server before returning control to the caller. This applies to all RECEIVE response and command requests, and to data requests where the minimum length to be received is greater than zero.
2. RECEIVE_IMMEDIATE, where CICS immediately returns control to the caller without waiting for z/OS Communications Server to complete the request unless the data is already in the z/OS Communications Server buffer, in which case it processes the data in the same way as for RECEIVE_WAIT before returning to the caller. This is requested via a minimum length of zero. It is used by the RECEIVE_IMMEDIATE call for the SAA communications interface, by a LOOK_AHEAD call, and in support of timely receipt of responses, ensuring earlier detection of an ISSUE_ERROR response from the partner LU.

The receive buffer is set up to receive the data, and the address of the receive exit DFHZRLX (driven on completion of the request) is stored into the receive RPL (RPL_B) before the RECEIVE macro is issued to z/OS Communications Server. DFHZRVL is used by DFHZERH to determine the state of the session.

DFHZRLP

This module completes the LU6.2 receive specific processing for LU6.2 requests.

RECEIVE_IMMEDIATE requests are processed in two phases, that is, on two passes through DFHZRLP:

1. The RPL_B state machine (TCTERPBS) is set to indicate that the RECEIVE has been completed by z/OS Communications Server; then the exit is taken from DFHZRLP.

2. This phase corresponds to the single phase used for processing RECEIVE_WAIT requests, that is, the requests are checked for successful completion, examined to determine whether data, a command, or a response has been received, and parameters indicating what has been received are then returned to the caller.

Data received: When data is received, DFHZRLP:

1. Sets the bracket and chain state machines, and returns indicators to DFHZARL according to the DFC flags received with the data:
   - Response type
   - CD
   - EC
   - CEB
   - FMH
2. If more data is required, DFHZRLP recalls DFHZRVL via the activate scan routine (DFHZACT) to reissue the RECEIVE, for example when:
   - End-chain has not yet been received, and there is still room in the receive buffer. If the minimum length requested has already been received, the type of RECEIVE is altered from RECEIVE_WAIT to RECEIVE_IMMEDIATE resulting in a READ_AHEAD call in anticipation of there being more data available, and any data already in the z/OS Communications Server buffer is processed by DFHZRLP before returning to the caller.
   - The original request was for data, and what has been received and processed is a command (only LUSTAT or BIS can validly be processed by DFHZRLP).
3. Returns control to DFHZARL when:
   - Sufficient data has been received for a BUFFER or LL type request.
   - End-chain has been received because of CD, RQD2, or CEB.
   - FMH has been received.
   - The call was incomplete, but insufficient space remains in the receive buffer for further data.

If the data was received with RQD1, a response is sent synchronously by DFHZRLP using the receive RPL.

Command received: When a command is received, the actions of DFHZRLP depend on the command:

- For LUSTAT6 received, the command is treated as data. If BB is included, then an exception response is sent (sense X'0813' or X'0814').
- For BIS received, CLSDST is requested and the receive re-driven.

All other commands are incorrect.

Response received: When a response is received, DFHZRLP:

1. Carries out checks:
   - Does the sequence number match the number of the BB request?
   - If it is a definite response, was it expected?
   - If it is an exception response, was it a session-level error?
2. Sets the state machines.
3. Passes back the return code to the caller.

DFHZSDL

This module issues the SEND request to z/OS Communications Server to transmit data, commands, and responses on LU6.2 sessions.

DFHZSDL transmits:

- Data from a send buffer or an application area
- The commands:
  - LUSTAT
  - RTR
  - BIS
- Responses.

Data transmission: If a SEND LAST command is issued, any outstanding completed receive RPL is first processed by queuing the TCTTE for RECEIVE processing by DFHZRLP, and any incomplete receive RPL is canceled via RESETSR.

For data transmission, DFHZSDL uses:

LMPEO
: Large message performance enhancement outbound. z/OS Communications Server slices large messages into RUs.

BUFFLST
: Buffer list. z/OS Communications Server accepts data from non-contiguous buffers.

USERRH
: User request header. The request header is passed in BUFFLST.

A maximum of two buffer list entries are used. The first buffer list entry addresses the data in the send buffer, and the second the data in the application area.

The request header is built in the first buffer list entry using parameters passed from DFHZARL. If an implicit send was requested, then CD, RQD2, and CEB are not checked. The first-in-chain (FIC) indicator is set after checking the chain state machine, and last-in-chain (LIC) is set whenever CD, RQD2, or CEB is included. Null data sent only-in-chain (OIC) is converted to an LUSTAT6 command. The address of the send exit DFHZSLX is stored in the send RPL, and the z/OS Communications Server SEND macro is issued. On completion of the SEND request, the bracket and chain state machines are set according to the DFC indicators. These state machines are used extensively by DFHZERH to determine the state of the session before executing an error request.

Command transmission: The LUSTAT6 command is sent with:

- CEB to terminate the BIND_in_bracket state
- Null data for OIC
- CB, RQD1 to BID for bracket.

The RTR command requests BB after a BID request is rejected with sense code X'0814'.

The BIS command shows bracket termination before CLSDST.

On completion of the SEND request, the exit DFHZSLX is invoked. LUSTAT causes the bracket and chain state machines to be set as for normal data flow.

Response transmission: DFHZSDL transmits ER1 and DR2 responses. The sequence number associated with the response is that of the path information unit (PIU) that initiated the current bracket. DFHZSDL uses the receive RPL (RPL_B) to send responses thus ensuring that the RU is returned with the response, unless the response is an ISSUE_ERROR request, in which case the send RPL is used. The response is sent synchronously, and POST=SCHED is included in the z/OS Communications Server command, so that an exit routine is not involved. On return from z/OS Communications Server, DFHZSDL sets the bracket and chain state machines accordingly.

DFHZSLX

The DFHZSLX module is the z/OS Communications Server exit that is driven on completion of a SEND request. If the request completed successfully, the bracket and chain state machines are set to show the new state of the session. If the SEND request was data DR1, DFHZRVL is invoked via DFHZACT to receive the response.

DFHZRLX

The DFHZRLX module is the z/OS Communications Server exit that is scheduled on completion of an LU6.2 RECEIVE_SPECIFIC request. DFHZRLX queues the completed RPL to a chain anchored from TCTVRLPQ in the TCT prefix. DFHZDSP dequeues the RPLs for further processing by DFHZRLP.

DFHCLS3

In the local CICS system, DFHCLS3 is invoked using the DFHLUS macro, which issues a DFHIC TYPE=PUT macro to start the appropriate transaction (CLS3) with data recorded on temporary storage indicating the requested operation.

The DFHLUS operations can be:

SIGNOFF
: Sign off a user on the other LU

TIMEOUT
: Time out users.

The SIGNOFF and TIMEOUT operations apply to persistent verification signons only.

DFHCLS3 retrieves the temporary-storage record.

The SIGNOFF and TIMEOUT operations are performed directly by DFHCLS3. These operations are supported outbound only.

For SIGNOFF, DFHCLS3 is started by DFHZCUT when a user on the other LU must be signed off.

For TIMEOUT, DFHCLS3 is started by DFHZCUT during timeout processing of a **persistent verification signed-on-from list**, also known to CICS as a local userid table (LUIT).

DFHCLS3 performs the following actions:

1. Calls DFHZCUT to find a userid that needs to be timed out
2. Makes a sign-off call to the other LU
3. Calls DFHZCUT to remove the userid from the LUIT.

This sequence is repeated until there are no more userids to be timed out.

If DFHCLS3 abends during timeout processing, control passes to a SETXIT routine in DFHCLS3, which calls DFHZCUT to tidy up the relevant LUIT.

DFHZLS1

DFHZLS1 is the main program for the CICS implementation of the CNOS SNA service transaction.

When acting as the initiator of a CNOS request (the CNOS source), it is invoked by the DFHZLS1M macro issuing a DFHIC TYPE=PUT for transaction id CLS1. The possible commands on the CNOS source system are as follows:

- INITIALIZE_SESSION_LIMIT

  Acquire the specified connection, using the MAXIMUM values from the RDO SESSIONS definitions (for the required session limit and number of winner sessions) on the CNOS command for each modegroup.
- CHANGE_SESSION_LIMIT

  Negotiate a change of the current session limit for a specified modegroup.
- RESET_SESSION_LIMIT

  Release the connection, negotiating all modegroups to a session limit of zero.

When acting as the receiver of a CNOS request (the CNOS target), DFHZLS1 is invoked by an attach FMH for TPN X'06F1' sent from the CNOS source system, which is not necessarily CICS. The CNOS command sent with the attach FMH requests changes to the sessions in specified modegroups. In SNA terms, DFHZLS1 is handling a PROCESS_SESSION_LIMIT command. It issues a DFHLUC RECEIVE for the CNOS GDS that contains the details of the required command.

DFHZLS1 passes the parameters for each of the commands in the previous list through to DFHZGCN, where the detailed processing takes place.

DFHZGCN

DFHZGCN is an AP domain subroutine. It handles the following four architected CNOS functions.

INITIALIZE_SESSION_LIMIT: This is a two pass function in CICS. In the first pass, DFHZGCN initiates the bind of the SNASVCMG winner session and returns. The bind processing eventually causes the “session started” routine in DFHZNAC to run. This routine re-issues the DFHZLS1M INITIALIZE_SESSION_LIMIT request, and the CNOS negotiation can then take place.

DFHZGCN performs the following actions:

1. Does a “privileged” allocate (for a SNASVCMG session).
2. Builds an attach header.
3. Completes the building of the CNOS command, using MAXIMUM values in the TCTME.
4. Issues a SEND INVITE WAIT.
5. Issues a RECEIVE LLID.
6. Analyzes the responses to the command; SNA decrees that the CNOS source must accept the values returned.
7. Calls DFHZGCA to action the new values.
8. Sends messages DFHZC4900 and DFHZC4901 as appropriate.

9. Frees the session.

These steps are repeated for each user modegroup in the connection.

RESET_SESSION_LIMIT: A connection release request is passed via DFHZLS1 to DFHZGCN.

DFHZGCN performs the following actions:

1. Does a 'privileged' allocate.
2. Builds an attach header.
3. Completes the building of one CNOS command, setting MAX, WIN, and LOS values to zero, and mode names affected to ALL.
4. Issues SEND INVITE WAIT.
5. Issues RECEIVE LLID.
6. Analyzes the response to the command; the CNOS target must accept zero sessions (DRAIN can be changed from ALL to NONE).
7. Calls DFHZGCA to action the new values.
8. Sends message DFHZC4900.
9. Frees the session.

CHANGE_SESSION_LIMIT: DFHZLS1 is started from the EXEC API or CEMT via DFHEIQSM to change the session limit for a specific modegroup.

DFHZGCN performs the following actions:

1. Does a 'privileged' allocate.
2. Builds an attach header.
3. Completes the building of one CNOS command, setting MAX and WIN values.
4. Issues SEND INVITE WAIT.
5. Issues RECEIVE LLID.
6. Analyzes the responses to the command; SNA decrees that the CNOS source must accept the values returned.
7. Calls DFHZGCA to action the new values.
8. Sends messages DFHZC4900 and DFHZC4901 as appropriate.
9. Frees the session.

PROCESS_SESSION_LIMIT: DFHZLS1 is attached, and calls DFHZGCN.

DFHZGCN performs the following actions:

1. Addresses the CNOS command that DFHZLS1 passed.
2. For each mode group specified, determines whether the values for session limit, source contention winners and source contention losers are acceptable. If not, the values are adjusted (negotiated) according to rules laid down by SNA.
3. If this system is currently performing shutdown, negotiates down to session limit zero.
4. Calls DFHZGCA to action the new values.
5. Sends the CNOS reply containing the negotiated values.
6. Sends messages DFHZC4900 and DFHZC4901 as appropriate.

DFHZGCA

DFHZGCA is an AP domain subroutine. It has the following three separate functions.

ACTION_CNOS_AND_CONNECT: After a CNOS negotiation DFHZGCA is responsible for changing the state of a specified modegroup to reflect the new values. There are three types of action required.

1. Put sessions in/out of service for session limit increase/decrease.
2. Set sessions to winner/loser in line with negotiated values.
3. Bind/unbind sessions for session limit decrease, autoconnect processing or contention polarity switch.

SET_NEGOTIATED_VALUES: This function is used by DFHZGPC during persistent sessions restart to set the saved CNOS values in the modegroup without any binding/unbinding of sessions.

ENSURE_SESSIONS_BOUND: DFHZXRE0 invokes this function during persistent sessions restart because recovery processing can lead to LU6.2 sessions becoming unbound. It is important to ensure that they are re-bound in accordance with the autoconnect setting.

Exits

No global user exit points are provided for this function.

Trace

All of the modules listed in "Modules" on page 1888 have entry and exit trace points. Several of them also have exception and level 2 trace points. All of these trace points are from the AP domain and have IDs in the range FB00-FCFF.

z/OS Communications Server persistent sessions support

This diagnosis information describes in detail how CICS handles z/OS Communications Server persistent sessions support. When persistent sessions support is exploited by a CICS region, sessions can be recovered if CICS, z/OS Communications Server, or z/OS fails, depending on the type of support.

Design overview

CICS support of persistent sessions includes the support of all LU-LU sessions, except LU0 pipeline and LU6.1 sessions. With multinode persistent sessions support, if z/OS Communications Server fails, LU62 synclevel 1 sessions are restored, but LU62 synclevel 2 sessions are not restored.

The CICS system initialization parameter **PSTYPE** specifies the type of persistent sessions support for a CICS region:

SNPS, single-node persistent sessions
: Persistent sessions support is available, so that Communications Server sessions can be recovered after a CICS failure and restart. This setting is the default.

MNPS, multinode persistent sessions
: In addition to the SNPS support, Communications Server sessions can also be recovered after a Communications Server or z/OS failure in a sysplex.

NOPS, no persistent sessions
Persistent sessions support is not required for the CICS region. For example, a CICS region that is used only for development or testing might not require persistent sessions.

The time specified by the **PSDINT** system initialization parameter for the region determines how long the sessions are retained. If CICS, z/OS Communications Server, or z/OS fails, if a connection to z/OS Communications Server is reestablished within this time, CICS can use the retained sessions immediately; there is no need for network flows to rebind them.

You can change the persistent sessions delay interval using the **CEMT SET VTAM** command, or the **EXEC CICS SET VTAM** command. The changed interval is not stored in the CICS global catalog, and therefore is not restored in an emergency restart.

Note: VTAM is now the z/OS Communications Server.

If CICS fails or undergoes immediate shutdown (by means of a **PERFORM SHUTDOWN IMMEDIATE** command), z/OS Communications Server holds the CICS LU-LU sessions in recovery pending state, and they can be recovered during startup by a newly starting CICS region. With multinode persistent sessions support, sessions can also be recovered if z/OS Communications Server or z/OS fails in a sysplex.

During an emergency restart of CICS, CICS restores those sessions pending recovery from the CICS global catalog and the CICS system log to an in-session state. This process of persistent sessions recovery takes place when CICS opens its VTAM ACB. With multinode persistent sessions support, if the Communications Server or z/OS fails, sessions are restored when CICS reopens its VTAM ACB, either automatically by the COVR transaction, or by a CEMT or **EXEC CICS SET VTAM OPEN** command. Although sessions are recovered, any transactions inflight at the time of the failure are abended and not recovered.

Subsequent processing depends on the LU. Clean up and recovery for non-LU6 persistent sessions is similar to that for non-LU6 backup sessions under XRF. Clean up and recovery for LU6.2 persistent sessions maintains the bound session when possible, but in some cases it might be necessary to unbind and rebind the sessions, for example, where CICS fails during a session resynchronization.

When a terminal user enters data during persistent sessions recovery, CICS appears to hang. The screen that was displayed at the time of the failure remains on display until persistent sessions recovery is complete. You can use options on the TYPETERM and SESSIONS resource definitions for the CICS region to customize CICS so that either a successful recovery can be transparent to terminal users, or terminal users can be notified of the recovery, allowing them to take the appropriate actions.

If APPC sessions are active at the time of the CICS, Communications Server or z/OS failure, persistent sessions recovery appears to APPC partners as CICS hanging. The Communications Server saves requests issued by the APPC partner, and passes them to CICS when recovery is complete. When CICS reestablishes a connection with the Communications Server, recovery of terminal sessions is determined by the settings for the PSRECOVERY option of the CONNECTION resource definition and the RECOVOPTION option of the SESSIONS resource definition. You must set the PSRECOVERY option of the CONNECTION resource definition to the default value SYSDEFAULT for sessions to be recovered. The

alternative, NONE, means that no sessions are recovered. If you have selected the appropriate recovery options and the APPC sessions are in the correct state, CICS performs an **ISSUE ABEND** to inform the partner that the current conversation has been abnormally ended.

Situations in which sessions are not reestablished

When z/OS Communications Server persistent sessions support is in use for a CICS region, CICS does not always reestablish sessions that are being held by the Communications Server in a recovery pending state. In the situations listed here, CICS or the Communications Server unbinds and does not rebind recovery pending sessions.

- If CICS does not restart within the persistent sessions delay interval, as specified by the **PSDINT** system initialization parameter.
- If you perform a COLD start after a CICS failure.
- If CICS cannot find a terminal control table terminal entry (TCTTE) for a session; for example, because the terminal was autoinstalled with AIRDELAY=0 specified.
- If a terminal or session is defined with the recovery option (RECOVOPTION) of the TYPETERM or SESSIONS resource definition set to RELEASESESS, UNCONDREL or NONE.
- If a connection is defined with the persistent sessions recovery option (PSRECOVERY) of the CONNECTION resource definition set to NONE.
- If CICS determines that it cannot recover the session without unbinding and rebinding it.

The result in each case is as if CICS has restarted following a failure without Communications Server persistent sessions support.

In some other situations APPC sessions are unbound. For example, if a bind was in progress at the time of the failure, sessions are unbound.

With multinode persistent sessions support, if a Communications Server or z/OS failure occurs and the TPEND failure exit is driven, the autoinstalled terminals that are normally deleted at this point are retained by CICS. If the session is not reestablished and the terminal is not reused within the AIRDELAY interval, CICS deletes the TCTTE when the AIRDELAY interval expires after the ACB is reopened successfully.

Situations in which the z/OS Communications Server does not retain sessions

When z/OS Communications Server persistent sessions support is in use for a CICS region, in some circumstances the Communications Server does not retain LU-LU sessions.

- If you close the Communications Server with any of the following CICS commands:
  - **SET VTAM FORCECLOSE**
  - **SET VTAM IMMCLOSE**
  - **SET VTAM CLOSED**

  Note: VTAM is now the z/OS Communications Server.
- If you close the CICS node with the Communications Server command **VARY NET INACT ID**=*applid*.

- If your CICS system performs a normal shutdown, with a **PERFORM SHUTDOWN** command.

If single-node persistent sessions support (SNPS), which is the default, is specified for a CICS region, sessions are not retained after a Communications Server or z/OS failure. If multinode persistent sessions support (MNPS) is specified, sessions are retained after a Communications Server or z/OS failure.

Persistent sessions restart flow

Diagnostic information about the process of persistent sessions recovery.

Enabling of persistence:

CICS requests persistent sessions support when it opens the z/OS Communications Server ACB.

Summary

1. z/OS Communications Server ACB opened with PARM=PERSIST=YES
2. z/OS Communications Server levels checked.
3. z/OS Communications Server SETLOGON OPTCD=PERSIST or NPERSIST

More detail

Persistence is enabled as follows:

1. The z/OS Communications Server ACB is opened with PARM=PERSIST=YES, specified in DFHTCTPX.
2. DFHZSLS calls DFHZGSL to issue SETLOGON OPTCD=PERSIST/NPERSIST. DFHZSLS copies 8 bytes of z/OS Communications Server information into the TCT prefix. These bytes contain details of the z/OS Communications Server level and the functions that it supports. Releases of CICS that did not support persistent sessions copied only 4 bytes of z/OS Communications Server data.

The use of persistent sessions depends on the level of z/OS Communications Server being at least V3R4.1 for single-node persistent sessions support. This level of z/OS Communications Server returns more function bit data to CICS than previous versions and supports the use of persistent sessions. Checks are made by CICS of the current z/OS Communications Server level and the z/OS Communications Server level against which the TCT was generated. If either level is not high enough, parameters relating to the use of persistent sessions are not used when macros are called.

Sessions that persist at CICS startup:

These tasks and modules are involved when z/OS Communications Server persistent sessions are restored on a CICS restart.

Summary

1. Task CGRP runs DFHZCGRP.
2. DFHZCGRP calls DFHZGRP.
3. DFHZGRP issues a VTAM INQUIRE instruction.
4. DFHZGRP performs one of the following actions:
   - Terminates session via DFHZGUB issuing CLSDST/TERMSESS.
   - Restores the session with OPNDST TYPE=RESTORE.
5. DFHZGRP queues restored sessions for further processing.

6. DFHZGRP issues RECEIVE_ANY commands.
7. DFHZGRP does some CNOS work.
8. DFHZGRP does some URD work.
9. Queued sessions are restored.

Note: VTAM is the previous name for z/OS Communications Server.

More detail

Sessions that persist at startup time are processed in the following way:

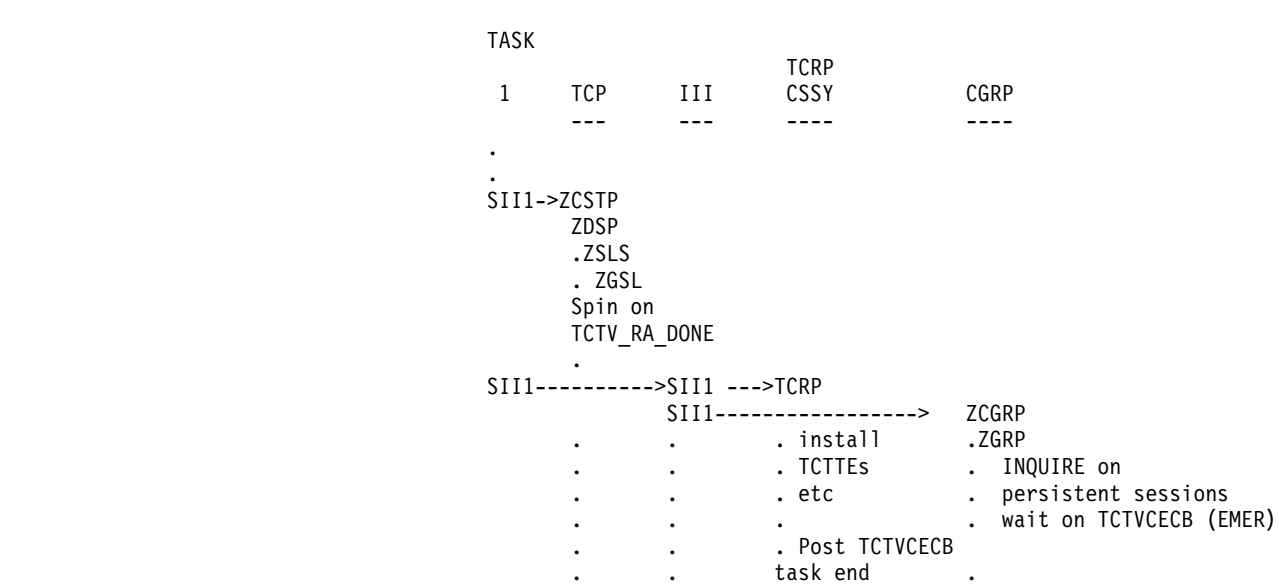
1. Attach task CGRP - program DFHZCGRP in DFHSII1 after TCRP is attached.
2. DFHZCGRP calls DFHZGRP with a START_TYPE of one of the following:
   - COLD
   - WARM
   - EMER_XRF
   - EMER
3. DFHZGRP issues VTAM INQUIRE instructions in chunks; that is, z/OS Communications Server is passed an area with a size defined in the TCT prefix.

   The area is filled with NIBs by z/OS Communications Server. DFHZGRP scans the NIBs and determines whether to UNBIND or OPNDST each session.

   For COLD, WARM, and EMER_XRF, all sessions are unbound.

   For EMER, some sessions are unbound and some restored depending on the circumstances.
4. Restored sessions are queued to DFHZACT for further processing by DFHZXRC or DFHZXPS.
5. RECEIVE_ANY initialization done.
6. CNOS records are processed by making calls to DFHZGPC.
7. URDS are reset to AWAITING RE_SYNCHRONIZATION for EMER only.
8. DFHZACT calls DFHZXRC or DFHZXPS for each session queued by DFHZGRP.

Task and module flow diagram



```
.       .                       .  process persistent
.       .                       .  sessions
.       .                       .  RECEIVE_ANY processing
ZDSP continues <------------------ set TCTV_RA_DONE
        .                          post TCTV_ZGRP_FIN_ECB
        .                       task end
        . Wait on
        . TCTV_ZGRP_FIN_ECB
        SIJ1
        . SETLOGON START
        . Start CXRE task
        . Control is Given to CICS
ZACT
. ZXRC
. ZXPS
```

Task and module flow: more detail

1. Startup runs as normal until DFHSII1 has started the TCP (CSTP) task and DFHZDSP runs.
2. DFHZDSP calls DFHZSLS.
   - If z/OS Communications Server is at V3R4.1 or later, DFHZSLS calls DFHZGSL to issue SETLOGON OPTCD=PERSIST if the value of the system initialization parameter PSDINT is a valid nonzero value.
   - If the z/OS Communications Server level is at V3R4.0, or PSDINT is 0 or defaulted with higher levels of z/OS Communications Server, DFHZSLS calls DFHZGSL to issue SETLOGON OPTCD=NPERSIST.
   - If the z/OS Communications Server level is earlier than V3R4.0, the SETLOGON OPTCD call is not made because PERSIST and NPERSIST are not supported for these z/OS Communications Server releases.

   DFHZSLS does *not* issue RECEIVE OPTCD=ANY. It returns to DFHZDSP, which "spins" until TCTV_RA_DONE is set by DFHZGRP when the RECEIVE_ANY commands have been successfully issued.
3. DFHSII1 attaches the III task which continues to run code in DFHSII1.
4. DFHSII1 (III) attaches and calls DFHTCRP as a system task and then attaches task CGRP, which runs program DFHZCGRP which calls ZGRP.
5. DFHZGRP calls DFHZGUB if there are any sessions to unbind.
6. DFHZGRP queues any sessions to be restored to DFHZACT.
7. DFHZGRP sets TCTV_RA_DONE after issuing RECEIVE_ANY commands to allow DFHZDSP to continue.
8. DFHZGRP posts TCTV_ZGRP_FIN_ECB.
9. When DFHZGRP finishes, control is returned to code in DFHZCGRP.

   DFHZCGRP checks the RESPONSE and REASON code. It sets TCTV_ZGRP_FAILED off if RESPONSE(OK) or RESPONSE(EXCEPTION) with REASON(ACB_CLOSED|INQUIRE_FAILED). Otherwise, it sets TCTV_ZGRP_FAILED on.
10. DFHSII1 waits on TCTV_ZGRP_FIN_ECB and checks if TCTV_ZGRP_FAILED was set on by DFHSII1.

    If TCTV_ZGRP_FAILED is off, DFHSII1 continues. Otherwise, it sets INITDERR, which causes CICS to stop when the other tasks have finished.
11. Just before CONTROL IS GIVEN to CICS, DFHSIJ1 attaches the CXRE task to run DFHZXRE0, which does some additional PRSS processing.
12. DFHZXRC or DFHZXPS are then called to process any TCTTEs queued to DFHZACT.

13. DFHZXRC is called by DFHZACT to process non-APPC sessions that have not been unbound by DFHZGRP. It takes one of the following actions depending on the state of the session, the terminal type, and how the TYPETERM for the session has been defined to CICS:
    - Send END_BRACKET.
    - Send CLEAR (followed by START_DATA_TRAFFIC for SNA devices which support it).
    - Unbind.

    For those devices for which the clean up action is not to unbind, the TCTTE is queued to DFHZNAC and message DFHZC0146 is issued for the session.

    As part of the processing for message DFHZC0146, any recovery notification requested for the session is initiated:
    - If the requested recovery notification is MESSAGE, DFHZNCA sends a BMS map to the terminal.
    - If the requested recovery notification is TRANSACTION, DFHZNCA initiates the requested transaction.
14. DFHZXPS is called by DFHZACT to process APPC sessions.

    DFHZXPS takes one of the following courses of action depending on the setting of TCTE_PRSS on entry.
    - Examines the data pointed to by TCTV_PRSS_CV29_PTR to determine the state of the session at system failure.
        a. If a task is attached, calls DFHZGDA to issue DEALLOCATE,ABEND for the task still running on the partner.
        b. If no task is attached but there is further recovery to be done, for example, bid recovery or outstanding responses, sets the TCTTE to a state which allows this further recovery to proceed. If the existing mechanism will carry out the recovery without further intervention by DFHZXPS, removes the TCTTE from the DFHZACT queue; otherwise, requeues the TCTTE to DFHZACT and DFHZXPS will be recalled at a later stage to finish recovery processing.
        c. If no task is attached and there is no further recovery to be done, removes the TCTTE from the DFHZACT queue because recovery is now complete.
    - Recalls DFHZGDA to continue with DEALLOCATE,ABEND or REJECT_ATTACH processing.
    - Requeues the TCTTE to DFHZACT if a SEND (for example, of an outstanding response), which was set in motion by an earlier instance of DFHZXPS, is still in progress.
    - Issues CLSDST for the session if an error has occurred during the recovery process.
    - Carries out further recovery as described earlier, if required, following successful completion of DEALLOCATE,ABEND processing.
    - Removes the TCTTE from the DFHZACT queue when all recovery has completed.

Sessions that persist when CICS opens the z/OS Communications Server ACB:

These tasks and modules are involved when the z/OS Communications Server ACB is dynamically opened by a **SET VTAM OPEN** command from a running CICS system. With single-node persistent sessions support (SNPS), if z/OS Communications Server fails but CICS continues to run, sessions no longer exist.

Summary

With multinode persistent sessions support (MNPS), sessions do persist if z/OS Communications Server or z/OS fails. CICS does not delete the autoinstalled resources, and resets all the terminal and connection sessions to unopened state.

1. CEMT SET VTAM OPEN.
2. DFHEIQVT calls DFHZOPA.
3. DFHZOPA calls DFHZSLS.
4. DFHZSLS call DFHZGSL.
5. DFHZGSL issues SETLOGON PERSIST or NPERSIST.
6. DFHZOPA calls DFHZGRP.
7. DFHZGRP issues INQUIRE PERSESS.
8. DFHZGRP terminates the session by means of DFHZGUB issuing CLSDST/TERMSESS. However, if MNPS is in use, the sessions are restored using OPNDST RESTORE instead.
9. DFHZGRP issues RECEIVE_ANY commands.
10. DFHZGRP deletes CNOS catalog records.
11. DFHZOPA issues SETLOGON START.

More detail

Sessions that persist after the ACB has been opened using a **SET VTAM OPEN** command are processed in the following way:

1. CICS is running with the z/OS Communications Server ACB closed. **CEMT SET VTAM OPEN** or **EXEC CICS SET VTAM OPEN** is issued.
2. DFHEIQVT calls DFHZOPA to open the ACB.
3. DFHZOPA calls DFHZSLS.
4. DFHZSLS calls DFHZGSL.
5. DFHZGSL issues z/OS Communications Server macro calls dependent on the z/OS Communications Server level and PSDINT value.
   - If z/OS Communications Server is V3R4.1 or later, DFHZGSL issues SETLOGON OPTCD=PERSIST if the value of the **PSDINT** system initialization parameter is a valid nonzero value.
   - If the z/OS Communications Server level is V3R4.0, or PSDINT is 0 or defaulted with higher levels of z/OS Communications Server, DFHZGSL issues SETLOGON OPTCD=NPERSIST.
   - If the z/OS Communications Server level is earlier than V3R4.0, the SETLOGON OPTCD call is not made because PERSIST and NPERSIST are not supported for these z/OS Communications Server releases.
6. DFHZOPA calls DFHZGRP with startup type of DYNOPEN.
7. DFHZGRP issues INQUIRE PERSESS with a storage area that will take up to about 400 sessions. INQUIRE PERSESS is reissued until all the NIBs have been obtained from z/OS Communications Server.
8. DFHZGRP calls DFHZGUB if there are any sessions to unbind. For MNPS, DFHZGRP instead issues OPNDST RESTORE for each session that persists.
9. DFHZGRP issues RECEIVE_ANY commands.
10. DFHZGRP calls DFHZGCC to delete CNOS records.
11. If ZGRP returns RESPONSE(OK) or RESPONSE(EXCEPTION) with REASON(ACB_CLOSED|INQUIRE_FAILED), DFHZOPA issues SETLOGON

OPTCD=START. Otherwise, it causes DFHZSHU to close the z/OS Communications Server ACB and then returns to DFHEIQVT.

Persistent sign-on under persistent sessions:

If CICS has persistent verification defined, the sign-on is not active under persistent sessions until the first input is received by CICS from the terminal.

1. After the persistent session has been recovered, the TCTTE is marked to indicate that the sign-on will persist.
2. The RECOVNOTIFY message or transaction is processed. Because RECOVNOTIFY is processed before persistent sign-on is recovered, only the first transaction specified in the RMTRAN system initialization parameter is processed; the second transaction specified cannot be processed because security has not yet been restored.
3. The user presses an Attention IDentifier (AID) key.
4. CICS runs the CPSS transaction to recover the sign-on.

Modules

These modules are involved in z/OS Communications Server persistent sessions recovery.

ZC (terminal control) together with the following modules:

Module	Function
DFHZCGRP	Program initiated by task CGRP to set up the start type and to call DFHZGRP during initialization. It then analyses the response from DFHZGRP and decides if CICS can continue or not.
DFHZGCA	Sets the appropriate ZC control blocks to reflect the currently agreed Change Number Of Session (CNOS) values for an LU6.2 connection.
DFHZGCC	Performs catalog and retrieval of CNOS data. This module is called when CICS needs either to store or to recover CNOS values. During a CICS run, all CNOS values are written to the global catalog. Under normal circumstances they are not needed. However, if a persistent sessions restart is performed, it is necessary to recover the CNOS values that were in operation at the time of the CICS failure. This recovery is achieved by having a record on the global catalog that can be read in during PRSS restart and used to restore the sessions to their state before failure. This module handles the maintenance of the CNOS records during normal CICS operation and the recovery of the records during PRSS recovery.
DFHZGCN	Handles the process of LU6.2 Change Number Of Sessions (CNOS) negotiation, acting as either the source or target end of the conversation, and calls DFHZGCA to action the resulting changes.

Module	Function
DFHZGDA	Takes control of APPC conversations that have persisted across a CICS failure, and ensures that they are ended cleanly, by issuing a Deallocate(Abend) informing the partner LU that the CICS transaction has abended. If DFHZGDA is working correctly the CICS failure and restart is transparent to the partner LU, which understands only that the CICS transaction with which it was communicating has ended. DFHZGDA also performs REJECT_ATTACH processing for synclevel 2 conversations that are started by the partner before Exchange Lognames has been done after a persistent sessions restart.
DFHZGPC	Performs recovery of CNOS values for modegroups. This module is called when CICS is performing a persistent sessions (PRSS) restart. When a PRSS restart is performed, it is not enough to recover the sessions. It is also necessary to recover the CNOS state that the sessions had before the CICS failure. DFHZGCC will have maintained a record of the CNOS state on the global catalog. This record is now used in this module in an attempt to restore CNOS values.
DFHZGPR	The role of DFHZGPR is to update the global catalog whenever it is necessary to add, delete, or test for a record indicating that an APPC connection has a Persistent Resource associated with it. A Persistent Resource can be defined as some session state, or piece of work upon which the partner LU depends, and which will be lost if CICS fails. The only Persistent Resource so far identified is a shipped AID. Before persistent sessions, the failure of the APPC session tells the partner that these resources have been lost, and drives his recovery. With the advent of persistent sessions, it is necessary for a persisting CICS to know that an APPC session had a Persistent Resource associated with it, so that the connection can be unbound (to drive the partner's clean up) and then rebound.
DFHZGRP	Initialize z/OS Communications Server persistent sessions. DFHZGRP is a domain subroutine but is called by DFHZCGRP (task CGRP) during initialization. DFHZGRP is called during ZC initialization or when the z/OS Communications Server ACB is opened dynamically by CEMT SET VTAM OPEN or EXEC CICS SET VTAM OPEN by DFHEIQVT. The module performs these tasks: 1. OPNDST RESTOREs or CLSDST/TERMSESS any session that z/OS Communications Server has held persisting, depending on startup type and session parameters. 2. It calls DFHZGPC to reinstate CNOS records during an emergency restart, or calls DFHZGCC to delete CNOS catalog records. 3. It initializes the RECEIVE_ANY RPLs and issues the RECEIVE_ANYs.
DFHZGSL	Informs z/OS Communications Server whether sessions are to persist or not. This module is called when CICS needs to set, unset, or change the Persistent Sessions PSTIMER value.

Module	Function
DFHZGUB	Issue CLSDST or TERMSESS for individual NIBs in a NIBLIST. This module is called by DFHZGRP to unbind NIBs in a NIBLIST in two ways: • Unbind the entire NIBLIST for COLD, WARM, EMER+XRF and dynamic open. • Unbind only the NIBs with NIBUSER = 0 for EMER starts.
DFHZXPS	DFHZXPS handles Persistent Sessions recovery for APPC sessions. It does not deal with non-APPC sessions, which are dealt with by DFHZXRC. DFHZXPS is called by DFHZACT after OPNDST OPTCD=RESTORE has been issued successfully for a persisting APPC session. Both single and parallel APPC sessions are dealt with, but there is no difference in the processing. The task of DFHZXPS is to examine z/OS Communications Server session tracking data which was hung off TCTE_PRSS_CV29_PTR by DFHZGRP following a Persistent Sessions restart, and, if possible, to update the TCTTE to allow work to continue on the session. If it is not possible to determine the state of the session before system failure, or the session was not in a state which allows it to be recovered, the session is unbound.
DFHZXRC	DFHZXRC analyses the Session State Vector data that is hung off TCTE_PRSS_CV29_PTR by DFHZGRP during an emergency restart, for each persisting session. The necessary action to clean up and recover the session is then initiated.

Diagnosing persistent sessions problems

Consult this data when diagnosing problems with z/OS Communications Server persistent sessions support.

Note: VTAM is the previous name for z/OS Communications Server.

- Trace, TC level 1, 2, and exception in the range of AP FB10-FBFF.
- **CEMT INQUIRE VTAM** showing the PSTYPE and PSDINT values. The setting NOPS for PSTYPE means that persistent sessions support was not requested at startup of the CICS region. A zero setting for PSDINT means that sessions are not retained.
- Console and CSNE logs:
  - Persistent session messages (DFHZC0001 to DFHZC0162)
  - Information produced by DFHZNAC
- Dumps taken by some of the messages listed earlier. If a NIBLIST was present at the time the dump was taken, you can examine it by printing the TCP section of the dump.
- Last flow information, that is, the CV29, FMH5, BIS, and BID information, is useful if a session is in the wrong state after a persistent session restart. This might have been diagnosed by an error message, or maybe missed and message DFHZC0146 or DFHZC0156 issued.

  TCTE_PRSS_CV29_PTR points to the CV29, FMH5, BIS, and BID information which was created by DFHZGRP and used by DFHZXPS or DFHZXRC. It is freed when DFHZNAC issues message DFHZC0146 or DFHZC0156. Otherwise, it is freed when the session is unbound.

The last flow information is traced by DFHZXPS as a TC level 1 trace. If you have a dump, but no trace level 1 available, it is dumped in the TCP section for each TCTTE for which it still exists.

- The contents of byte TCTE_PRSS are useful. Values other than X'00' and X'FF' indicate that something went wrong during the PRSS recovery. The possible values are listed in the Data areas. If a value remains in this byte, the meaning can indicate where the recovery went wrong. The values are described in "Persistent sessions status byte (TCTE_PRSS)" on page 1908.
- The contents of the state machines are useful.
  - TCTECNTS, contention state machine
  - TCTEBKTS, bracket state machine
  - TCTECHSS, chain state machine
  - TCTEUSRS, user state machine
- The contents of TCTE_BID_STATUS are useful. They are described in "Bid status byte (TCTE_BID_STATUS)" on page 1910.

Here are some possible problems:

- DFHZGRP can cause CICS to stop during initialization for the following reasons:
  - DFHZGRP has been called with invalid parameters.
  - DFHZGRP cannot complete the receive any process.
  - DFHZGRP has had a loop or abend.
  - DFHZGRP cannot switch back to the QR TCB.
  - DFHZGRP has failed before any NIBs have been obtained from z/OS Communications Server (with INQUIRE OPTCD=PERSESS).
  - DFHZGRP or DFHZGUB has issued a z/OS Communications Server request that failed to respond within 5 minutes. Issued with message DFHZC0128 and a system dump.

  In each case DFHZGRP or a function it has called issues a message giving a reason for the failure.
- Sessions might be unbound by DFHZGRP for the following reasons:
  - The open of the ACB is dynamic, for example, CEMT SET VTAM OPEN. However, if MNPS is in use, sessions are normally restored at this point.
  - The TCTTE has not been found, probably because it has not been cataloged. Either the terminal was autoinstalled with AIRDELAY=0, or it was an APPC clone. No message is written because this state is considered to be normal.
  - CICS does not support recovery for LU61 or pipeline sessions.
  - The TCTTE does not match the NIB, possibly because of an operational failure. Has the correct global catalog data set been used?
  - A terminal or session had RECOVOPT UNCONDREL|NONE specified.
  - A connection had PSRECOVERY NONE specified.
  - A matching mode group was not found. Have you got the correct global catalog data set?
  - A suitable session was not found, perhaps because the CNOS values created many "up for grabs" sessions which were in use when CICS failed. This situation would occur if the session limit was high and the contention winners was low. The situation might also occur if CICS was in the process of CNOSing from a high session limit to a low session limit at the time CICS failed. Message DFHZC0111 is issued in this case.

  - An URD was found for the session so the entire connection is unbound to allow the connection to recover correctly.
- APPC Sessions might be unbound by DFHZXPS for the following reasons. Some of the reasons are known states for which the session cannot be recovered. Others are unexpected errors.

  Known states for which the session cannot be recovered are as follows:
  - The last flow was a positive response to a bid with data.
  - Exchange log names (transaction CLS2) was running when the system failed.
  - A bind or bind security had not completed when the system failed.
  - Because of the last thing to flow, for example, SIGNAL, the state of the session at the time of system failure cannot be determined.

  Unexpected errors are as follows:
  - A bad return code was received from a call to DFGZGDA.
  - An attempt to reset the session from CS mode to CA mode or vice versa failed.
  - The TCTE_PRSS byte contained an unexpected value on entry to DFHZXPS.
  - The BIS, BID, or CV29 data pointed to by TCTV_PRSS_CV29_PTR contained an unknown value or was inconsistent.
  - An error occurred during some other part of the recovery process.
  - An internal logic error occurred in DFHZXPS.
- Sessions might be unbound by DFHZGDA for the following reasons:
  - A SEND issued as part of Deallocate(Abend) processing has failed.
  - A RECEIVE issued as part of Deallocate(Abend) processing has failed.
  - A logic error is detected during Deallocate(Abend) processing.
- Sessions might be unbound by DFHZXRC for the following reasons:
  - The user has specified RECOVOPT(RELEASESESS) and the session was in bracket at the time CICS failed.
  - End-Bracket and Clear/SDT cannot be used to clean up the session.
  - Cold Start has been requested for the session.
- Message DFHZC0124 can be issued with inconsistent counts if these conditions occur::
  - DFHZGRP loops or abends.
  - The ACB is closed by z/OS Communications Server operator commands while DFHZGRP is in control.
- LU6.2 connections, which might be expected to persist, might be unbound if a persistent resource is associated with the connection when CICS fails (that is, there was an asynchronous processing request in progress at the time CICS failed).
- Following a persistent sessions restart, LU6.2 partners might experience a series of unexpected abends with sense code 08640001 from the persisting CICS. This condition can occur either because there was a conversation in progress at the time CICS failed, and CICS has ended the conversation with this code, or for synclevel 2 conversations, the partner has attempted to initiate a conversation before Exchange Lognames has run following a persistent sessions restart.
- Some APPC sessions might hang following a persistent sessions restart because CICS has determined that it was in RECEIVE state at the time of the CICS failure, and issued a RECEIVE for the expected data, but the partner has not sent the expected data; the RECEIVE will not time out in this situation, because RTIMOUT does not apply to sends issued by DFHZGDA.

Persistent sessions status byte (TCTE_PRSS)

The byte TCTE_PRSS in the TCTTE tracks the stage reached in the persistent sessions recovery of a session. If, for some reason, persistent sessions recovery does not complete, this field can give a useful indication of the stage reached in recovery when the problem occurred.

TCTE_NO_PRSS_RECOVERY (X'00')
X'00' is the value that TCTE_PRSS normally contains. It means one of the following:

- Persistent sessions are not being used.
- The session was successfully recovered following a persistent sessions restart.
- The session has been closed using CLSDST and restarted since a persistent sessions restart.
- The session was started after any persistent sessions restart.

If this session was a persisting z/OS Communications Server session, TCTE_PRSS has been set to this value on completion of recovery notification for non-LU6.2 (see NAPES84 and NAPES83 routines), or in the session restarted logic of NAPES51 for LU6.2 sessions.

TCTE_NIB_MATCHED (X'01')
Placed in TCTE_PRSS by DFHZGRP after a TCTTE has been found which matches the NIB of a persisting z/OS Communications Server session. This value is a transient value, because the OPNDST OPTCD=RESTORE is issued soon after, which causes TCTE_PRSS to be updated.

TCTE_OPNDST_RESTORE_COMPLETED (X'02')
Placed in TCTE_PRSS after an OPNDST OPTCD=RESTORE has been successfully issued for a z/OS Communications Server Session by DFHZGRP. After this value has been placed in TCTE_PRSS, the TCTTE is put onto the activate scan queue to await processing by DFHZXRC or DFHZXPS.

TCTE_ZXRC_CLEANUP (X'20')
Placed in TCTE_PRSS by DFHZXRC when it begins processing a TCTTE. All TCTE_PRSS values relating to DFHZXRC processing are X'2x'. This value remains in TCTE_PRSS until the TCTTE is queued to DFHZNAC for the issuing of message DFHZC0146. If, for some reason, the TCTTE is not recovered and TCTE_PRSS contains this value, DFHZXRC might have a problem.

TCTE_ZXRC_ISSUE_RECOVERY_MSG (X'21')
DFHZXRC has identified the clean up and recovery actions required, and has queued the TCTTE to DFHZNAC for recovery message processing (message DFHZC0146). If any problem occurs with the recovery notification processing in DFHZNCA, TCTE_PRSS is likely to contain this value; possibly, the TCTTE has been taken off the DFHZACT or DFHZNAC queues for an unexpected reason.

TCTE_ZXPS_CLEANUP (X'30')
All TCTE_PRSS values beginning (X'3x') indicate that DFHZXPS is doing its recovery and clean up processing for this TCTTE. TCTE_PRSS is updated to this value on entering DFHZXPS for the first time. DFHZXPS only processes LU6.2 sessions.

TCTE_ZXPS_DEALLOCATE_ABEND (X'31')
DFHZXPS places this value into TCTE_PRSS before calling DFHZGDA for

the first time. It indicates that DFHZXPS has determined that an APPC conversation was taking place at the time CICS failed, and that DFHZXPS is calling DFHZGDA to stop that conversation. Again, this value is transient, because DFHZGDA updates TCTE_PRSS as it proceeds with its DEALLOCATE(ABEND) processing.

TCTE_ZXPS_SEND_IN_PROGRESS (X'32')
DFHZXPS has determined that bidding activity was taking place at the time CICS failed, and that some kind of SEND is required to complete the bid flows. If the session hangs with this value in TCTE_PRSS, a problem might have occurred with unexpected bid flows taking place.

TCTE_ZXPS_ISSUE_RECOVERY_MSG (X'33')
When DFHZXPS has completed recovery and clean up for the session, it puts this value into TCTE_PRSS before queueing the TCTTE to DFHZNAC for recovery message processing.

TCTE_ZGDA_FMH7_SEND (X'41')
All TCTE_PRSS values with X'4x' indicate that DFHZGDA is stopping the APPC conversation that was in progress on the session at the time CICS failed. This value indicates that DFHZGDA is in the process of issuing a SEND for the FMH7 that is to stop the conversation.

TCTE_ZGDA_FMH7_COMP (X'42')
DFHZGDA has completed its Deallocate(Abend) processing. This value in TCTE_PRSS indicates to DFHZXPS that it can continue with any outstanding recovery and clean up processing of its own.

TCTE_ZGA_FMH7_REC (X'43')
DFHZGDA has determined that CICS was in RECEIVE state at the time CICS failed, and has issued a RECEIVE for the RU expected from the partner. This value might appear in sessions that appear to be in an endless loop following a persistent sessions restart. If the partner does not issue the expected SEND, the RECEIVE is not run. Because this RECEIVE is issued under the TCP task, the RECEIVE is not subject to any RTIMEOUT.

TCTE_ZGDA_REC_EOC (X'44')
Placed in TCTE_PRSS if the first RECEIVE of the DFHZGDA module following the persistent sessions reveals that the partner is in the middle of sending a chain of RUs. If TCTE_PRSS contains this value, DFHZGDA has issued a RECEIVE_PURGE for the session. Again, depending on how quickly the partner sends the expected data, this session might appear to stop.

TCTE_ZGDA_SEND_RESP (X'45')
Placed in TCTE_PRSS if DFHZGDA has to issue a SEND for a response during Deallocate(Abend) processing.

TCTE_PRSS_CLSDST_SCHEDULED (X'FF')
This value is placed in TCTE_PRSS if an error occurs, or if, in the course of persistent sessions recovery, it is decided to stop the persisting session for one of a number of reasons:

- An error occurred issuing a SEND or RECEIVE during persistent sessions recovery.
- RECOVOPT(NONE) or RECOVOPT(UNCONDREL) was specified for the session.
- The only recovery action that DFHZXRC could take was to end the session.

The X'FF' value remains in TCTE_PRSS as an indicator that the session was ended during PRSS recovery. Only when the session is restarted is the value overwritten with X'00'.

Bid status byte (TCTE_BID_STATUS)

DFHZXPS uses a byte in the TCTTE, TCTE_BID_STATUS, to track the various stages of recovery. You can examine this byte to determine the stage of recovery reached by DFHZXPS.

The byte values have the following meanings:

- X'00'

  This session has not been processed by DFHZXPS.
- X'01' TCTE_SEND_POSITIVE_RESPONSE

  A positive response is to be sent to a bid that was received before system failure. This value is changed to X'07' TCTE_SENT_POSITIVE_RESPONSE before the TCTTE is requeued to DFHZACT for the SEND and so is only seen if DFHZXPS abends. When the response is sent DFHZXPS is recalled.
- X'02' TCTE_SEND_NEGATIVE_RESPONSE

  A negative response is to be sent to a bid with data that was sent before system failure. This response must be followed by RTR and so the status byte is changed to X'03' SEND_RTR before the TCTTE is requeued to DFHZACT for the SEND. This value is seen only if DFHZXPS abends. DFHZXPS is recalled when the response has been sent.
- X'03' TCTE_SEND_RTR

  Recovery is complete apart from the need to send RTR. This send is done by DFHZDET and DFHZXPS is not recalled.
- X'04' TCTE_SENT_RTR

  RTR was sent before system failure. No recovery is required. DFHZXPS is not recalled.
- X'05' TCTE_SEND_LUSTAT_EB

  Either a positive response to a bid was received, or a positive response was sent to RTR before the system failed. The bid now must be canceled. DFHZDET performs the cancellation and DFHZXPS is not recalled.
- X'06' TCTE_AWAITING_BB_RESPONSE

  A bid was sent before the system failed. No further recovery is required. When the response arrives from the partner, the bid is canceled. DFHZXPS is not recalled.
- X'07' TCTE_SENT_POSITIVE_RESPONSE

  Either a positive response has been sent to a bid or one is about to be sent (see TCTE_SEND_POSITIVE_RESPONSE). In the former case, DFHZXPS is not recalled, in the latter case, it is.
- X'08' TCTE_0814_RECEIVED

  A negative response was sent to a bid before the system failed. Any further recovery is carried out by DFHZDET and DFHZXPS will not be recalled.
- X'09' TCTE_0813_RECEIVED

  As for TCTE_0814_RECEIVED, except that no RTR is expected in this case. No further recovery processing is needed from either DFHZXPS or DFHZDET.
- X'0A' TCTE_SEND_RECOVERY_MESSAGE

  All recovery is now complete.
- X'0B' TCTE_DR1_OUTSTANDING

The last flow was inbound with CEB,RQD1 and so, although there is no task to ABEND, a response is still expected by the partner. DFHZSDL sends the response and any further recovery processing is done by DFHZDET. DFHZXPS is not recalled.

- X'0C' TCTE_DR1_EXPECTED

  As for TCTE_DR1_OUTSTANDING except that the last flow was inbound. DFHZDET arranges for the response to be received. DFHZXPS is not recalled.

TCTE_BID_STATUS must be used with TCTE_PRSS to determine the state of the recovery. If TCTE_PRSS is set to TCTE_ZXPS_ISSUE_RECOVERY_MESSAGE, or to a state that indicates that recovery is complete, DFHZXPS has finished processing. If not, DFHZXPS is recalled at a later stage.

Summary of persistent session waits

The DFHDSSRM waits are summarized here. They are all posted by DFHZGRP apart from PSUNBECB.

```
Module  Type Resource_name Resource_type ECB

DFHSII1 MVS  ZGRPECB       AP_INIT       TCTV_ZGRP_FIN_ECB

DFHZGUB OLDC PSUNBECB      ZC_ZGUB       WAIT_RPL_ECB

DFHZGRP MVS  PSOP1ECB      ZC_ZGRP       OPNDST_ECB

DFHZGRP MVS  PSOP2ECB      ZC_ZGRP       OPNDST_ECB

DFHZGRP MVS  PSINQECB      ZC_ZGRP       INQUIRE_ECB

DFHZGRP OLDC TCTVCECB      ZC_ZGRP       TCTVCECB
```

where the waits are issued for the following reasons:

ZGRPECB
: Wait for DFHZGRP to complete.

PSUNBECB
: Wait for free unbind RPL from RPL pool anchored from TCTV_PRSS_RPL_POOL_PTR.

PSOP1ECB
: Wait for OPNDST RESTORE to complete.

PSOP2ECB
: Wait for OPNDST RESTORE to complete after UNBINDs have failed.

PSINQECB
: Wait for INQUIRE PERSESS to complete.

TCTVCECB
: Wait for TCTTEs to finish installing (DFHTCRP).

z/OS Communications Server exits

The z/OS Communications Server exits SYNAD (DFHZSYX) or LERAD (DFHZLEX) might be driven during persistent sessions recovery.

In DFHZGRP, before INQUIRE OPTCD=PERSIST is issued, or in DFHZGUB before CLSDST or TERMSESS are issued, CICS sets the RPL user field to -2 to indicate to the exits that they must do *no* processing at all, because these macros might be issued under the concurrent TCB.

In DFHZGRP, before OPNDST OPTCD=RESTORE is issued, CICS sets the RPL user field to -1 to indicate to the exits that they must try minimum recovery; that is, they set the return code to TCZSYXPR if an error can be retried, or TCZSYXCF if it is a permanent error.

If an error occurs in DFHZGSL for SETLOGON OPTCD=PERSIST, DFHZSYX returns immediately (as for RPL user field = -2).

If MNPS is in use and the Communications Server crashes, DFHZTPX is driven with a code of 8. If the system initialization parameter PSTYPE=MNPS was specified, DFHZTPX does *not* schedule the autoinstalled TCTTEs for deletion. They are scheduled for CLSDST CLEANUP instead by DFHZSHU.

See *z/OS Communications Server: SNA Programming* for general Communications Server exit information.

Trace

The trace point IDs AP FB10 through AP FBFF, for which the trace levels are TC 1 and TC 2, are provided for persistent sessions recovery.

These trace point IDs relate to the persistent sessions recovery modules DFHZGCA, DFHZGCC, DFHZGCN, DFHZGDA, DFHZGPC, DFHZGPR, DFHZGRP, DFHZGSL, DFHZGUB, DFHZCGRP, DFHZXPS, and DFHZXRC.

Statistics

The following statistics are produced by DFHZGRP. They are treated in the same way as other terminal control z/OS Communications Server statistics.

A03_PRSS_NIB_COUNT
: The number of active z/OS Communications Server sessions when INQUIRE OPTCD=COUNTS was issued; this value represents the number of persisting sessions.

A03_PRSS_INQUIRE_COUNT
: The number of times DFHZGRP issues INQURE OPTCD=PERSESS. Each INQUIRE is given about 400 sessions.

A03_PRSS_UNBIND_COUNT
: The number of times CLSDST or TERMSESS were issued by DFHZGUB.

A03_PRSS_OPNDST_COUNT
: The number of sessions that OPNDST RESTORE restored successfully.

A03_PRSS_ERROR_COUNT
: The number of sessions, with NIBUSER=tctte address, that z/OS Communications Server failed to restore with OPNDST RESTORE. This value is incremented if z/OS Communications Server operator commands are issued while DFHZGRP is in control and sessions are closed as a result.

Chapter 3. CICS executable modules

CICS executable modules are used to help in diagnosing problems with CICS.

For each module in the list, click the module name to see the following information:

- The name of the module
- Its entry points
- Callers of the module
- A brief description of the module
- Where the module returns to; this information is omitted where the module returns to its caller (the normal situation).

In general, this list is restricted to non-OCO modules. In the few cases where OCO modules are included, no design details are given.

DFHACP

Entry points

DFHACPNA

Called by

DFHAPRM, DFHAPXME

Description

The abnormal condition program writes a message to the terminal and to the CSMT destination if a transaction abends or cannot be started. Subject to tests on the type of terminal, DFHACP invokes DFHMGP to output the message. It calls DFHPEP and, depending on the result, may disable the transaction. For each error, there is an entry in a table which contains the number of the message to be written to the principal facility (terminal) and the number of the message to be written to CSMT. If, in either case, there is no message, zero is entered.

The main subroutines of DFHACP are:

```
ABCSMTWT - Write to CSMT
ACPCALMG - Use DFHMGP to output a message
ACPCLPEP - Invoke DFHPEP
ACPFENTY - Identify message for terminal
TERMERR  - Terminal error.
```

DFHAICBP

Entry points

DFHAICB

Called by

User application program

© Copyright IBM Corp. 1997, 2017

Description

The application interface control block program acts both as a control block and, for compatibility with early releases of CICS/VS, as executable code. DFHAICBP provides addressability between application programs and CICS entry points, namely those of the EXEC interface and the common programming interface.

DFHAICBP is link-edited with the EXEC interface programs (DFHEIG, DFHEIP, and the prolog and epilog modules), and the common programming interface program (DFHCPI) to form the application interface program (DFHAIP) load module.

DFHALP

Entry points

DFHALPNA

Called by

DFHCRQ, DFHCRS, DFHICP, DFHTPQ, DFHTPR, DFHTPS, DFHZATI, DFHZISP, DFHZNAC, DFHZTSP

Description

The terminal allocation program contains the logic to allocate TCTTE resources to requesting transactions. The request operates in a multiple exchange between the requesting transaction and terminal control. DFHALP passes a SCHEDULE request to terminal control as an ATI terminal control, then responds with an AVAIL command. The requests are represented by AIDs (AID chain manipulations being performed by calls to DFHALP). For LU6.2, DFHALP issues a terminal control allocate mode name macro.

DFHAMP

Entry points

DFHAMPNA

Called by

DFHEIP, DFHSII1

Description

The allocation management program is invoked by the CEDA transaction. It analyzes commands and calls the definition file management program, DFHDMP, to process changes to records in the CSD. For the INSTALL command, DFHAMP also calls program manager, transaction manager, and DFHSPP. DFHPUP is called to convert data between address list format and the CSD record format.

DFHAPJC

Entry points

DFHAPJCN

Called by

User

Description

The AP domain journal control gate service module handles WRITE_JOURNAL_DATA calls made by the user exit's XPI. It gets a TCA if the task doesn't currently have one, and also a JCA. If the task already has a JCA, this is stacked. It then copies the parameter list passed in the domain call, to the JCA, and then issues one of four journal writes, depending on the request. Finally the return code from the JC write is copied into the domain parameter list, and the JCA and TCA are released if they were obtained by DFHAPJC.

DFHAPSIP

Entry points

DFHSIPNA

Called by

DFHAPDM

Description

The main AP domain initialization program provides DFHWTO support and common subroutines used by DFHSIA1 through DFHSIJ1. In sequence, DFHAPSIP performs the following functions:

- Defines the AP domain subpools
- Acquires the SIT address
- Passes control to the DFHSIA1, DFHSIB1, and so on.

The main subroutines of DFHAPSIP are:

```
CHKRLVLR - Check release level
OVERLSUP - Overlay supervisor
SIGETCOR - Storage allocation
SILOADR  - Program loader
SIPCONS  - Console WRITE.
```

DFHAPRX

Entry points

Called by

DFHISP, DFHMIRS

Description

DFHAPRX performs logical transformations of data for transaction routing requests to remote CICS systems using IP interconnectivity (IPIC) over TCP/IP connections.

DFHAPST

Entry points

DFHAPST

Called by

DFHEIP, DFHSTST

Description

The supervisory statistics program within the AP domain accepts a request for and then supervises the copying/resetting of statistics counters in the AP domain by calling the appropriate DFHSTxx modules to access the counters.

This module is called when:

- Statistics domain is collecting INTERVAL statistics and calls this module to pass it copies of and to reset all statistics in AP domain. This module then sequentially calls all of the DFHSTxx modules to do the copying and resetting.
- A CEMT PERFORM STATISTICS command results in a call to the statistics domain which then makes an appropriate call to this module to pass it copies of the requested statistics. This module then calls the DFHSTxx modules required to do the copying.
- An EXEC CICS COLLECT STATISTICS command results in a call to this module which then calls the DFHSTxx module required to pass copies of the statistics back to the application program.

Thus, this module is called only by the statistics domain or by DFHEIP.

This module provides two functions:

COLLECT_STATISTICS
: collects statistics for all resources in the AP domain and calls the statistics domain to write them out to the SMF data set.

COLLECT_RESOURCE_STATS
: collects statistics for the named resource type (optionally qualified by the resource identifier) and either copies them to a buffer available through the API, or causes them to be written to the SMF data set.

DFHAPTD

Entry points

DFHAPTD

Called by

DFHETD, DFHTDA, DFHTDB, ME domain

Description

DFHAPTD handles DFHTDTDM macro requests; as such, it provides the transient data gate into the AP domain. DFHTDTDM macro requests are routed from DFHAPTD to DFHTDP using the corresponding DFHTD CTYPE requests.

DFHAPTI

Entry points

DFHAPTI

Called by

the timer domain to handle NOTIFY calls for the application domain.

Description

The DFHAPTO module looks at the token passed by the timer domain and resumes either the DFHAPTI or DFHAPTIX module, as appropriate.

DFHAPTIM

Entry points

DFHAPTIM

Called by

runs as a system task attached by the DFHSII1 module.

Description

The DFHAPTIM module is part of the interval control mechanism. When it first gets control, it suspends itself to wait for an interval control ICE to expire. Interval control uses the timer domain to handle time intervals. When the timer domain detects the expiry of an interval control related interval, it calls the DFHAPTI module, which in turn resumes the DFHAPTIM module. The DFHAPTIM module then makes an “expiry analysis” call to the DFHICP module, which processes any expired ICEs. On return, the DFHAPTIM module suspends itself again to wait for the next ICE to expire.

DFHAPTIX

Entry points

DFHAPTIX

Called by

runs as a system task attached by the DFHSII1 module.

Description

The DFHAPTIX module is part of the interval control mechanism. When it first gets control, it tells the timer domain that it wants to be told every time it is midnight. It then suspends itself to wait for the next midnight. When that occurs, the timer domain calls the DFHAPTI module, which resumes the DFHAPTIX module, which in turn calls the DFHICP module to do midnight processing.

DFHASV

Entry points

DFHASVNA

Called by

DFHCSVC

Description

DFHASV is one of the modules that run under the CICS type 3 SVC. On entry to DFHASV, register 0 contains one of the following request codes:

```
  0 - Paging request
  8 - SRB termination
  9 - HPO initialization
 24 - Monitoring services
 64 - Authorize general purpose subtask TCB
 80 - Issue SDUMP
136 - Bind AP domain.
```

DFHBSIB3

Entry points

DFHBSIB3

Called by

DFHTBSxx

Description

DFHBSIB3 adds BMS 3270 support to a TCT table entry.

DFHBSIZ1

Entry points

DFHBSIZ1

Called by

DFHTBSxx

Description

DFHBSIZ1 adds SCS support to a TCT table entry.

DFHBSIZ3

Entry points

DFHBSIZ3

Called by

DFHTBSxx

Description

DFHBSIZ3 adds DFHZCP 3270 support to a TCT table entry.

DFHBSMIR

Entry points

DFHBSMIR

Called by

DFHTBSxx

Description

DFHBSMIR builds a TCT table entry for a session.

DFHBSMPP

Entry points

DFHBSMPP

Called by

DFHTBSxx

Description

DFHBSMPP builds a TCT table entry for a pipeline pool entry.

DFHBSM61

Entry points

DFHBSM61

Called by

DFHTBSxx

Description

DFHBSM61 builds sessions for an LU6.2 mode group.

DFHBSM62

Entry points

DFHBSM62

Called by

DFHTBSxx

Description

DFHBSM62 builds the mode entry for an LU6.2 mode group.

DFHBSS

Entry points

DFHBSS

Called by

DFHTBSxx

Description

DFHBSS adds a new connection (system entry) to a CICS system.

DFHBSSA

Entry points

DFHBSSA

Called by

DFHTBSxx

Description

DFHBSSA initializes DFHKCP support in a new TCT system entry.

DFHBSSF

Entry points

DFHBSSF

Called by

DFHTBSxx

Description

DFHBSSF initializes the statistics counters in a new TCT system entry.

DFHBSSS

Entry points

DFHBSSS

Called by

DFHTBSxx

Description

DFHBSSS builds security support for a new TCT system entry.

DFHBSSZ

Entry points

DFHBSSZ

Called by

DFHTBSxx

Description

DFHBSSZ builds z/OS Communications Server interface support for a new TCT system entry.

DFHBSSZB

Entry points

DFHBSSZB

Called by

DFHTBSxx

Description

DFHBSSZB adds a new batch interregion connection to a CICS system.

DFHBSSZG

Entry points

DFHBSSZG

Called by

DFHTBSxx

Description

DFHBSSZG adds a new advanced program-to-program communication (APPC) single-session connection to a CICS system.

DFHBSSZI

Entry points

DFHBSSZI

Called by

DFHTBSxx

Description

DFHBSSZI adds an indirect terminal control system table entry to a CICS system.

DFHBSSZL

Entry points

DFHBSSZL

Called by

DFHTBSxx

Description

DFHBSSZL adds a local terminal control system table entry to a CICS system.

DFHBSSZM

Entry points

DFHBSSZM

Called by

DFHTBSxx

Description

DFHBSSZM introduces a new connection (system) to ZCP.

DFHBSSZP

Entry points

DFHBSSZP

Called by

DFHTBSxx

Description

DFHBSSZP builds an advanced program-to-program communication (APPC) parallel-session connection to a CICS system.

DFHBSSZR

Entry points

DFHBSSZR

Called by

DFHTBSxx

Description

DFHBSSZR builds an MRO session entry.

DFHBSSZS

Entry points

DFHBSSZS

Called by

DFHTBSxx

Description

DFHBSSZS builds an advanced program-to-program communication (APPC) session entry.

DFHBSSZ6

Entry points

DFHBSSZ6

Called by

DFHTBSxx

Description

DFHBSSZ6 builds an LU6.1 connection entry.

DFHBST

Entry points

DFHBST

Called by

DFHTBSxx

Description

DFHBST performs TCTTE initialization common to terminals, pipeline pool entries, and sessions for IRC and ISC.

DFHBSTB

Entry points

DFHBSTB

Called by

DFHTBSxx

Description

DFHBSTB adds support for BMS to a new TCT terminal or session entry.

DFHBSTBL

Entry points

DFHBSTBL

Called by

DFHTBSxx

Description

DFHBSTBL adds support for logical device components (LDCs).

DFHBSTB3

Entry points

DFHBSTB3

Called by

DFHTBSxx

Description

DFHBSTB3 adds partition support to a new TCT terminal or session entry.

DFHBSTC

Entry points

DFHBSTC

Called by

DFHTBSxx

Description

DFHBSTC performs those operations that are executed after the installation of a terminal.

DFHBSTD

Entry points

DFHBSTD

Called by

DFHTBSxx

Description

DFHBSTD adds data interchange program (DFHDIP) support for a new TCT table entry.

DFHBSTE

Entry points

DFHBSTE

Called by

DFHTBSxx

Description

DFHBSTE adds EXEC diagnostic facility (EDF) support for a new TCT table entry.

DFHBSTH

Entry points

DFHBSTH

Called by

DFHTBSxx

Description

DFHBSTH initializes EXEC interface fields for a new TCT table entry.

DFHBSTI

Entry points

DFHBSTI

Called by

DFHTBSxx

Description

DFHBSTI adds interval control program (DFHICP) support for a new TCT table entry.

DFHBSTM

Entry points

DFHBSTM

Called by

DFHTBSxx

Description

DFHBSTM adds message generation program (DFHMGP) support for a new TCT table entry.

DFHBSTO

Entry points

DFHBSTO

Called by

DFHTBSxx

Description

DFHBSTO is the spooler builder.

DFHBSTP3

Entry points

DFHBSTP3

Called by

DFHTBSxx

Description

DFHBST adds 3270-copy support for a new TCT table entry.

DFHBSTS

Entry points

DFHBSTS

Called by

DFHTBSxx

Description

DFHBSTS adds signon program (DFHSNP) support for a new TCT table entry.

DFHBSTT

Entry points

DFHBSTT

Called by

DFHTBSxx

Description

DFHBSTT adds task control program (DFHKCP) support for a new TCT table entry.

DFHBSTZ

Entry points

DFHBSTZ

Called by

DFHTBSxx

Description

DFHBSTZ builds a session or terminal resource.

DFHBSTZA

Entry points

DFHBSTZA

Called by

DFHTBSxx

Description

DFHBSTZA adds DFHZCP activity scan support to a new TCT terminal or session entry.

DFHBSTZB

Entry points

DFHBSTZB

Called by

DFHTBSxx

Description

DFHBSTZB appends or deletes a BIND image for a TCT terminal or session entry.

DFHBSTZC

Entry points

DFHBSTZC

Called by

DFHTBSxx

Description

DFHBSTZC adds a single-session LU6.2 system as an advanced program-to-program communication (APPC) terminal.

DFHBSTZE

Entry points

DFHBSTZE

Called by

DFHTBSxx

Description

DFHBSTZE sets error message writer fields for a new TCT table entry.

DFHBSTZH

Entry points

DFHBSTZH

Called by

DFHTBSxx

Description

DFHBSTZH adds an interregion (IRC) batch session to a CICS system.

DFHBSTZL

Entry points

DFHBSTZL

Called by

DFHTBSxx

Description

DFHBSTZL adds logical device code support to a new TCT terminal or session entry.

DFHBSTZO

Entry points

DFHBSTZO

Called by

DFHTBSxx

Description

DFHBSTZO adds an MVS console to a CICS system.

DFHBSTZP

Entry points

DFHBSTZP

Called by

DFHTBSxx

Description

DFHBSTZP adds a pipeline pool entry to a CICS system.

DFHBSTZR

Entry points

DFHBSTZR

Called by

DFHTBSxx

Description

DFHBSTZR adds an interregion (IRC) session to a CICS system.

DFHBSTZS

Entry points

DFHBSTZS

Called by

DFHTBSxx

Description

DFHBSTZS adds an advanced program-to-program communication (APPC) session to the terminal control program.

DFHBSTZV

Entry points

DFHBSTZV

Called by

DFHTBSxx

Description

DFHBSTZV adds the parts of a terminal or session TCT table entry that are special to z/OS Communications Server and IRC.

DFHBSTZZ

Entry points

DFHBSTZZ

Called by

DFHTBSxx

Description

DFHBSTZZ adds a non-APPC session to the TCT. (APPC is advanced program-to-program communication.)

DFHBSTZ1

Entry points

DFHBSTZ1

Called by

DFHTBSxx

Description

DFHBSTZ1 adds support for a remote terminal to a CICS system.

DFHBSTZ2

Entry points

DFHBSTZ2

Called by

DFHTBSxx

Description

DFHBSTZ2 adds support for a remote advanced program-to-program communication (APPC) connection.

DFHBSTZ3

Entry points

DFHBSTZ3

Called by

DFHTBSxx

Description

DFHBSTZ3 adds a 3270 to the TCT.

DFHBSXGS

Entry points

DFHBSXGS

Called by

DFHBSMIR, DFHZTSP

Description

DFHBSXGS generates a unique session name for an LU6.2 TCTTE.

DFHBSZZ

Entry points

DFHBSZZ

Called by

DFHTBSxx

Description

DFHBSZZ adds a terminal or session to the TCT.

DFHBSZZS

Entry points

DFHBSZZS

Called by

DFHTBSxx

Description

DFHBSZZS adds a new session to LU6.2 support.

DFHBSZZV

Entry points

DFHBSZZV

Called by

DFHTBSxx

Description

DFHBSZZV adds a z/OS Communications Server terminal or session to the TCT.

DFHCAPB

Entry points

DFHCAPNA

Called by

DFHTCRP

Description

DFHCAPB processes command analysis for z/OS Communications Server terminal definitions contained in a load module table DFHRDTxx for TCT migration.

DFHCCNV

Entry points

DFHCCNV

Called by

DFHCHS, DFHMIRS

Description

DFHCCNV provides conversion of user data from ASCII to EBCDIC and from EBCDIC to ASCII for function-shipped requests from external clients. DFHCCNV is called from either the LU2 remote server program DFHCHS or the mirror program DFHMIRS, for EXEC CICS requests and replies originating from the identified server or mirror. For any function-shipped request it is invoked twice, once on the inbound side and once on the outbound path. DFHCCNV is passed the EXEC CICS parameter list by its caller. On the request side, this occurs after DFHCHS or DFHMIRS has called transformer 2 but before DFHEIP is invoked. On the response side, this occurs after DFHEIP returns to DFHCHS or DFHMIRS but before transformer 3 is invoked.

DFHCMP

Entry points

DFHCMPNA

Called by

DFHETR

Description

The CICS monitoring compatibility module is invoked by the old event monitoring point of EXEC CICS ENTER TRACEID to interface to the monitoring domain.

DFHCPY

Entry points

DFHCPYNA

Called by

DFHPRK

Description

The 3270 copy program (transaction CSCY) causes data to be copied from screen to printer in a (z/OS Communications Server) 3270 system. DFHCPY is invoked by DFHPRK (only if the 3270 has the copy feature) and issues a DFHTC TYPE=COPY macro to the printer. DFHCPY then initiates DFHRKB.

DFHCRC

Entry points

DFHCRCNA

Called by

MVS

Description

The interregion abnormal exit module is a CICS module that contains an ESTAE exit to terminate interregion communication in abnormal conditions. DFHCRC issues a CLEAR request to the interregion SVC.

DFHCRNP

Entry points

DFHCRNNA

Called by

DFHCRSP, dispatcher

Description

DFHCRNP, the connection manager (transaction CSNC), controls IRC connections. It establishes and breaks these connections and processes inbound requests to attach tasks (for example, mirror) to communicate with connected systems.

DFHCRQ

Entry points

DFHCRQNA

Called by

transaction CRSQ

Description

The remote schedule page program is invoked periodically to delete requests to attach a transaction on a remotely owned terminal if those requests have been outstanding for more than the ATI purge delay interval.

DFHCRR

Entry points

DFHCRRNA

Called by

DFHCRNP

Description

The interregion session recovery program performs session recovery on behalf of primary or secondary IRC sessions.

DFHCRS

Entry points

DFHCRSNA

Called by

transaction CRSR

Description

The remote scheduler program builds and ships AIDs for automatic transaction initiation when the terminal is in a remote address space. It receives requests to schedule an AID shipped to it from a remote address space.

DFHCRSP

Entry points

DFHCRSNA

Called by

DFHEIP, DFHSIJ1

Description

The interregion communication startup module can be invoked, either at system initialization or by a CEMT request, in order to make the CICS address space available for communication by other address spaces. DFHCRSP issues a logon request to the interregion communication SVC routine and attaches transaction CSNC (DFHCRNP).

DFHCRT

Entry points

DFHCRTNA

Called by

transaction CXRT

Description

DFHCRT is the relay program used when a transaction attempts to allocate a conversation to a remote advanced program-to-program (APPC) terminal.

DFHCSA

Entry points

DFHCSANA

Called by

Not applicable

Description

The DFHCSA module contains the common system area (CSA) and CSA optional features list, the queue control area (QCA) and, for HPO systems, the SRB interface control area.

DFHCSDUP

Entry points

DFHCUCNA

Called by

MVS

Description

The CSD utility program is an offline program that provides services for the CSD. The utility command processor (DFHCUCP) validates commands and invokes the appropriate routine to execute the requested function. DFHCSDUP calls DFHDMP to access the CSD.

DFHCSSC

Entry points

DFHCSSNA

Called by

DFHSIJ1, DFHSNSN, DFHSUSN, DFHTCRP, DFHZCUT

Description

DFHCSSC, the signon timeout program, is invoked as a system task by DFHSIJ1 and DFHTCRP to perform XRF takeover sign-off timeout processing. It is invoked elsewhere as the CSSC transaction for timeout processing of the following:

- Terminals signed on with the TIMEOUT option
- Entries in the internally managed signon table (SNT)

- Entries in the local userid tables (LUITs).

The CSSC transaction is scheduled when task termination determines that a timeout is necessary. When DFHCSSC is executed, it examines all signed-on terminals, all entries in the SNT managed by DFHTMP, and all entries in the LUITs. It signs off or deletes expired entries as appropriate, and then reschedules itself to perform later timeouts if required.

DFHCSVC

Entry points

DFHCSVC

Called by

MVS

Description

This module is a type 3 SVC that passes control to the various required routines, dependent on the parameter passed to it. On a first request for a particular function, it loads the required module and puts its address in the AFCB and then branches to that code. Further calls result in the address in the AFCB being branched to.

Returns to

Type 3 SVC

DFHCSVCU

Entry points

DFHCSVCU

Called by

MVS

Description

The DFHCSVCU utility program dynamically installs the CICS TYPE 3 SVC, removing the need to IPL the z/OS system. This utility replaces a single SVC entry in the SVC table with a pointer to the entry point of the supplied module.

Note: You can use the DFHCSVCU utility program only if you have the required authorization.

DFHCSVCU installs TYPE 3 SVCs only, and the module specified on the PARM statement must be loaded into the link pack area (LPA) before the utility is run. You can load the module by using the following MVS console command:

```
SETPROG LPA,ADD,MODNAME=module,DSNAME=data set
```

- *module* is the name of the program to be invoked by the new SVC.
- *data set* is the load library that contains the module.

Specify the SVC to be replaced and the module name to be used in the JCL PARM statement for this program:

```
//SVCUPD EXEC PGM=DFHCSVCU,PARM=('SVCnnn=progname')
```

- *nnn* is the number of the SVC to be replaced, in the range 200-255.
- *progname* is the program to be invoked by the new SVC. This program must already be loaded into the LPA.

Returns to

MVS

DFHCUCAB

Entry points

DFHCUCA

Called by

DFHCAPB

Description

The resource definition online command analyzer interprets a z/OS Communications Server resource definition in command form and produces a parameter list.

DFHCUCB

Entry points

DFHCUCB

Called by

DFHCUCP

Description

The resource definition online command builder receives commands and transforms them to a format for use by the command processors.

DFHCUCCB

Entry points

DFHCUCC

Called by

DFHCAPB

Description

This program extracts a single entry from a loaded RDT table containing z/OS Communications Server resource definitions for TCT migration.

DFHCUCDB

Entry points

DFHCUCD

Called by

DFHCAPB

Description

The resource definition online command default values program modifies the parameter list produced by DFHCUCAB by inserting the default values.

DFHCWTO

Entry points

DFHCWTNA

Called by

CWTO transaction

Description

The console write-to-operator module is a CICS-supplied transaction that allows an operator to send a message to the console operator. DFHCWTO issues SVC 35 (WTO) to pass the message to the operator's console.

DFHDBAT

Entry points

AENTRY

Called by

DFHERM, IMS database resource adapter (DRA).

Description

This program provides a mapping between the external architectures of CICS (the resource manager interface (RMI)) and of DBCTL (the database resource adapter (DRA)). Both are independently defined and different. DFHDBAT is part of the support for the CICS-DBCTL interface and runs in an application program environment.

DFHDBAT is invoked by a DFHRMCAL request through the CICS RMI. The RMI supplies DFHDBAT with a parameter list from which DFHDBAT constructs the

DRA INIT, DRA TERM, and DRA THREAD parameter lists. It must also transform the DRA parameter list back, after a DL/I call, to the format expected by CICS. Thus, DFHDBAT is also referred to as the CICS-DBCTL adapter-transformer.

DFHDBAT is defined as a threadsafe program.

DFHDBCON

Entry points

DFHDBCON

Called by

DFHDBME

Description

This program issues a CICS-DBCTL interface connection request to the CICS-DBCTL adapter-transformer, DFHDBAT. DFHDBCON is part of the support for the CICS-DBCTL interface and runs in an application program environment.

DFHDBCR

Entry points

DFHDBCR

Called by

DFHSII1 via attach

Description

DFHDBCR is the CICS/DBCTL XRF tracking program. DFHDBCR runs in an alternate CICS system during the tracking phase. DFHDBCR receives messages from the active CICS system regarding the state of the connection to DBCTL, and drives the XXDFB and XXDTO exits and takes appropriate action.

DFHDBCT

Entry points

DFHDBCT

Called by

DFHDBCTX, DFHDBAT

Description

This program processes any elements placed on the CICS-DBCTL control work element (CWE) chain. DFHDBCT is part of the support for the CICS-DBCTL interface and runs in an application program environment. It is invoked when the CICS-DBCTL connection program, DFHDBCON, attempts to connect to DBCTL.

The program then issues a wait. The DFHDBCT program is posted whenever an element is placed on the CWE chain.

DFHDBCTX

Entry points

DFHDBCTX

Called by

DFHDBAT

Description

This program notifies the CICS-DBCTL control transaction of changes in the state of the CICS-DBCTL interface. DFHDBCTX is part of the support for the CICS-DBCTL interface. It does not run in a CICS environment and thus does not use any CICS services. This exit is invoked by the DBCTL adapter on behalf of the DBCTL DRA.

DFHDBDI

Entry points

DFHDBDI

Called by

DFHDBCT

Description

This program disables the CICS-DBCTL adapter program and cleans up the storage used by the CICS-DBCTL interface programs. DFHDBDI is part of the support for the CICS-DBCTL interface and runs in an application program environment. DFHDBDI is invoked by the CICS/VS DBCTL control program, DFHDBCT, just before it terminates.

DFHDBDSC

Entry points

DFHDBDSC

Called by

DFHDBCT, DFHDBME

Description

This program issues a CICS-DBCTL interface disconnection request to the CICS-DBCTL adapter-transformer. DFHDBDSC is part of the support for the CICS-DBCTL interface and runs in an application program environment.

DFHDBIQ

Entry points

DFHDBIQ

Called by

CDBI transaction

Description

This program is the CDBI CICS-supplied transaction. Its function is to inquire on the current status of the CICS-DBCTL interface. DFHDBIQ is part of the support for the CICS-DBCTL interface.

DFHDBME

Entry points

DFHDBME

Called by

CDBC transaction

Description

This program is the CDBC CICS-supplied transaction. Its function is to provide a front end for making certain changes to the status of the CICS-DBCTL interface. DFHDBME is part of the support for the CICS-DBCTL interface.

DFHDBMOX

Entry points

DFHDBMOX

Called by

DFHDBAT

Description

This program outputs monitoring information supplied by DBCTL to the monitoring domain, using monitoring domain services. The information is supplied by DBCTL when it has processed a PSB schedule request and a thread termination request. This exit forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment. This exit is invoked by the CICS-DBCTL adapter.

DFHDBP

Entry points

DFHDBPNA

Called by

DFHAPRC

Description

This program invokes DWE processors when a UOW backs out.

DFHDBREX

Entry points

DFHDBREX

Called by

DFHDBAT

Description

This program is the CICS-DBCTL resume exit. The resume exit is driven whenever the adapter or the DRA requires to resume a task which they have suspended. This exit forms part of the support for the CICS-DBCTL interface. It does not run in a CICS environment and thus cannot use CICS services.

DFHDBSPX

Entry points

DFHDBSPX

Called by

DFHDBAT

Description

This program is the CICS-DBCTL suspend exit. The suspend exit is driven whenever the adapter or the DRA requires to suspend a task. DFHDBSPX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment.

DFHDBSSX

Entry points

DFHDBSSX

Called by

DFHDBAT

Description

DFHDBSSX is the CICS/DBCTL status exit. In the event of a DRA thread failure, DFHDBSSX is called to transfer ownership of PCB storage to CICS. When the task

ends, DFHDBSSX is called to release this storage.

DFHDBSTX

Entry points

DFHDBSTX

Called by

DFHDBAT

Description

This program is the CICS-DBCTL statistics exit. The exit outputs CICS-DBCTL session termination statistics to the statistics domain. DFHDBSTX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment, but it can also be invoked during CICS orderly termination. This exit is invoked by the CICS-DBCTL adapter.

DFHDBTOX

Entry points

DFHDBTOX

Called by

DFHDBAT

Description

This program is the CICS-DBCTL token exit. The function of this exit is to provide the CICS-DBCTL adapter with task tokens for tasks that have not been through the DBCTL call processor ,DFHDLIDP, or the DBCTL connection program, DFHDBCON, or the DBCTL disconnection program, DFHDBDSC, where task tokens are usually generated. DFHDBTOX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment. This exit is invoked by the CICS-DBCTL adapter.

DFHDBUEX

Entry points

DFHDBUEX

Called by

DFHDBCT, DFHDBDSC

Description

DFHDBUEX is the user-replaceable CICS-DBCTL exit program. It is invoked whenever CICS successfully connects to DBCTL and whenever CICS disconnects from DBCTL. DFHDBUEX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment.

DFHDCP

Entry points

DFHDCPNA

Called by

DFHDC macro, DFHEDC

Description

DFHDCP translates DFHDC macro requests for a transaction dump to DU domain TRANSACTION_DUMP calls.

DFHDES

Entry points

DFHDESNA

Called by

DFHZEV1, DFHZEV2, DFHZOPN

Description

DFHDES performs data encryption and bind-time security.

DFHDIP

Entry points

DFHDIPNA

Called by

DFHACP, DFHDI macro, DFHEDI, DFHKCP, DFHMCP, DFHTOM, DFHZEMW, DFHZRSP, DFHZSUP

Description

The data interchange program acts as a function manager when transactions want to communicate with batch devices using SNA support. DFHDIP builds and receives FMHs, which control the data set selection and function currently being performed by the batch device.

The main subroutines of DFHDIP are:

```
DESTCHEK - Destination change
D1ABORTE - Abort
D1CONRTE - Continue
D1ENDRTE - End
D1INARTE - Transaction attach
D1INPRTE - Input
D1NOTRTE - Note
D1QUERTE - Query.
```

DFHDLI

Entry points

DFHDLINA

Called by

User application, DFHMIRS, DFHSPP

Description

DFHDLI is the DL/I call router program. It decides which call processor is to be used for the request: DBCTL or REMOTE. It then invokes the appropriate processor: DFHDLIDP or DFHDLIRP.

DFHDLIAI

Entry points

ASMTDLI, CBLTDLI, PLITDLI

Called by

User application using DL/I CALL interface

Description

This module is used by the CICS-DL/I interface. It is link-edited with the application program to provide D/I communication between the application and the CICS-DL/I interface routine DFHDLI. Calls for DL/I to the ASMTDLI, CBLTDLI, or PLITDLI entry points are resolved by this processor.

DFHDLIDP

Entry points

DFHDLIDP

Called by

DFHDLI

Description

DFHDLIDP is the DBCTL call processor. It services DL/I calls for PSBs that are owned by a DBCTL subsystem, and invokes the DL/I task-related user exit (adapter) to interface with DBCTL.

DFHDLIRP

Entry points

DFHDLIRP

Called by

DFHDLI

Description

DFHDLIRP is the remote call processor. It services DL/I calls that are function-shipped to another CICS system.

DFHDMP

Entry points

DFHDMPNA

Called by

DFHAMP, DFHCSDUP

Description

The definition file management program handles physical changes to the CSD. The main processes in DFHDMP are:

```
BUILDKWA (DM16)  - Build key work area
CONNECT (DM01)   - CONNECT
CREATSET (DM11)  - Create SET
DELETE (DM05)    - DELETE
DISCONN (DM02)   - DISCONNECT
ENDBRO (DM10)    - End BROWSE
ERASESET (DM12)  - Delete SET
GETNEXT (DM09)   - Get next record
LOCK (DM06)      - LOCK
QUERYSET (DM13)  - QUERYSET
READ (DM04)      - Read CSD control records
RELSEKWA (DM17)  - Free key work area
SETBRO (DM08)    - Set browse
UNLOCK (DM06)    - UNLOCK
WRITE (DM03)     - WRITE.
```

DFHDRPG

Entry points

DFHDRPNA

Called by

DFHEIP

Description

DFHDRPG is the EXEC interface processor for EXEC DLI commands for database sharing. It receives the parameters of the command and from them builds a list that is appropriate to call DFHDRPE, the program request handler. On return from DFHDRPE, the status code in the PCB is examined. For some codes, an MVS abend is executed; the other codes are passed back to the application program.

DFHDSBA$, DFHDSB1$

Entry points

DFHDSBNA

Called by

DFHPBP

Description

The data stream build program produces the final device-dependent data stream for each page of BMS output. It is invoked only for processing data streams that are not in 3270 format. DFHDSB removes blanks from the ends of lines, converts logical new-line characters into the device-dependent equivalents (adding idle characters where necessary), and inserts horizontal and vertical tab characters if supported.

DFHDU710

Entry points

DFHDUPNA

Called by

MVS

Description

The dump utility program formats and prints transaction dumps from a CICS transaction dump data set (DFHDMPA or DFHDMPB). The transaction dumps are written to the data set by the dump domain. They contain information about the state of a particular transaction at the time of a transaction abend or user-requested dump.

DFHDXACH

Entry points

DFHDXACH

Called by

DFHDBCR, DFHDBCT

Description

DFHDXACH is a stub that is also MVS-attached, and which branches to an input address.

DFHDXSTM

Entry points

DFHDXSTM

Called by

DFHDBCT, DFHDBCR

Description

DFHDXSTM is used to attach, detach, and inquire on MVS subtasks attached by DFHDBCR and DFHDBCT.

DFHDYP

Entry points

DFHDYP

Called by

DFHAPRT

Description

This is the system-provided (default) dynamic routing program invoked from the CICS relay program (DFHAPRT) when a remote transaction is defined as being dynamic.

DFHEAG

Entry points

DFHEG1

Called by

AMODE(64) assembler user application

Description

DFHEAG is a stub that is link-edited with an assembler-language AMODE(64) application program to provide communication with DFHEIG. The command-language translator turns each EXEC CICS command into a call statement. The external entry point that is invoked by the call is resolved to an entry point in this stub. The address of the entry point in DFHEIG (DFHEIGCN) is found through a chain of system and CICS control blocks.

DFHEAG0

Entry points

DFHEAG0

Called by

AMODE(64) assembler user application

Description

DFHEAG0 is a stub that is link-edited with an assembler-language AMODE(64) application program to provide communication with the module in the EXEC interface layer for the prolog and epilog calls generated by the command-language translator in the application program. The external entry point invoked by the calls is resolved to an entry point in this stub. The address of the entry point in this module is found using a chain of system and CICS control blocks.

DFHEAI

Entry points

DFHEI1

Called by

User application

Description

DFHEAI is a stub that is link edited with an assembler-language application program to provide communication with DFHEIP. The command-language translator turns each EXEC CICS command into a call statement. The external entry point invoked by the call is resolved to an entry point in this stub. The address of the entry point in DFHEIP (DFHEIPCN) is found through a chain of system and CICS control blocks.

The DFHEAI stub must be included at the beginning of the program in the output from the link edit. To achieve this, ORDER and INCLUDE statements for DFHEAI must be in the link-edit step of your JCL. When you use the CICS-supplied assembler procedure DFHEITAL in the SDFHPROC library to translate, assemble, and link-edit application programs written in assembler language, the COPYLINK step of this procedure copies SDFHMAC(DFHEILIA). DFHEILIA contains the following statements that must be included:

```
ORDER DFHEAI
INCLUDE SYSLIB(DFHEAI)
```

The statements are put into a temporary file that is concatenated before the assembled application program in the LKED step of the procedure.

DFHEAI0

Entry points

DFHEAI0

Called by

User application

Description

This is a stub that is link-edited with an assembler-language application program to provide communication with DFHEIPA, part of the EXEC interface layer, for the prolog and epilog calls generated by the command-language translator in the application program. The external entry point invoked by the calls is resolved to an entry point in this stub. The address of the entry point in DFHEIPA (DFHEIPAN) is found using a chain of system and CICS control blocks.

DFHEAP1$

Entry points

PREPROC

Description

The assembler-language translator module performs the following functions:

- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:

```
 0 - no message
 4 - warning
 8 - error
12 - severe error
16 - translator failure.
```

- Replaces CICS commands by invocations of the DFHECALL macro, and inserts invocations of DFHEIENT, DFHEIRET, DFHEISTG, and DFHEIEND macros at appropriate places.
- Inserts diagnostics resulting from errors in commands, as comments in the output program that are not listed on the listing file.

DFHEBF

Entry points

DFHEBFNA

Called by

DFHEIP, DFHEIG

Description

DFHEBF is the EXEC interface processor for the field edit built-in function, DEEDIT.

DFHEBU

Entry points

DFHEBUNA

Called by

DFHETL, DFHETC

Description

The EXEC function management header (FMH) construction module is called by DFHETC when a SEND or CONVERSE command is being processed, and ATTACH function management headers have to be built and concatenated ahead of user data.

DFHECI

Entry points

DFHEI1

Called by

User application

Description

This is a link-edit stub similar to DFHEAI, except that it is used for COBOL application programs.

DFHECID

Entry points

DFHEIN01

Called by

DFHECIP

Description

The command interpreter module analyzes CECI commands, and manages its displays. It uses the EXEC interface to invoke other CICS functions.

DFHECIP

Entry points

DFHEIN00

Called by

CECI transaction

Description

The command interpreter program performs preliminary validation and initialization for the CECI transaction, and links to DFHECID.

DFHECP1$

Entry points

PREPROC

Description

The COBOL translator module performs the following functions:

- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:

```
 0 - no message
 4 - warning
 8 - error
12 - severe error
16 - translator failure.
```

- Inserts DFHEIBLK and COMMAREA declarations in the LINKAGE section.
- Inserts the EIB definition in the LINKAGE section.
- Inserts the DIB definition (for DL/I HLPI) in the WORKING_STORAGE section.
- In the PROCEDURE DIVISION, the translator inserts a USING clause in the DIVISION statement, and replaces all CICS and DL/I commands by COBOL CALL statements.
- Inserts diagnostics resulting from any errors in commands, as messages in the translator listing file.

DFHEDAD

Entry points

DFHESP01

Called by

DFHEDAP

Description

The resource definition online (RDO) transactions module analyzes the commands, and manages the displays for CEDA, CEDB, and CEDC. It uses the EXEC interface.

DFHEDAP

Entry points

DFHESP00

Called by

CEDA, CEDB, CEDC transaction

Description

The resource definition online (RDO) transactions program performs preliminary validation and initialization for CEDA, and links to DFHEDAD.

Returns to

DFHEIP

DFHEDC

Entry points

DFHEDCNA

Called by

DFHEIP, DFHEIG

Description

DFHEDC is the EXEC interface processor for dump commands.

DFHEDFBR

Entry points

DFHEDFBR

Called by

CEBR transaction, DFHEDFD

Description

The temporary-storage browse transaction browses, copies, or deletes entries in a temporary-storage queue. It interprets commands and PF key actions.

DFHEDFD

Entry points

DFHEDFD

Called by

DFHEDFP

Description

The EDF display program is invoked from DFHEDFP to analyze and display the current status of the user program. DFHEDFD stores control information about a temporary-storage message queue and uses BMS to format the display screen. DFHEDFD interfaces with other CICS control programs using the EXEC interface.

DFHEDFM

Description

The EDF map set contains BMS maps used by DFHEDFD to format the EDF display.

DFHEDFP

Entry points

DFHEDFNA

Called by

transaction CEDF

Description

The EDF main program is the control program for EDF. DFHEDFP can be invoked in one of two ways:

1. Directly from the EDF display terminal by entering the CEDF transaction identification
2. By pressing the user-defined PF key.

DFHEDFP is also attached by DFHEDFX as the main program of the EDF task.

DFHEDFR

Entry points

DFHEDFNA

Called by

Not applicable

Description

The EDF response table contains a description of the exception responses for each EXEC command and the abend codes associated with error responses. DFHEDFR is used by DFHEDFD to interpret the responses obtained from an EXEC command.

DFHEDFX

Entry points

DFHEDFNA

Called by

DFHACP, DFHEIP, DFHEIG, program manager

Description

The EDF task switch program is invoked from DFHACP, DFHEIP, DFHEIG, or program manager when a program is running in debug mode. DFHEDFX suspends the user task and attaches the debugging task, passing it information about the user task in the TWA of the debugging task.

DFHEDI

Entry points

DFHEDINA

Called by

DFHEIP, DFHEIG

Description

DFHEDI is the EXEC interface processor for data interchange commands.

DFHEDP

Entry points

DFHEDPNA

Called by

DFHERM

Description

DFHEDP converts command-level DL/I statements into a call parameter list acceptable to DL/I. In addition, it provides 31-bit application support by moving segment I/O areas above and below the 16 MB line as required.

DFHEDP is defined as a threadsafe program.

DFHEDP1$

Entry points

PREPROC

Description

The C translator module performs the following functions:

- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:

```
 0 - no message
 4 - warning
 8 - error
12 - severe error
16 - translator failure.
```

- Inserts the EIB definition at the head of the translated output.
- If the DLI translator option is specified, inserts the DIB definition
- Replaces all CICS and DL/I commands in the input program by function calls (dfhexec) in the output program.
- Inserts diagnostics from any errors in commands, as messages on the translator listing file.

DFHEEI

Entry points

DFHEEINA

Called by

DFHEIP, DFHEIG

Description

DFHEEI is the EXEC interface processor for DFHEIP or DFHEIG ADDRESS, ASSIGN, PUSH, POP, and HANDLE commands.

DFHEEX

Entry points

DFHEEXNA

Called by

DFHETC

Description

The EXEC function management header (FMH) extraction module is called by DFHETC when a RECEIVE or CONVERSE command is being processed, and when data has to be extracted from ATTACH function management headers.

DFHEFRM

Entry points

DFHEFRM

Called by

DFHDBP, DFHSPP

Description

DFHEFRM is the EXEC interface file control syncpoint processor. At syncpoint commit or rollback time, DFHEFRM deletes the FFLE entries that were created by DFHFCEI for the task.

DFHEGL

Entry points

DFHEGLNA

Called by

DFHEIP, DFHEIG

Description

DFHEGL is the EXEC interface processor for unmapped LU6.2 commands.

DFHEIDTI

Entry points

DFHEIDTI

Called by

DFHEIP, DFHEIG

Description

DFHEIDTI is the EXEC interface processor for ASKTIME and FORMATTIME. DFHEIDTI updates the time and date fields in the EIB and certain time fields in the CSA, and returns the current time, or date, to the application.

DFHEIG

Entry points

DFHEIGCN

Called by

AMODE(64) application programs

Description

DFHEIG is the initial EXEC interface module called by AMODE(64) applications. See “EXEC interface” on page 1570 for further information.

DFHEIIC

Entry points

DFHEICNA

Called by

DFHEIP

Description

DFHEIIC is the EXEC interface processor for interval control commands.

Exits

DFHEIIC has the following global user exit points:
XICERES

DFHEIFC

Entry points

DFHEIFC

Called by

DFHEIP

Description

DFHEIFC is the file control EXEC interface module, providing an interface between DFHEIP and file control. It locates the AFCTE, and performs the security check. For a remote file, DFHEIFC passes the request to a transformer, which then ships the request to the other system. For a local file, DFHEIFC converts the EXEC argument list to an FCFR parameter list (as defined by the DFHFCFRA DSECT) and calls DFHFCFR, the file control file request handler. After the request completes, DFHEIFC builds return code information in the EIB.

DFHEIP

Entry points

DFHEIPNA

Called by

Application programs

Description

DFHEIP is the main EXEC interface module called by AMODE(24) and AMODE(31) application programs. See "EXEC interface" on page 1570 for further information.

DFHEIPA

Entry points

DFHEIPAN

Called by

DFHEAI0

Description

DFHEIPA is part of the EXEC interface layer. It acquires and partially initializes the DFHEISTG dynamic storage when called from the DFHEIENT macro in an assembler language AMODE(24) or AMODE(31) application program. It frees this storage when called from the DFHEIRET macro.

DFHEISC

Entry points

DFHEISC

Called by

DFHEIP, DFHEIG

Description

DFHEISC is the EXEC interface processor for all storage control commands, including 64-bit storage. DFHEISC replaces the previous DFHESC module.

DFHEISP

Entry points

DFHESPNA

Called by

DFHEIP, DFHEIG

Description

DFHEISP is the EXEC interface processor for syncpoint commands.

DFHEISR

Entry points

DFHEISR

Called by

DFHEDI, DFHEGL, DFHEIQMS, DFHEMS, DFHEOP, DFHETC, DFHETL, DFHTDB, DFHXFFC, DFHXFX

Description

DFHEISR obtains buffers and copies data for the calling EXEC interface modules, at the location and in the storage key required by the application.

DFHEJC

Entry points

DFHEJCNA

Called by

DFHEIP, DFHEIG

Description

DFHEJC is the EXEC interface processor for journaling commands.

DFHEKC

Entry points

DFHEKCNA

Called by

DFHEIP, DFHEIG

Description

DFHEKC is the EXEC interface processor for task control commands.

DFHELII

Entry points

DFHEI1

Called by

User application

Description

This is a link-edit stub similar to DFHEAI, except that it is used for C application programs.

DFHEMS

Entry points

DFHEMSNA

Called by

DFHEIP, DFHEIG

Description

DFHEMS is the EXEC interface processor for BMS commands.

DFHEMTA

Entry points

DFHEMT00

Called by

User application

Description

The master terminal programmed interface program is a special version of DFHEMTP that a user application can link to for master terminal services.

DFHEMTD

Entry points

DFHEMT01

Called by

DFHEMTA, DFHEMTP, DFHEOTP, DFHESTP

Description

The master terminal module analyzes the commands, and manages displays for CEMT, CEOT, and CEST transactions. It uses the EXEC interface.

DFHEMTP

Entry points

DFHEMT00

Called by

CEMT transaction

Description

The master terminal program performs preliminary validation and initialization for the CEMT transaction, and links to DFHEMTD.

DFHEOTP

Entry points

DFHEMT00

Called by

CEOT transaction

Description

The master terminal program performs preliminary validation and initialization for the CEOT transaction, and links to DFHEMTD.

DFHEPC

Entry points

DFHEPCNA

Called by

DFHEIP, DFHEIG

Description

DFHEPC is the EXEC interface processor for program control commands.

DFHEPI

Entry points

DFHEI1

Called by

User application

Description

This is a link-edit stub similar to DFHEAI, except that it is used for PL/I application programs.

DFHEPP1$

Entry points

PREPROC

Description

The PL/I translator module performs the following functions:

- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:

```
 0 - no message
 4 - warning
 8 - error
12 - severe error
16 - translator failure.
```

- If the input program is a MAIN procedure, inserts DFHEIPTR as the first parameter on the PROCEDURE statement to address the EIB. The translator also inserts declarations of the EIB and certain temporary variables.

- Replaces all CICS and DL/I commands in the input program by CALL statements in the output program.
- Inserts diagnostics from any errors in commands, as messages on the translator listing file.

DFHEPS

Entry points

DFHEPSNA

Called by

DFHEIP

Description

DFHEPS is the link between DFHEIP and the JES interface program, DFHPSP.

DFHERM

Entry points

DFHERMNA

Called by

DFHEIP

Description

DFHERM is called by DFHEIP on behalf of the other components of CICS to manage the connection between CICS and non-CICS products.

DFHESTP

Entry points

DFHEMT00

Called by

CEST transaction

Description

The master terminal program performs preliminary validation and initialization for the CEST transaction, and links to DFHEMTD.

DFHETC

Entry points

DFHETCNA

Called by

DFHEIP, DFHEIG

Description

DFHETC is the EXEC interface processor for terminal control commands.

DFHETD

Entry points

DFHETDNA

Called by

DFHEIP, DFHEIG

Description

DFHETD is the EXEC interface processor for transient data commands. The EXEC requests are routed from DFHETD to DFHTDP using the corresponding DFHTD CTYPE requests.

DFHETL

Entry points

DFHETLNA

Called by

DFHETC

Description

DFHETL is the EXEC interface processor for mapped LU6.2 commands.

DFHETR

Entry points

DFHETRNA

Called by

DFHEIP, DFHEIG

Description

DFHETR is the EXEC interface processor for trace commands.

DFHETS

Entry points

DFHETSNA

Called by

DFHEIP, DFHEIG

Description

DFHETS is the EXEC interface processor for temporary-storage commands.

DFHEXI

Entry points

DFHEXINA

Called by

DFHZARQ

Description

The exceptional input program is invoked from DFHZCP when unexpected input is received from a z/OS Communications Server 3270 terminal that has a task attached. DFHEXI checks whether the input is the result of a 3270 print function key being pressed; if so, DFHEXI issues a DFHTC TYPE=PRINT macro, and then unlocks the keyboard; in any case, DFHEXI then passes control back to DFHZCP.

DFHFCAT

Entry points

DFHFCAT

Called by

DFHFCDN, DFHFCN

Description

DFHFCAT processes inquire and update requests on the state of the backup while open (BWO) attributes in the ICF catalog for VSAM data sets, and inquires on the quiesce state in the ICF catalog.

DFHFCBD

Entry points

DFHFCBD

Called by

DFHFCFR

Description

DFHFCBD handles BDAM file control requests except for OPEN and CLOSE.

DFHFCDN

Entry points

DFHFCDN

Called by

DFHAMFC, DFHAMPFI, DFHEIQDN, DFHEIQDS, DFHFCLF, DFHFCMT, DFHFCN, DFHFCRC, DFHFCRO, DFHFCRD, DFHFCRP

Description

DFHFCDN builds data set name blocks at cold start or in response to CEDA requests. It also examines or modifies data set name blocks in response to EXEC CICS INQUIRE or EXEC CICS SET commands.

DFHFCDTS

Entry points

DFHFCDTS

Called by

DFHFCFR

Description

DFHFCDTS processes file control requests to access data table records for READ-ONLY requests against CICS-maintained tables, and for all record requests against user-maintained tables. It calls data table services to retrieve or modify table records, calls DFHFCVS to retrieve data from the VSAM source data set if it is not in the table, and calls DFHFCDTX to function ship requests that cannot be satisfied by sharing.

DFHFCFR

Entry points

DFHFCFR

Called by

DFHAPLI, DFHAPSM, DFHDTLX, DFHDMPCA, DFHEIFC, DFHERM, DFHFCDTS, DFHFCFR, DFHFCFS, DFHFCRC, DFHFCRP, DFHUEH

Description

DFHFCFR is the central module in the file control component. It handles file control requests issued by DFHFCEI (requests from application programs), or by other CICS modules (internal file control requests). DFHFCFR ensures that the file is both opened and enabled, acquires an FRTE as necessary, performs request validity checking, and then routes the request to the appropriate access-method dependent module (DFHFCBD for BDAM, DFHFCVS for non-RLS VSAM and also for update or browse requests against a CICS-maintained data table, DFHFCRS for RLS VSAM, and DFHFCDTS for all other data table requests).

DFHFCFS

Entry points

DFHFCFS

Called by

DFHAMFC, DFHDMPCA, DFHDMRM, DFHDTLX, DFHEIQDS, DFHFCDTS, DFHFCFR, DFHFCLF, DFHFCQU, DFHFCRC, DFHFCRD, DFHFCRU, DFHFCSD, DFHFCU, DFHFCVS

Description

DFHFCFS changes the state of a file. It invokes DFHFCN to open, or close, files.

DFHFCL

Entry points

DFHFCLNA

Called by

DFHFCN

Description

DFHFCL is a file control program that is link-edited into DFHFCFS. DFHFCL builds and deletes VSAM LSR pools. It is called by DFHFCN with a parameter list that specifies the pool number (1 through 8) and the action to be taken (build or delete).

DFHFCM

Entry points

DFHFCMNA

Called by

DFHFCFS

Description

DFHFCM is a file control program that is link-edited into DFHFCFS. When records are added via a VSAM path, DFHFCM is called to open the base associated with the path.

DFHFCMT

Entry points

DFHFCMT

Called by

DFHAFMT, DFHAMFC, DFHAMPFI, DFHDMPCA, DFHEDFX, DFHEIQDS

Description

DFHFCMT builds file control table entries in response to CEDA commands. It also examines or modifies FCT entries in response to EXEC CICS INQUIRE or EXEC CICS SET commands.

DFHFCN

Entry points

DFHFCNNA

Called by

DFHFCFS

Description

DFHFCN is a file control program that is link-edited into DFHFCFS. DFHFCN opens and closes files. If a file has not been allocated, DFHFCN allocates it, and frees it on closure.

DFHFCRL

Entry points

DFHFCRL

Called by

DFHAMFC

Description

DFHFCRL modifies SHRCTL blocks (describing VSAM LSR pools) in response to CEDA requests.

DFHFCRP

Entry points

DFHFCRP

Called by

DFHFCIN2

Description

The file control restart program builds the file control environment and initializes file control.

DFHFCSD

Entry points

DFHFCSD

Called by

DFHSTP

Description

DFHFCSD is called during CICS controlled shutdown processing to close all open files managed by CICS file control.

DFHFCST

Entry points

DFHFCST

Called by

DFHSTFC, DFHSTLS

Description

DFHFCST is called to collect or reset file or LSRPOOL statistics on request from DFHSTFC or DFHSTLS.

DFHFCU

Entry points

DFHFCUNA

Called by

CSFU transaction

Description

DFHFCU issues an OPEN for files specified in the file control table (FCT). This program examines the FCT, and calls DFHFCFS to open all specified files.

DFHFCVR

Entry points

DFHFCVR, UPADEXIT

Called by

DFHFCBD, DFHFCFR, DFHFCVR, DFHFCVS, VSAM

Description

DFHFCVR is a file control program that is link-edited into DFHFCVS. It handles requests to VSAM, and also contains the VSAM UPAD exit.

DFHFCVS

Entry points

DFHFCVS

Called by

DFHFCDTS, DFHFCFR

Description

DFHFCVS handles requests for file control services made against VSAM files. These services include:

- Communication with files defined in the file control table
- Logging of changes to these files by DFHFCJL and the log manager.
- Syncpoint services.

DFHFDP

Entry points

DFHFDPNA

Called by

DFHFD macro

Description

DFHFDP translates DFHFD macro requests for a system dump to DU domain SYSTEM_DUMP calls.

DFHFEP

Entry points

DFHFEPNA

Called by

CSFE transaction

Description

The FE terminal test program can be used to send a complete character set to a terminal or to echo input or to turn tracing on or off. This program is an application program and does not exit to any other CICS modules. However it does use CICS facilities.

DFHGMM

Entry points

DFHGMMNA

Called by

DFHKCP

Description

The “good morning” program is invoked by the CSGM system transaction to write a “good morning” message to z/OS Communications Server logical units when a satisfactory OPNDST has occurred (and if the message has been requested in the TYPETERM resource definition or the TCT TYPE=TERMINAL macro).

DFHHPSVC

Entry points

IGCnnn

Called by

DFHZHPSR (via an SVC call)

Description

This is a type 6 SVC module used only on MVS. Its sole purpose is to cause MVS to dispatch an SRB. DFHHPSVC provides part of the CICS high performance option (HPO) code, and is invoked only if HPO is in use. In the entry point name, nnn is the number of the SVC.

Returns to

MVS

DFHICP

Entry points

DFHICPNA

Called by

DFHEIIC, DFHIC macro

Description

The interval control program is used for time management and has two main functions:

1. Services DFHIC macros under the control of a requesting task's TCA
2. Detects the expiration of time-dependent events, as defined in ICEs.

The main subroutines of DFHICP are:

```
ICCANCLN - Cancel a time-ordered request
ICEXPANL - Time expiration analysis
ICGTIMEN - Current time of day
ICGTTTDM - Data retrieval
ICICECRN - Build basic ICE
ICPCTSN  - Task initiation
ICPOSTN  - Signal expiration of a specified time
ICRESETN - Time of day clock reset support
ICSCHEDN - ICE schedule
ICWAITN  - Delay processing of a task.
```

DFHIIPA$, DFHIIP1$

Entry points

DFHIIPNA

Called by

DFHMCP

Description

The non-3270 input mapping program performs all BMS input mapping functions for all devices except the 3270. On exit from the module, the input data has been mapped into a newly acquired TIOA that is returned to the application program and is then addressable using BMS DSECTs in the application.

The main subsections of DFHIIP are:

```
IIMID  - GETMAINs TIOA to return to user, and maps
         page buffer into it using specified map.
IIREAD - Reads input data, issuing DFHTC or DFHDI
         requests to get data from the terminal.
IISCAN - Scans data stream for device-dependent
         control characters and creates page
         buffer.
```

DFHIRP

Entry points

DFHIRPNA

Called by

DFHCRC, DFHCRNP, DFHCRSP, DFHDRPD, DFHDRPE, DFHDRPF, DFHSRP, DFHSTP, DFHZCX

Description

The interregion communication program is used to pass data from one region to another in the same MVS image. The programs being run in the regions are usually CICS programs, but DFHIRP does not assume this.

DFHIRW10

Entry points

As defined in interest ladder [3]

Called by

DFHIRP

Description

The interregion work exit delivers work to the IRC control task (CSNC). DFHIRW10 is called whenever DFHIRP has work to deliver to a system that logged on with DFHIRW10 as its interregion work exit. This module checks whether the work being delivered to the target system requires that work be enqueued on CSNC; if so, it enqueues the work and posts CSNC. DFHIRW10 is invoked in access register (AR) mode and user key.

DFHISP

Entry points

DFHISPNA

Called by

DFHDLI, DFHEIP, DFHEIFC

Description

The intersystem communication program is invoked when a request to access a resource has to be shipped to a remote system (through ISC or MRO).

These requests are passed to DFHISP:

- File control

3. **Interest ladder**: ladder within DFHIRW10 that expresses interest in all types of MRO work.

- Interval control
- Temporary storage
- Transient data
- DL/I

DFHISP controls the acquisition, use, and freeing of a session to the remote system, and invokes DFHXFP or DFHXFX to process requests and replies. Two user exits are provided in DFHISP: XISCONA can be used to control the queuing of requests from DFHISP to allocate intersystems sessions and XISLCLQ can be used to override the LOCALQ option of the transaction attributes. XISCONA is invoked for any function-shipping requests that cannot be processed immediately. XISLCLQ supports the local queuing of function-shipped START NOCHECK requests when the link to the remote system is out of service. If a START NOCHECK request is queued, DFHISP starts the CMPX transaction when the link is brought into service.

DFHISFS

Entry points

Called by

DFHISIS

Description

DFHISFS performs logical transformations of data that is shipped to remote CICS systems using IP interconnectivity (IPIC) over TCP/IP connections.

DFHISXF

Entry points

Called by

DFHISP, DFHMIRS

Description

DFHISXF performs logical transformations of data for DPL, START, and CANCEL requests to remote CICS systems using IP interconnectivity (IPIC) over TCP/IP connections.

DFHJCP

Entry points

DFHJCPNA

Called by

DFHEJC, DFHJC macro

Description

The journal control program (DFHJCP) either processes a request to get a JCA control block, or has been called to write to a journal. In the latter case it examines the information in the JCA that is passed with the request and decides whether to call the recovery manager or the log manager based on whether it finds journalname DFHLOG in the JCA or not. There are three separate calls to the DFHLGGL gate of the log manager: one for a write, a put or a wait request. The same is true for the recovery manager calls, which use the DFHRMRE gate. In addition there is a call to this gate for requests which have keypoint record data with them.

When control returns from either of these domains, the domain's outcome is mapped onto a valid return code which is put into the JCA before control returns back to the calling program

DFHJUP

Entry points

DFHJUPNA

Called by

MVS

Description

The journal print utility program examines, selects, and displays data in QSAM data sets, such as the CICS and IMS logs. Data selection is controlled by input parameters, and an optional user exit. DFHJUP provides access to the MVS log streams via the SUBSYS keyword in the JCL.

DFHKCP

Entry points

DFHKCPNA

Called by

DFHEKC, DFHKC macro

Description

This is a startup routine that passes control to either DFHXCP or DFHXCPC. It also deals with some ENQ and DEQ calls.

DFHKCQ

Entry points

DFHKCQNA

Called by

DFHXCPC

Description

DFHKCQ processes DFHKC INITIALIZE, REPLACE, WAITINIT, and DISCARD macro calls to the transaction manager.

DFHKCRP

Entry points

DFHKCRP

Called by

DFHKCP (attaches DFHKCRP as a CICS task)

Description

DFHKCRP is the task control restart program.

DFHKCSC

Entry points

DFHKCSC

Called by

DFHKCQ

Description

This module forms part of the transaction manager. It provides the QUERY_TRANSACTION function for use in determining whether the transaction specified on a DISCARD TRANSACTION command can validly be discarded. DFHKCSC examines the ICE chain, the AID chains, and the SIT, looking for references to the transaction that is the subject of the DISCARD.

DFHKCSP

Entry points

DFHKCSPA, DFHKCSPI, DFHKCSPD, DFHKCSPF, DFHKCSPP

Description

The task SRB control program is part of the high performance option (HPO) code available on CICS on MVS. It runs in SRB mode and resides in protected storage.

DFHLUP

Entry points

DFHLUPNA

Description

DFHLUP is the LU6.2 services manager. It initializes and shuts down a network, and resynchronizes flows.

DFHMCPA$, DFHMCPE$, DFHMCP1$

Entry points

DFHMCPNA

Called by

DFHBMS macro, DFHEMS

Description

The mapping control program processes DFHBMS macro requests and completes the processing of a logical message when a task terminates without issuing a DFHBMS TYPE=PAGEOUT. DFHMCP's main function is to analyze DFHBMS requests and to pass control to the appropriate modules. Other functions include the loading of maps and partition sets, and scheduling of output messages transmitted by temporary storage.

The main subsections of DFHMCP are:

```
MCPCPO   - Completes logical message build message
           control record for temporary storage
MCPDWEXT - DWE processing, invoked by DFHKCP to
           complete BMS processing at application
           termination
MCPINPT  - Handles all input requests
MCPIN    - TYPE=IN (EXEC CICS RECEIVE MAP)
MCPMAPLO - Loads map set and locates map
MCPPGBLD - TYPE=PAGEBLD|TEXTBLD (EXEC SEND TEXT)
MCPPGOUT - TYPE=PAGEOUT (EXEC CICS SEND PAGE)
MCPPURGE - TYPE=PURGE (EXEC CICS PURGE MESSAGE)
MCPROUTE - TYPE=ROUTE (EXEC CICS ROUTE).
```

DFHMCX

Entry points

DFHMCXNA

Called by

DFHMCP

Description

DFHMCX is the BMS fast path module for standard and full-function BMS, and the program for minimum BMS support. It is called by DFHMCP if the request satisfies one of the following conditions:

- It is a noncumulative direct terminal send map or receive map issued by a command-level program.
- It is for a 3270 display or an LU3 printer which does not support outboard formatting. If the terminal supports partitions, it is in the base state.
- The CSPQ transaction has been started.
- The message disposition has not changed.

DFHMGP

Entry points

DFHMGPNA

Called by

DFHACP, DFHCRQ, DFHCRT, DFHEOP, DFHFEP, DFHRTC, DFHRTE, DFHZEMW, DFHZERH, DFHZIS1, DFHZTSP, DFHZXRL

Description

The message generation program provides an interface for sending CICS messages to the terminal user.

DFHMGT

Entry points

DFHMGTNA

Called by

DFHMGP

Description

The message prototype control table, or message generation table, consists of a series of copybooks, DFHMGTnn, each of which contains up to 100 messages that are issued by DFHMGP.

DFHMIRS

Entry points

DFHMIRNA

Called by

Task initiation

Description

The mirror program is called when a request to access a resource is received from a remote CICS system with an IPIC over TCP/IP, MRO, or ISC over SNA connection. DFHMIRS controls the receipt of requests and transmission of replies. For non IPIC there is only one transaction which calls the mirror program, but for IPIC connections the Listener task, CIEP handles control flows and attaches a mirror task as required.

DFHMIRS processes requests from these systems:

- MRO-connected systems
- LU6.1 connected systems
- LU6.2 sync level 1 connected systems
- LU6.2 sync level 2 connected systems
- IPIC-connected systems

DFHMIRS is defined as a threadsafe program. For IPIC connections only, CICS runs DFHMIRS on an L8 open TCB whenever possible. To enable file control operations to run on the open TCB as well, which can produce performance benefits, you must specify the system initialization parameter **FCQRONLY=NO** for the CICS system where DFHMIRS runs. For other connection types, CICS does not run DFHMIRS on an open TCB.

For a non-IPIC connection the input to DFHMIRS consists of a TCTTE representing the session between CICS and its session partner and a TIOA containing the function shipping request. The TIOA is passed to DFHXFP (transformer 2) for conversion of the request from transmission format to the parameter list format required for DFHEIP or DFHDLI. If the data requires conversion (transaction CPMI), an EXEC CICS LINK is used to link to the data conversion program DFHCCNV, passing a COMMAREA that contains the EXEC CICS parameter list for the request where applicable. DFHMIRS then passes the request to DFHEIP or DFHDLI for execution.

On return from DFHEIP or DFHDLI, the data conversion program is called to convert the reply (if applicable), and then the transformer program DFHXFP (transformer 3) is called to convert the reply parameter list to transmission format. DFHMIRS then determines the DFC to send with the reply and transmits the reply to the requesting system. If the mirror task has modified protected resources, it continues receiving requests and transmitting replies until a sync point request is received from the remote system.

DFHML1

Entry points

DFHML1NA

Called by

DFHMCP, DFHPBP

Description

The SCSPRT logical unit type 1 output mapping routine is called by DFHPBP to build a page of data stream from a chain of map and application data structure

copies. The data contains only features that the TTP says are supported by the target terminal. This routine is called when NLEOM is specified for 3270 printers or LU3 printers.

The main subsections of DFHML1 are:

ML1SPACE
: Calculate space for chaining and mapping

ML1FMCA
: Format the chains that describe the maps

ML1PF
: Process map fields

DFHMROQP

Entry points

DFHMRONA

Called by

DFHCRNP, DFHCRSP

Description

The MRO work queue enable/disable program is invoked by the DFHMROQM macro for ENABLE and DISABLE requests (other requests are processed by an inline expansion). DFHMROQP is called by DFHCRSP to enable the MRO work queues when starting interregion communication, and by either DFHCRSP or DFHCRNP to disable the work queues when stopping interregion communication. MRO work queues are used to deliver work to the IRC control task (CSNC).

DFHMSP

Entry points

DFHMSPNA

Called by

CMSG transaction

Description

The message switching program routes a message entered at the terminal to one or more operator-defined terminals or to other operators. DFHMSP can be used in conversational mode to process operands entered from separate input operations. In this case the operands already processed are preserved in temporary storage.

The main sections and subroutines of DFHMSP are:

```
MSBMSRT  - Check for complete operands
MSCNVRS  - Issue conversational response
MSCONTIN - Process conversational response
MSMSG4   - MSG operand
MSNTRY   - Process operands
MSROUTE  - Route operand.
```

DFHMXP

Entry points

DFHMXPNA

Called by

Automatic transaction initiation

Description

The local queuing shipper provides the means of transferring to a remote system a START request that has been temporarily deferred by use of the local queuing option.

DFHM32A$, DFHM321$

Entry points

DFHM32NA

Called by

DFHMCP, DFHPBP

Description

For a BMS output request, the 3270 mapping program generates the appropriate data stream for a 3270 device, and returns control to DFHPBP which invokes the DFHTPP module to send the data to the appropriate destination, which is either to the direct terminal, or to temporary storage, or back to the caller. For a BMS input request, the data stream from a 3270 device is examined and mapped into a user application TIOA format.

The main subsections of DFHM32 are:

```
BMFMHTST - Create beginning of 3270 data stream
           (FMH cursor positioning)
BMMID    - Input mapping
BMMMS    - Merge maps (output mapping)
M32PF    - Process field.
```

DFHPBPA$, DFHPBP1$

Entry points

DFHPBPNA

Called by

DFHMCP

Description

The page and text build program positions maps or text, including header or trailer maps or text, within a page of output. For non-3270 devices, the module

creates a page buffer containing the user's data which is then passed to DFHDSB to produce a device-dependent data stream. When mapping, this includes merging the data supplied by the application with the constant data included in the map. For 3270 devices, copies of the maps and application-supplied data for a page are chained together, to be processed by module DFHM32, to produce a 3270 data stream. The page and text build program creates dummy maps, and chains them in the same way for 3270 text building. For LU1 printers with extended attributes, copies of the maps and application-supplied data for a page are chained together, to be processed by module DFHML1 to produce an SCS data stream. The page and text build program creates dummy maps, and chains them in the same way for text building. After the maps have been processed by DFHDSB, DFHM32, or DFHML1, DFHPBP calls DFHTPP to write them out.

The main subroutines of DFHPBP are:

PBDOUTPT
: Mapping/text build complete, decide whether to call data stream generator and which one (DFHDSB or DFHM32). Return to caller (DFHMCP)

PBD00005
: Main control logic, request analysis.

PBD01000
: Map placement logic (3270 and non-3270 mapping).

PBD01130
: Non-3270 mapping.

PBD10000
: Pageout routine.

PBD11000
: Modify field positions within map (used by 3270 and non-3270 mapping).

PBD20000
: Text processing (3270 and non-3270).

PBD30000
: 3270 mapping.

PBFMHBLD
: Build FMH if FMHPARM specified (non-3270 text and map processing).

DFHPD710

Entry points

DFHPD710

Called by

MVS IPCS program

Description

DFHPD710 runs as an exit from the MVS IPCS program. It formats an MVS system dump (SDUMP) using the IPCS service routines to extract data and print output, including interpreted trace.

DFHPEP

Entry points

DFHPEPNA

Called by

DFHACP

Description

The program error program is CICS-supplied and establishes a base register, establishes addressability to the COMMAREA passed from DFHACP using a DFHPC CTYPE=LINK_URM macro call, and returns control to DFHACP. DFHPEP can be modified by the user to perform further recovery operations.

DFHPHP

Entry points

DFHPHPNA

Called by

DFHMCP, DFHTOM

Description

The partition handling program has one entry point, and starts with a branch table that passes control to the required routine according to the request.

The main routines of DFHPHP are:

```
PHPPSI - Loads a partition set
PHPPSC - Destroys any existing partitions and
         creates new partitions
PHPPIN - Extracts the AID, cursor position, and
         partition ID
PHPPXE - Activates the appropriate partition if
         data is received from an unexpected
         partition.
```

DFHPL1OI

Description

The PL/I interface module contains the following routines:

DFHPL1N
: Initial entry point for PL/I programs under CICS

DFHPL1I
: CICS macro service interface

DFHPL1C
: Set the CSA address

IBMBOCLA/B/C
: Startup routines for open/close functions.

DFHPRK

Entry points

DFHPRKNA

Called by

DFHZATT

Description

The 3270 print key program (transaction CSPK) is invoked when, under z/OS Communications Server, the 3270 program access key designated as the print key is pressed and no task is attached to the terminal. If the 3270 hardware copy feature is present, DFHPRK attaches task CSCY to the printer designated in the TCTTE, and DFHCPY is executed. If the copy feature is not present, DFHPRK executes a DFHTC TYPE=PRINT macro.

DFHPSP

Entry points

DFHPSPNA

Called by

DFHEPS

Description

DFHPSP is the system spooling interface control module.

DFHPSPDW

Entry points

DFHPSPDW

Called by

DFHSPP

Description

DFHPSPDW is the system spooling interface DWE.

DFHPSPSS

Entry points

DFHPSPSS

Called by

DFHPSP

Description

The system spooling JES interface subtask module attaches a subtask to check whether a writer name and a token have been supplied. It opens and closes JES data sets, reads a record, and writes a record.

DFHPSPST

Entry points

DFHPSPST

Called by

DFHPSPSS

Description

DFHPSPST is the system spooling JES interface control module.

DFHPSSVC

Entry points

DFHPSSNA

Called by

DFHPSPSS, DFHPSPST

Description

DFHPSSVC is the system spooling interface module that retrieves a data set name for a given external writer name, dynamically allocates it, and returns its DDNAME.

DFHPUP

Entry points

DFHPUPNA

Called by

DFHAMP, DFHCSDUP

Description

The parameter utility program transforms the definition data of the CSD. In the CSD, the data is held in a compacted form and each field is self-identifying. Elsewhere in the processing, these fields are handled in parameterized form, using an argument address list. It also serves to transform the resource definition to the original high-level command.

DFHP3270

Entry points

DFHP32NA

Called by

CSPP transaction, DFHTCP, DFHZCP

Description

The 3270 print program prints 3270 data received from a screen on a 3270 printer. The data is compressed where possible and then transmitted to the printer.

DFHQRY

Entry points

DFHQRY

Called by

DFHALP, DFHTCTI, DFHZATT

Description

The query transaction (DFHQRY) sends a READ PARTITION QUERY structured field to a 3270, analyzes the response, and completes information in the corresponding TCTTE. DFHQRY can be attached by DFHALP, DFHTCTI, or DFHZATT.

DFHRCEX

Entry points

DFHRCEX

Called by

DFHFCBP, DFHTCBP, DFHUSBP

Description

DFHRCEX enables the global user exits for emergency restart processing.

DFHRKB

Entry points

DFHRKBNA

Called by

DFHCPY

Description

The release 3270 keyboard program is initiated by DFHCPY to release a 3270 keyboard. It does this by issuing a DFHTC TYPE=WRITE macro that sends a 3270 write control character.

DFHREST

Entry points

DFHREST

Called by

DFHXMTA

Description

The transaction restart program, DFHREST, is a user-replaceable module that helps you to determine whether or not a transaction is restarted. The default DFHREST module requests a transaction restart under certain conditions; for example, for a program isolation deadlock, one of the tasks is backed out and automatically restarted, and the other is allowed to complete its update.

DFHRLRA$, DFHRLR1$

Entry points

DFHRLRNA

Called by

DFHMCP

Description

The route list resolution program builds a terminal type parameter (TTP) control block for each type of terminal for which a message is to be built. A TTP is acquired for each terminal type in the user route list and the direct terminal if there is one.

The main subsections of DFHRLR are:

```
RLRALL   - Routing with ROUTE=ALL specified in
           application
RLRLIST  - Routing with route list specified in
           application
RLROPCL  - Routing with OPCLASS= specified in
           application
RLRRTEBY - Nonrouting, non-LDC device (that is
           direct terminal)
RLR3601  - Nonrouting LDC device.
```

DFHRMSY

Entry points

DFHRMSNA

Called by

DFHERMSP, DFHERMRS

Description

The purpose of task-related user exit resynchronization is to resolve any indoubt LUWs. Task-related user exit resynchronization is called by DFHERMRS during execution of the RESYNC command to restore the CICS end of the thread that was interrupted by the failure of the connection with the resource manager.

It is also called by DFHERMSP when a wait is unshunted and requires RMI resynchronization with a resource manager.

DFHRTC

Entry points

DFHRTCNA

Called by

CSSF transaction

Description

The CSSF transaction is invoked on the remote system when a CRTE routing session is to be canceled. CSSF runs the CRTE cancel command processor, DFHRTC, to sign off the user and terminate the extended routing session. DFHRTC calls DFHSUSN to sign off the surrogate.

DFHRTE

Entry points

DFHRTENA

Called by

transaction CRTE, DFHSNTU

Description

The transaction routing program establishes a transaction routing session with a remote region specified by the user. Subsequent input is analyzed by DFHRTE, the transaction code extracted, and a request issued to DFHZTSP to route the transaction to the required system.

DFHSFP

Entry points

DFHSFP

Called by

CESF trans.

Description

The sign-off program signs off the user who invoked the CESF transaction.

DFHSIA1

Entry points

DFHSIANA

Called by

DFHAPSIP

Description

The DFHSIA1 system initialization program loads and initializes the CSA.

DFHSIB1

Entry points

DFHSIBNA

Called by

DFHAPSIP

Description

The DFHSIB1 system initialization program loads the CICS nucleus.

DFHSIC1

Entry points

DFHSICNA

Called by

DFHAPSIP

Description

The DFHSIC1 system initialization program initializes the transaction manager and the storage manager domain's macro compatibility interface, acquires a TCA for LIFO functions during initialization, initializes user exits, and processes the START parameter.

DFHSID1

Entry points

DFHSIDNA

Called by

DFHAPSIP

Description

The DFHSID1 system initialization program performs the following functions:

- Adds storage subpools for transient data use
- Allocates storage for transient data control blocks:
  - TDST
  - MBCA, MBCBs, and MQCBs, I/O buffers if required
  - MRCA, ACBs, MRCBs, and RPLs
- Sets up the extrapartition transient data queue CXRF.

DFHSIF1

Entry points

DFHSIFNA

Called by

DFHAPSIP

Description

The DFHSIF1 system initialization program initializes terminal control. DFHSIF1:

- Opens the z/OS Communications Server ACB
- Builds hash-table entries for non-z/OS Communications Server terminals
- Constructs a DFHZCP module list in the TCT prefix
- Initializes the attach tables.

DFHSIG1

Entry points

DFHSIGNA

Called by

DFHAPSIP

Description

The DFHSIG1 system initialization program opens the dump data set.

DFHSIH1

Entry points

DFHSIHNA

Called by

DFHAPSIP

Description

The DFHSIH1 system initialization program:

- Loads the DBCTL call processor (DFHDLIDP)
- Loads the remote DBCTL call processor (DFHDLIRP) if necessary
- Attaches the TCP task.

DFHSII1

Entry points

DFHSIINA

Called by

DFHAPSIP

Description

The DFHSII1 system initialization program establishes AP domain recovery routines in DFHSRP and calls DFHICRC to initialise Interval Control services. It attaches the CPLT transaction to run the first stage PLTPI programs, the CSTP transaction (the TCP task) and a system transaction to run the rest of AP initialization (the III task). The rest of DFHSII1, running as the III task:

- Starts XRF control transactions if required
- Attaches the CICS restart tasks to run in parallel:
  - Security interface
  - Transient data
  - Terminal control
  - Program control
  - Task control
  - File control
  - Common programming interface (CPI)
  - Partner resource manager
  - Object recovery
  - Autoinstall terminal model manager
- Waits for the restart tasks to complete
- Processes the GRPLIST parameter

DFHSIJ1

Entry points

DFHSIJNA

Called by

DFHAPSIP

Description

DFHSIJ1 is the last to be executed in the process of system initialization. It issues the message 'CONTROL IS BEING GIVEN TO CICS' and passes control back to DFHAPSIP. DFHSIJ1:

- Links to DFHCRSP, if IRCSTRT=YES is specified as a system initialization parameter, to start up the interregion communication session
- Links to DFHPSIP to enable the system spooling interface
- Enables the DL/I high-level programming interface by acquiring an exit program block and addressing DFHEDP
- Enables AUTOINSTALL
- Links to the second-stage PLT programs listed in DFHPLT, then deletes this table
- Issues a DFHLDLDM SET_OPTIONS call to instruct the loader domain to write all outstanding program definitions to the catalogs.

DFHSIP

Entry points

DFHKESIP

Called by

MVS

Description

DFHSIP initializes CICS and also contains code for the following domains:

- Kernel (KE)
- Domain manager (DM)
- Dispatcher (DS)
- Dump (DU)
- Global catalog (GC)
- Local catalog (LC)
- Loader (LD)
- Lock manager (LM)
- Message (ME)
- Parameter manager (PA)
- Storage manager (SM)
- Trace (TR).

DFHSKP

Entry points

DFHSKMNA, DFHSKC, DFHSKE

Called by

MVS, DFHFCL, DFHFCM, DFHFCN, DFHPSPSS, DFHSTP, DFHXSMX

Description

DFHSKP consists of these modules, which are link-edited together:

```
DFHSKM - subtask manager
DFHSKC - subtask control program
DFHSKE - subtask execution program.
```

DFHSKM calls and, if necessary, attaches DFHSKC to process the created work queue element (WQE). DFHSKM also causes termination of the subtask when requested, and handles DWE processing and task cancel requests. DFHSKC starts an operating system subtask, DFHSKE, and waits for its completion. DFHSKE processes WQEs, looking at in-progress and waiting queues on a first-in, first-out basis. DFHSKE intercepts program checks and operating system abends.

DFHSMSCP

Entry points

DFHSMSCP

Called by

DFHSC macro

Description

The storage control program is called as a result of DFHSC GETMAIN and FREEMAIN macro requests issued from CICS modules.

DFHSNAT

Entry points

DFHSNAT

Called by

DFHCRNP, DFHZISP, DFHZSUP (via DFHSUSN)

Description

The attach-time signon/sign off interface program provides support for the signon and sign off of LU6.2 sessions.

DFHSNNFY

Entry points

DFHSNNFY

Called by

IRRDPR10

Description

The CICS segment notify exit is called by RACF whenever a change is made to a user's CICS segment in the RACF database.

DFHSNP

Entry points

DFHSNP

Called by

CESN transaction

Description

The signon program is called in response to a CESN signon request. DFHSNP interprets the signon parameters, prompts the operator for more parameters if needed, and passes the values to the security manager for verification.

DFHSNSN

Entry points

DFHSNSN

Called by

DFHCSSC, DFHSNAT (via DFHSUSN)

Description

The optimized signon/sign off interface program provides a mechanism for optimizing calls to the security manager. It achieves this optimization using the signon table (SNT).

DFHSNVCL

Entry points

DFHSNVCL

Called by

IRRDPR02

Description

The OPCLASS validation exit is called by RACF to validate the operands of the OPCLASS subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands. DFHSNVCL checks whether the operands are in the range 1 through 24.

DFHSNVID

Entry points

DFHSNVID

Called by

IRRDPR02

Description

The OPIDENT validation exit is called by RACF to validate the operand of the OPIDENT subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands.

DFHSNVPR

Entry points

DFHSNVPR

Called by

IRRDPR02

Description

The OPPRTY validation exit is called by RACF to validate the operand of the OPPRTY subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands. DFHSNVPR checks whether the operand is in the range 0 through 255.

DFHSNVTO

Entry points

DFHSNVTO

Called by

IRRDPR02

Description

The TIMEOUT validation exit is called by RACF to validate the operand of the TIMEOUT subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands. DFHSNVTO checks whether the operand is in the range 1 through 60.

DFHSPP

Entry points

DFHSPPNA

Called by

DFHESP, DFHSP macro

Description

The syncpoint program is invoked during a user-specified syncpoint (by a DFHSP macro) or at task termination. For a rollback request only, DFHSPP calls DFHDBP to restore recoverable resources. It scans the DWE chain invoking the appropriate DWE processors, and performs the necessary syncpoint logging. It dequeues all resources enqueued by the transaction. DFHSPP processes any DWEs connected with the resource manager, and processes the RESYNC command.

The main subroutines of DFHSPP are:

```
SPP00005 - Write DWE log data
SPP02020 - Build a DWE chain that can be logged
SPP03000 - End.
```

DFHSRLI

Entry points

DFHSRLI

Called by

DFHSRP

Description

DFHSRLI is called during recovery processing after a system abend has occurred, to build the SRP_ERROR_DATA block and pass control to the XSRAB global user exit.

DFHSRP

Entry points

DFHSRPNA

Called by

AP domain recovery routines

Description

The system recovery program deals with program check interrupts, system abends, and runaway tasks in the AP domain. For a program check, DFHSRP abends the task with abend code ASRA. For a system abend, DFHSRP searches the SRT for the abend code that has arisen and, if a match is found, calls DFHSRLI to invoke the XSRAB global user exit (if active). Afterward, DFHSRP can either abend CICS or attempt to keep it running with only the faulty task abended (ASRB). For a runaway task, DFHSRP abends the task with abend code AICA.

DFHSSEN

Entry points

DFHSSEN

Called by

MVS subsystem interface

Description

The subsystem end-of-memory routine is invoked by the MVS subsystem interface at all end-of-task (EOT) and end-of-memory (EOM) events when the CICS subsystem has been initialized by module DFHSSIN. It cleans up any subsystem control blocks owned by the terminating CICS region.

DFHSSGC

Entry points

DFHSSGC

Called by

DFHCSVC, DFHSSEN (through the subsystem interface)

Description

The subsystem generic connect routine records the existence of active CICS address spaces. When the first CICS address space becomes active in an MVS image, DFHSSGC enables the subsystem broadcast facility of MVS console management. When the last CICS address space becomes inactive in an MVS image, it disables the broadcast facility.

DFHSSIN

Entry points

DFHSSIN

Called by

MVS master scheduler initialization

Description

The CICS subsystem initialization routine reads subsystem parameters from SYS1.PARMLIB, and creates a subsystem vector table (SSVT) for the CICS subsystem. DFHSSIN loads modules DFHSSEN, DFHSSGC, and DFHSSWT into MVS common storage, and saves their addresses in the SSVT.

DFHSSMGP

Entry points

DFHSSMGP

Called by

DFHSSIN

Description

The subsystem interface message program provides message formatting support for the subsystem interface routines, analogous to DFHMGP within CICS. (Neither DFHMGP nor the message domain can be used in this environment because CICS is not active.)

DFHSSMGT

Entry points

DFHSSMNA

Called by

DFHSSMGP

Description

The subsystem interface message table contains the text of messages that are issued by DFHSSMGP.

DFHSSWT

Entry points

DFHSSWTA

Called by

MVS console support

Description

The subsystem interface WTO router is invoked for all MVS console messages when the console message broadcast facility has been enabled by DFHSSGC. DFHSSWT routes DFH messages to DFHSSWTO, and routes MODIFY command text to DFHSSWTF.

DFHSSWTF

Entry points

DFHSSWTF

Called by

DFHSSWT

Description

This module suppresses sign-on passwords or password phrases that are supplied on CESL or CESN transactions entered through MODIFY commands on an MVS console. Any passwords or password phrases are replaced by eight asterisks.

DFHSSWTO

Entry points

DFHSSWTO

Called by

DFHSSWT

Description

This module inserts the CICS region's applid into all DFH messages issued under a CICS TCB whose applid can be determined.

DFHSTDT

Entry points

DFHSTDT

Called by

DFHAPST

Description

This module is called by DFHAPST to collect or reset dynamic transaction backout statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTFC

Entry points

DFHSTFC

Called by

DFHAPST

Description

This module is called by DFHAPST to collect or reset file control statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTIB

Entry points

DFHSTIB

Called by

DFHAPST

Description

This module and called by DFHAPST to collect or reset IRC batch system connected statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTJC

Entry points

DFHSTJC

Called by

DFHAPST

Description

This module is called by DFHAPST to collect or reset journal control statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTLK

Entry points

DFHSTLK

Called by

DFHAPST

Description

This module is called by DFHAPST to collect or reset ISC/IRC statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTLS

Entry points

DFHSTLS

Called by

DFHAPST

Description

This module is called by DFHAPST to collect or reset LSRPOOL statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTP

Entry points

DFHSTPNA

Called by

DFHEMTP

Description

The main function of the system termination program is to shut down CICS. In sequence, DFHSTP performs the following functions (according to options specified):

1. Collects statistics now if immediate shutdown
2. Shuts down the resource managers
3. Terminates subsystem interface
4. Resumes suspended tasks
5. Executes the programs defined in the first part of DFHPLT
6. Rebuilds AIDs for paging sessions
7. Breaks the ICE and AID chains
8. Quiesces IRC
9. Executes the programs defined in the second part of DFHPLT
10. Closes all open files managed by CICS file control
11. Synchronize with Recovery Manager shutdown keypoint
12. Call WKP to catalog terminals and profiles
13. Terminate extra partition TD
14. Signs off from the CAVM
15. Terminates general-purpose subtasking facility
16. Calls the kernel to terminate the system.

Returns to

MVS

DFHSTSZ

Entry points

DFHSTSZ

Called by

DFHAPST

Description

DFHSTSZ is called by DFHAPST to collect or reset FEPI statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTD

Entry points

DFHSTTD

Called by

DFHAPST

Description

DFHSTTD is called by DFHAPST to collect or reset transient data statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTM

Entry points

DFHSTTM

Called by

DFHAPST

Description

DFHSTTM is called by DFHAPST to collect or reset table manager statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTR

Entry points

DFHSTTR

Called by

DFHAPST

Description

DFHSTTR is called by DFHAPST to collect or reset terminal statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTS

Entry points

DFHSTTS

Called by

DFHAPST

Description

DFHSTTS is called by DFHAPST to collect or reset temporary-storage statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSUSN

Entry points

DFHSUSN

Called by

DFHACP, DFHBSTS, DFHCRNP, DFHCSSC, DFHEEI, DFHEIQST, DFHERM, DFHESN, DFHMGPME, DFHMGP00, DFHRTC, DFHSUSX, DFHTCTI, DFHTPQ, DFHTPR, DFHXSMN, DFHZCUT, DFHZEV1, DFHZEV2, DFHZISP, DFHZIS2, DFHZNAC, DFHZOPN, DFHZSUP, DFHZTSP, DFHZXCU

Description

DFHSUSN is used to create, destroy, and query the contents of a signon table element (SNTTE). It calls DFHSUSX to notify the XRF alternate system of the creation and destruction of SNTTEs. It also provides an interface for the creation and validation of encrypted passwords used in LU6.2 bind password processing.

DFHSUSX

Entry points

DFHSUSX

Called by

DFHTCRPU, DFHZXCU, DFHSUSN

Description

DFHSUSX provides tracking for SNTTEs. This module is responsible for:

- Sending messages to an alternate system to reflect the current state of the SNTTEs in the active system
- Actioning an add or delete of an SNTTE in an alternate system, based on information tracked from another CICS system
- Making changes to the signed-on state in an alternate system, based on information tracked from another CICS system.

Entry points

DFHSUWT

Called by

DFHMEME, DFHSUWT

Description

The DFHSUWT module provides the following support for executing MVS WTO and WTOR SVCs:

- SEND support for Write To Operator (WTO)
- CONVERSE support for Write To Operator With Reply (WTOR).

For further information about DFHSUWT, see "WTO and WTOR" on page 1877.

DFHSUZX

Entry points

DFHSUZX

Called by

DFHBSTZV, DFHEIQSC, DFHEIQST, DFHEIQTR

Description

The ZC trace controller is responsible for actioning set, cancel, and inquire requests for the CICS z/OS Communications Server exit tracing facility. It sets or unsets the control flags and gets or releases the storage used by the DFHZETR function located in the ACB and RPL exits.

DFHTACP

Entry points

DFHTACNA

Called by

DFHTCP

Description

The terminal abnormal condition program is invoked by DFHTCP and performs the following functions:

- Analyzes error codes in the TACLE
- Sends appropriate messages to the CSMT transient data destination (for terminal errors), or to the CSTL transient data destination (for logical errors)
- Invokes the user-supplied (or sample) terminal error program (DFHTEP)
- Takes the appropriate actions resulting from the defaults which may have been modified by the terminal error program.

DFHTAJP

Entry points

DFHTAJNA

Description

The time adjustment program calls DFHICP to reset the CSA's time fields according to the host-supplied time-of-day. DFHTAJP then scans the ICE chain and adjusts the expiry time of interval-controlled ICEs. Time-controlled ICEs are not adjusted but the ICE chain is reordered so that it is left in order by expiry time. Times held in the TCT and CSATCNDT are decreased, and negative times are made zero. Lastly, DFHTAJP writes a message.

DFHTBSB

Entry points

DFHTBSB

Called by

DFHZCQIS

Description

DFHTBSB adds a node to the control-block structure. It is called during the dynamic installation of TCT resources, and calls routines in the control block builder.

DFHTBSBP

Entry points

DFHTBSBP

Called by

DFHTBSB, DFHTBSBP

Description

DFHTBSBP is the recursive part of DFHTBSB.

DFHTBSD

Entry points

DFHTBSD

Called by

DFHZCQDL

Description

DFHTBSD deletes a node in a CICS terminal network.

DFHTBSDP

Entry points

DFHTBSDP

Called by

DFHTBSD, DFHTBSDP

Description

DFHTBSDP is the recursive part of DFHTBSD.

DFHTBSL

Entry points

DFHTBSL

Called by

DFHTBSR, DFHZCQCH

Description

DFHTBSL creates the recovery record for a node during the dynamic installation of a TCT table entry using the CEDA INSTALL command, for example, and calls routines in the control-block builder.

DFHTBSLP

Entry points

DFHTBSLP

Called by

DFHTBSL, DFHTBSLP, DFHTBSSP

Description

DFHTBSLP is the recursive part of DFHTBSL.

DFHTBSQ

Entry points

DFHTBSQ

Called by

DFHZCQIQ

Description

DFHTBSQ is called to retrieve the parameters that were supplied to a TCT table entry at build time.

DFHTBSQP

Entry points

DFHTBSQP

Called by

DFHTBSQ

Description

DFHTBSQP is called by DFHTBSQ to retrieve parameters that were supplied to a TCT table entry at build time.

DFHTBSR

Entry points

DFHTBSR

Called by

DFHZCQRS

Description

DFHTBSR takes a table-builder recovery record and re-creates the corresponding table entry. It is called during warm or emergency restart.

DFHTBSRP

Entry points

DFHTBSRP

Called by

DFHTBSR

Description

DFHTBSRP is called by DFHTBSR.

DFHTBSSP

Entry points

DFHTBSSP

Description

DFHTBSSP performs a commit or rollback action for a previous table-builder change according to the outcome of a logical unit of work. Each action is dequeued from a DWE.

DFHTBS00

Entry points

DFHTBS

Description

DFHTBS00 is the main routine for DFHTBS and holds the addresses of the modules used to build control blocks for the dynamic installation of TCT resources.

DFHTCBP

Entry points

DFHTCBNA

Description

The terminal control backout program restores TCTTEs and other ISC state data during emergency restart.

DFHTCP

Entry points

DFHTCPNA

Description

DFHTCP is the terminal control program. The terminal control task is attached during system initialization and remains until termination. DFHTCP manages all non-z/OS Communications Server terminals, which involves:

- Ensuring that I/O operations are started when possible on the lines
- Analyzing completion information

- Attaching transactions when data is received from a terminal and no task is attached to that terminal
- Servicing terminal control requests from user transactions.

The modules and subsections of DFHTCP are:

DFHTCAM
: Terminal control TCAM device dependent

DFHTCCLC
: Terminal control line control scan routine

DFHTCCOM
: Terminal control common logic

DFHTCCSS
: Terminal control start-stop common logic

DFHTCDEF
: Terminal control symbol definition

DFHTCORS
: Terminal control storage handling

DFHTCSAM
: Terminal control sequential terminal device dependent

DFHTCTI
: Terminal control task initiation

DFHTCTRN
: Terminal control translate tables

DFHTCRP

Entry points

DFHTCRP

Description

DFHTCRP initializes and recovers terminal control definitions and protected messages. It is run as a task during CICS initialization.

DFHTCRPC

Entry points

DFHTCRPC

Called by

DFHZXQO

Description

DFHTCRPC is the XRF tracking interface for TCT contents. It is one of a set of routines called by DFHZXQO from the same CALL statement, the entry point address having been passed to DFHZXQO. This routine calls ZC RESTORE to add or delete a TCT entry based on information from another CICS system using the log, the catalog, or the XRF tracking queues.

DFHTCRPL

Entry points

DFHTCRPL

Called by

DFHTCRP

Description

DFHTCRPL installs TCT resources defined by the TCT macros.

DFHTCRPS

Entry points

DFHTCRPS

Called by

DFHZXQO

Description

DFHTCRPS is the XRF tracking interface for ZCP sessions. It is one of a set of routines called by DFHZXQO from the same CALL statement, the entry point address having been passed to DFHZXQO. This routine calls DFHZXST (through DFHZXS) to make changes to the session state.

DFHTCRPU

Entry points

DFHTCRPU

Called by

DFHZXQO

Description

DFHTCRPU is the XRF tracking interface for signon table elements (SNTTEs). It is one of a set of routines called by DFHZXQO from the same CALL statement, the entry point address having been passed to DFHZXQO. This routine calls DFHSUSX to add or delete tracked SNTTEs, and to make changes to the signed-on state.

DFHTDA

Entry points

DFHTDANA

Called by

DFHAMCSD, DFHAPTD, DFHCRNP, DFHCRQ, DFHDBP, DFHEIQMS, DFHEIQSQ, DFHESE, DFHETD, DFHJCP, DFHMCP, DFHMGP00, DFHRCRP, DFHRUP, DFHSII1, DFHSTP, DFHSTTD, DFHTCAP, DFHTDRP, DFHTEPM, DFHTPQ, DFHTRP, DFHTSRP, DFHWKP, DFHZNAC

Description

DFHTDA, which is link-edited with RMODE(ANY), handles DFHTD macro requests. In particular:

- DFHTD TYPE=GET|PUT|PURGE requests are converted to the corresponding DFHTD CTYPE=GET|PUT|PURGE requests.
- DFHTD CTYPE=GET|PUT|PURGE requests for intrapartition queues are routed to DFHTDQ for further processing.
- All of the processing for DFHTD CTYPE=GET|PUT for extrapartition queues is done under the QR TCB.
- Much of the processing for DFHTD CTYPE=OPEN|CLOSE for extrapartition queues is done under the RO TCB.

CICS Transaction Server for z/OS uses QSAM GL|PL mode processing.

DFHTDB

Entry points

DFHTDBNA

Called by

DFHTDA

Description

DFHTDB, which is link-edited with RMODE(ANY), handles DFHTD macro requests for intrapartition queues. In particular, DFHTDB:

- Manages the input and output cursors for each queue
- Manages space on the intrapartition data set
- Initiates transactions when trigger levels are reached
- Manages the buffers; processing is done under the QR TCB
- Manages the strings; processing is done under the CO TCB.

DFHTDEXL

Entry points

EX11RTNE

Called by

QSAM

Description

DFHTDEXL contains the DCB abend exit routine used for extrapartition processing.

DFHTDP

Entry points

DFHTDANA

Called by

DFHAMCSD, DFHAPTD, DFHCRNP, DFHCRQ, DFHDBP, DFHEIQMS, DFHEIQSQ, DFHESE, DFHETD, DFHMCP, DFHMGP00, DFHRCRP, DFHRUP, DFHSII1, DFHSTP, DFHSTTD, DFHTACP, DFHTDRP, DFHTEPM, DFHTPQ, DFHTRP, DFHTSRP, DFHWKP, DFHZNAC

Description

DFHTDP is a load module link-edited from object modules for DFHTDA, DFHTDEXL, and DFHTDX.

DFHTDQ

Entry points

DFHTDBNA

Called by

DFHTDA

Description

DFHTDQ is a load module link-edited from object modules for DFHTDB.

DFHTDRM

Entry points

DFHTDRM

Called by

DFHDBP

Description

DFHTDRM is the transient data recovery manager processor. If transient data has any outstanding resources, DFHTDRM is called at phase 1 syncpoint (or backout). For phase 1 syncpoint (or backout) requests, DFHTDRM issues a request to mainline transient data(DFHTDA) to reset any resources that have not yet been released.

DFHTDRP

Entry points

DFHTDRNA

Called by

DFHTDX

Description

DFHTDRP handles transient data recovery during CICS initialization. In particular, DFHTDRP:

- Restores input and output cursors for intrapartition queues on warm start; the cursors are recovered by DFHRUP on emergency restart
- Restores the CI state map on warm start
- Opens extrapartition queues
- Opens the intrapartition data set
- Recovers the CI state map on emergency restart

DFHTDTM

Entry points

DFHTDTM

Called by

DFHALP, DFHEIQMS, DFHEIQSQ, DFHESE, DFHSZRPM, DFHTDRP

Description

DFHTDTM manages the CICS internal record of transient data queue resource definitions, known as the DCT (destination control table). DFHTDTM adds, updates, and deletes entries in this table when transient data queue resource definitions are installed, changed, and discarded, and records images of each entry on the global catalog for use during a warm start or emergency restart.

DFHTDX

Entry points

DFHTDXNA

Called by

Task initiation

Description

DFHTDX is the initial program invoked by the transient data recovery task. It links to program DFHTDRP.

DFHTEP

Entry points

DFHTEPNA

Called by

DFHTACP

Description

The terminal error program is invoked by DFHTACP using a DFHPC CTYPE=LINK_URM macro. The sample DFHTEP (invoked only if there is no customer-supplied version) puts a terminal out of service if the number of terminal errors detected by DFHTACP exceeds default values contained in DFHTEP tables.

DFHTMP

Entry points

DFHTMPNA

Called by

DFHTM macro

Description

The table management program performs locates, adds, deletes, locks, and unlocks to entries in certain CICS tables. DFHTMP uses a hash table for these operations.

The main subroutines of DFHTMP are:

```
CHKTTC     - Check table type code
COMMIT     - Commit table changes
CRTCLE     - Create a change list element
CRTDWE     - Create deferred work element
DELDWE     - Cancel deferred work element
DEQALLDE   - Dequeue on directory element
DEQUEUE    - Dequeue on table modification
DYNHASH    - Dynamic re-hash
ENQDEQDE   - Enqueue/dequeue on directory element
ENQUEUE    - Enqueue on table modification
GET_STORAGE - Get storage from the CICS shared subpool
GET_TASK_STORAGE - Get task lifetime 31-bit storage
GET_TASK_STORAGE_COND - Get task lifetime 31-bit storage
                        (conditionally)
GET_STORAGE_FAILURE - Get storage failure routine
FREE_STORAGE - Release storage from the CICS shared subpool
FREE_TASK_STORAGE - Release task lifetime 31-bit storage
LOCATE_PREVIOUS_DE - Locate previous directory
             element in collating series
LOCATETE   - Locate a table/directory entry
LOCFDIRE   - Locate a free directory element
NOTERL     - Note Read Lock
SETABORD   - Set up alphabetic ordering pointer
             for a given table type
TMFINDLOCK - Find a read lock
TMPDWEEP   - Deferred work element processor
```

```
TMSETLOCK  - Set a read lock
TMUNLOCK   - Release a read lock
UNQUIES    - Unquiesce a directory element.
```

DFHTON

Entry points

DFHTONNA

Called by

DFHDBP, DFHSPP

Description

The terminal object resolution module is called by DFHDBP or DFHSPP during DWE processing for DFHTOR. It calls DFHTOR with end-LUW-cancel or end-LUW-commit code to perform cancel or commit of changes to TERMINAL, TYPETERM, CONNECTION, or SESSIONS definitions.

DFHTOR

Entry points

DFHTORNA

Called by

DFHAMP, DFHTON

Description

DFHTOR is the terminal object resolution program. DFHAMP calls DFHTOR for a TERMINAL, TYPETERM, CONNECTION, or SESSIONS object in a CICS system definition (CSD) file that is being installed, or when DFHAMP encounters an end-of-group. DFHTOR processes the objects and passes them to the terminal control builder program (DFHZCQ). The DFHTON entry is used for DWE processing.

DFHTORP

Entry points

DFHTORNA

Called by

DFHSII1

Description

DFHTORP is the terminal object recovery program. It is called during CICS initialization to purge TYPETERM and model terminal definitions from the catalog on a cold start, and to recover these definitions on an emergency restart.

DFHTPPA$, DFHTPP1$

Entry points

DFHTPPNA

Called by

DFHDSB, DFHM32

Description

The terminal page processor program handles DFHBMS TYPE=OUT, STORE, and RETURN requests. If OUT, DFHTPP sends the complete page using DFHTC macro requests; if STORE, the page is sent to temporary storage; and if RETURN, no output operation takes place but the page is returned to the application program.

The main subroutines of DFHTPP are:

```
TPNODDS - TYPE=STORE (PAGING) requests
TPOUT   - TYPE=OUT (TERMINAL) requests (the macro
          DFHTOM is used by both DFHTPP and DFHTPR
          to handle output to terminals)
TPRETPG - TYPE=RETURN (SET) requests.
```

Returns to

DFHPBP

DFHTPQ

Entry points

DFHTPQNA

Called by

DFHICP, DFHMCP, DFHTCP

Description

The undelivered messages cleanup program is initiated periodically in order to cancel the delivery of BMS messages that have been placed in temporary storage, but have remained undelivered for an interval exceeding the purge delay time interval specified by the PRGDLAY system initialization parameter, if this has a nonzero value.

DFHTPR

Entry points

DFHTPRNA

Called by

DFHMCP, DFHTCP

Description

The terminal page retrieval program (transaction CSPG) is invoked:

- By automatic transaction initiation as a result of a SCHEDULE issued by DFHTPS
- By a DFHPGLK LINK from DFHMCP, when CTRL=RETAIN or RELEASE on DFHBMS TYPE=PAGEOUT (RETAIN or RELEASE on SEND PAGE at command level)
- When CSPG or an operator paging command is entered at a terminal.

If the message is autopaged, DFHTPR retrieves the pages of the message in order, transmits them to the terminal, and then purges the message. Otherwise DFHTPR runs pseudo-conversationally. All further input is passed to DFHTPR, until the message is purged explicitly or implicitly. If the input is a valid paging command (page retrieval, page copy, page purge, or page chaining), it is processed. It is rejected if explicit purge is required, or passed back to normal task initiation if automatic purge is allowed.

The main subsections of DFHTPR are:

```
DFHMSPUT - Send error message to terminal
TPENCCHN - Encode and execute page chain
TPENCCOP - Encode and execute page copy
TPENCPUR - Execute page purge
TPENCRET - Encode page retrieval
TPERETA  - Reset to autopaging
TPERETQ  - Page query
TPEXIT   - Exit from program
TPEXPUR  - Execute page purge
TPEXRET  - Execute page retrieval
TPTSGET  - Get MCR or page from temporary storage.
```

DFHTPS

Entry points

DFHTPSNA

Called by

DFHICP, DFHMCP

Description

The terminal page scheduling program (transaction CSPS) is invoked for each terminal type to which a BMS logical message built with TYPE=STORE is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status. DFHTPS contains the following major subsections, each dealing with a separate function:

- DFHTPSNA—used when DFHTPS is invoked by automatic initiation on expiry of ICE, and as a result of an IC PUT request issued by DFHMCP (there is no associated terminal). This invocation schedules CSPG for terminals on this system, and schedules CSPS on the link to each remote system which owns terminals contained in the route list for the message (that is the function of TPS02000).

- TPS01000—used when DFHTPS is linked to from DFHMCP for direct paging requests to a terminal on a remote system. The task has a surrogate TCTTE as its primary facility, and owns a relay link connected to the terminal owning system. This section ships the pages of the message to the terminal-owning region, where it is re-created by the relay program (DFHAPRT) which issues BMS, STORE, TEXT, NOEDIT, and PAGEOUT requests.
- TPS02000—used when DFHTPS is scheduled by TPS01000 to run against the link to a remote system. This routine ships the logical message to the remote system and deletes the terminals on the remote system from the terminal list in the original message control record. (TPS03000 receives the information at the remote system.)
- TPS03000—used when DFHTPS is invoked by an ATTACH request from a remote system (that is, originated by TPS01000 or TPS02000). This routine receives the shipped logical message and issues BMS ROUTE, TEXTBLD, NOEDIT, and PAGEOUT requests to re-create the logical message on the terminal-owning region.

DFHTPS contains the following subroutine:

- TPSSHIPM—ships a complete logical message.

DFHTRAP

Entry points

DFHTRANA

Called by

DFHTRPT

Description

The FE global trap/trace exit is provided for diagnostic use only under the guidance of service personnel.

DFHTR710 and AMDUSREF

Entry points

DFHTRPRG

Called by

IPCS

Description

The CICS GTF trace formatting routine is invoked by IPCS processing of the GTFTRACE keyword when a CICS entry (USR F6C, format ID X'EF') is encountered. For each entry, it writes a line containing the job name and then formats the entry in the same form as DFHTU710 does for an auxiliary trace print. AMDUSREF is defined as an alias for DFHTR710 because IPCS looks for a program called AMDUSRxx to format entries with format ID xx.

DFHTRP

Entry points

DFHTRPNA

Called by

Many AP domain modules

Description

The trace control program translates DFHTR, DFHTRACE, and DFHLFM macro requests to write trace entries into TR domain TRACE_PUT requests. DFHTRP collects the data required in the trace for the specified trace ID into a standard layout and issues the TRACE_PUT call. For requests to change the various trace flags that control tracing, DFHTRP issues KEDD format calls to the kernel domain.

DFHTRZCP

Entry points

DFHTRZCP

Called by

CEDA transaction, DFHTCRP, DFHTOR

Description

DFHTRZCP builds a terminal builder parameter set.

DFHTRZIP

Entry points

DFHTRZIP

Called by

CEDA transaction, DFHTCRP, DFHTOR

Description

DFHTRZIP builds a chain of builder parameter sets for sessions.

DFHTRZPP

Entry points

DFHTRZPP

Called by

CEDA transaction, DFHTCRP, DFHTOR

Description

DFHTRZPP builds a pool builder parameter set.

DFHTRZXP

Entry points

DFHTRZXP

Called by

CEDA transaction, DFHTCRP, DFHTOR

Description

DFHTRZXP builds a connection builder parameter set.

DFHTRZYP

Entry points

DFHTRZYP

Called by

CEDA transaction, DFHTCRP, DFHTOR

Description

DFHTRZYP builds a TYPETERM builder parameter set.

DFHTRZZP

Entry points

DFHTRZZP

Called by

CEDA transaction, DFHTCRP, DFHTOR

Description

DFHTRZZP merges a TYPETERM builder parameter set into a terminal builder parameter set.

DFHTSP

Entry points

DFHTSPNA

Called by

DFHACP, DFHALP, DFHCRQ, DFHDBP, DFHDIP, DFHEDFP, DFHESE, DFHETS, DFHICP, DFHMCP, DFHMSP, DFHRTE, DFHSII1, DFHSTP, DFHTCBP, DFHTPP, DFHTPQ, DFHTPR, DFHTPS, DFHTSBP, DFHTSP, DFHTSRP, DFHZISP, DFHZRAQ, DFHZRAR, DFHZRSP

Description

The temporary-storage program services DFHTS requests. It maintains the tables, directories, and maps necessary to keep track of every temporary-storage record and of available space on the VSAM auxiliary storage or in main storage. The main subroutine of DFHTSP is DFHTSPAM, which manages auxiliary storage (including multiple buffers and strings).

DFHTU710

Entry points

DFHTRPRA

Called by

MVS

Description

The trace utility program formats and prints trace records stored on the auxiliary trace data set. This utility program is run as a separate job, and extracts selected trace entries as specified on parameter statements supplied as part of the input to the program.

DFHUCNV

Entry points

DFHUCNV

Called by

DFHCCNV

Description

DFHUCNV is a sample program for CICS OS/2 user data conversion. CICS OS/2 is no longer supported.

DFHUEH

Entry points

DFHUEHNA

Called by

CICS management modules containing exit points

Description

The user exit handler is the link between an exit point in a CICS management module in the AP domain, and the user code. DFHUEH invokes in turn each started exit program for that exit point, passing a parameter list defined in the CICS management module.

DFHUEM

Entry points

DFHUEMNA

Called by

DFHEIP, DFHEIG

Description

The EXEC interface processor for the ENABLE, DISABLE, and EXTRACT user exit commands.

DFHUSBP

Entry points

DFHUSBNA

Called by

DFHRCRP

Description

The user backout program sends records, journaled by the user to the system log, to a user exit during emergency restart. The records are extracted by DFHRUP from the restart data set. They may exist for any logical unit of work, whether in flight or not, depending on the JCRSTRID value specified when the record was written.

DFHWCCS

Entry points

DFHWCCS

Called by

Many CAVM modules

Description

DFHWCCS provides common services for the CAVM:

- MVS FREEMAIN
- MVS GETMAIN

- MVS POST
- Message or MVS ABEND
- Create CAVM process block.

Returns to

MVS abend, caller

DFHWCGNT

Entry points

DFHWCGNA

Description

DFHWCGNT is the entry point list for CAVM modules above the 16 MB line.

DFHWDATT

Entry points

DFHWDATT

Called by

DFHWDINA, DFHWMG1, DFHWMP1, DFHWSXPI

Description

DFHWDATT creates the CAVM process.

DFHWDINA

Entry points

DFHWDINA

Called by

DFHWSRTR

Description

DFHWDINA attaches the initial CAVM process. It sets up lock tables, the dispatcher control area, the LIFO control area, and the dispatcher ESPIE and ESTAE exits.

Returns to

DFHWDISP

DFHWDISP

Entry points

DFHWDISP, DFHWDIND

Called by

DFHWDWAT, DFHWDINA

Description

DFHWDISP is the CAVM process dispatcher. It dispatches the next ready CAVM process, or waits for an external event. It dispatches the initial CAVM process.

Returns to

Dispatched process, caller of DFHWDINA

DFHWDSRP

Entry points

DFHWDSRP

Called by

DFHWDINA, CAVM program check/abend

Description

DFHWDSRP establishes the ESPIE/ESTAE CAVM process. It performs CAVM process error handling for processes with ESPIE or ESTAE routines.

DFHWDWAT

Entry points

DFHWDWAT

Called by

Many CAVM modules

Description

DFHWDWAT causes the current CAVM process to wait for specific events.

Returns to

DFHWDISP

DFHWKP

Entry points

DFHWKPNA

Called by

DFHSTP

Description

DFHWKP takes a warm keypoint at the normal termination of CICS. This program is part of the restart component.

DFHWLFRE

Entry points

DFHWLFRE

Called by

Many CAVM modules

Description

DFHWLFRE frees the LIFO stack entry for CAVM modules running above the 16 MB line.

DFHWLGET

Entry points

DFHWLGET

Called by

Many CAVM modules

Description

DFHWLGET gets the LIFO stack entry for CAVM modules running above the 16 MB line.

DFHWMG1

Entry points

DFHWMG1

Called by

DFHWMI, DFHWDISP, DFHWDSRP

Description

DFHWMG1 is the main module of the CAVM message manager GET MESSAGE service. It is called by DFHWMI to initialize service, and attach itself as a message-reader CAVM process; by DFHWDISP to run as a message-reader CAVM process that reads messages and stores them; and by DFHWDSRP to handle ESPIE/ESTAE exits for the message reader.

DFHWMI

Entry points

DFHWMI

Called by

DFHWSXPI

Description

DFHWMI allocates the CAVM message-manager communication area. It calls each of the main message-manager modules, which then initialize themselves.

DFHWMMT

Entry points

DFHWMMT

Called by

DFHWMRD, DFHWMWR

Description

DFHWMMT provides VSAM GET and PUT services for the CAVM message data set.

DFHWMPG

Entry points

DFHWMPG

Called by

DFHWMP1, DFHWMWR

Description

DFHWMPG copies message data into the buffer provided by the user of PUTMSG, PUTREQ, PUTRSP, and CAVM message-manager services. It provides an ESPIE routine to handle program checks occurring during the copying.

DFHWMP1

Entry points

DFHWMP1

Called by

DFHWMI, DFHWDISP, DFHWDSRP

Description

DFHWMP1 is the main module of the CAVM message-manager PUT MESSAGE service. It is called by DFHWMI to initialize service, and attach itself as a message-writer CAVM process; by DFHWDISP to run as a message-writer CAVM process that writes messages to the CAVM message data set; and by DFHWDSRP to handle ESPIE and ESTAE exits for the message writer.

DFHWMQG

Entry points

DFHWMQG

Called by

DFHWMS20

Description

DFHWMQG runs under the CICS TCB above the 16MB line. It processes GETMSG CAVM message-manager requests. It waits for a message to arrive, then copies from the main-memory message queue created by the CAVM message-reader process.

DFHWMQH

Entry points

DFHWMQH

Called by

DFHWMG1, DFHWMQG

Description

The CAVM message-manager message input queue handler locates or creates message-queue anchor blocks, and adds copies of messages read by the CAVM reader process to the main-memory message queues.

DFHWMQP

Entry points

DFHWMQP

Called by

DFHWMS20

Description

DFHWMQP runs under the CICS TCB above the 16MB line. It processes CAVM message-manager PUTMSG, PUTREQ, and PUTRSP requests; places the request in the appropriate queue; and posts the queue to awaken CAVM process to handle request, waits for completion, and returns response to the caller.

DFHWMQS

Entry points

DFHWMQS

Called by

DFHWMP1, DFHWMWR

Description

The CAVM message-manager message output queue handler provides services to select the next work item to process, and posts items complete.

DFHWMRD

Entry points

DFHWMRD

Called by

DFHWMG1

Description

The CAVM message-manager message read routine reads messages from the CAVM message data set, taking account of the position of the active write cursor, and creates message blocks for copies of messages that have been read.

DFHWMS

Entry points

DFHWMSNA

Called by

Users of CAVM message services

Description

The CAVM message-manager service interface routine runs under the CICS TCB above the 16MB line. It builds a dummy CAVM process block, so that subsequent

modules can run in an XRF LIFO environment, and calls DFHWMS20 to process a request passed by the caller.

DFHWMS20

Entry points

DFHWMS20

Description

The CAVM message manager services interface selects the request type and passes requests to DFHWMQP (PUTMSG, PUTREQ, PUTRSP) or DFHWMQG (GETMSG).

DFHWMWR

Entry points

DFHWMWR

Called by

DFHWMP1

Description

The CAVM message-manager message write routine takes data from PUTMSG requests and copies them into CI buffers to be written to the CAVM message data sets.

DFHWOS

Entry points

DFHWOSNA

Description

The overseer startup module loads DFHWOSA and passes control to it.

DFHWOSA

Entry points

DFHWOSNA

Called by

DFHWOS

Description

The overseer services initialization module processes control parameters, loads DFHWOSB, and sets up entry points for overseer services.

DFHWOSB

Entry points

DFHWOSNA

Called by

Overseer program

Description

The overseer service module processes requests from the overseer program which are issued by the DFHWOSM macro.

DFHWSRTR

Entry points

DFHWSMNA

Called by

DFHXRA, MVS after attach of new TCB

Description

The CAVM state-management request router and subtask entry point is the initial entry point for a CAVM task attached by DFHWSSN1 to process the CAVM SIGNON command. It calls DFHWSSN2 to continue the processing of the SIGNON request and, if it is accepted, calls DFHWDINA to attach the tick generator module DFHWSTI as the first and highest-priority CAVM process. It is called under the CICS TCB to queue the CAVM TAKEOVER command for processing by the CAVM task, and to initiate processing of the CAVM SIGNOFF command by detaching the CAVM task. DFHWSRTR is the initial entry point for MVS subtasks attached by the CAVM task to perform various functions, such as issuing requests for CSVC services, or formatting new CAVM data sets when they are used for the first time.

DFHWSSN1

Entry points

DFHWSSNA

Called by

DFHXRA

Description

DFHWSSN1 is the CAVM state management SIGNON initial entry point. The CICS task issues an MVS LINK, specifying load module DFHWSSON to perform a CAVM SIGNON request. DFHWSSN1 attaches the CAVM task to execute the request, waits to see if it is successful, detaches the task and, if it is not successful, reports the result to CICS.

DFHWSSN2

Entry points

DFHWSSN2

Called by

DFHWSRTR

Description

The CAVM state management SIGNON request handler is entered under the CAVM TCB to process a CAVM SIGNON request. It allocates storage for, and initializes, key CAVM control blocks, sets up DFHWSSOF as an ESTAE exit, calls DFHWSSN3 to OPEN the CAVM data sets, reads the state management record from the control data set, uses the JES inquire-job-status CSVC service provided by DFHWTI, and looks for surveillance signals from other CAVM users to check whether the environment is such that the requested SIGNON can be accepted. It prompts the operator for job status information if necessary. If SIGNON is accepted, it updates the state management record and status CIs to record that this job has signed on to the CAVM. When possible, it also cleans up out-of-date information in the CAVM data sets left behind by jobs that were unable to sign off properly before terminating.

DFHWSSN3

Entry points

DFHWSSN3

Called by

DFHWSSN2

Description

The CAVM state management data set initialization routine builds ACBs, and opens and validates the CAVM control and message data sets for CAVM SIGNON. It builds the reserve parameter list for serializing accesses to the control data set. If new CAVM data sets are being used for the first time, it attaches an MVS subtask to record relevant information in each data set's control interval, and to format the CIs needed by state management.

DFHWSSOF

Entry points

DFHWSSOF

Called by

MVS recovery/termination manager

Description

DFHWSSOF is the CAVM state management SIGNOFF request handler. During SIGNON processing, this module is established as an ESTAE exit for the CAVM task. It purges outstanding I/O requests, reads the state management record from the control data set, and searches it to see if this job has signed on to the CAVM. If so, it updates the status CI and state management record to indicate that the job has signed off. It makes the TAKEOVER message available to DFHWSRTR when an active system signs off after takeover has started.

DFHWSSR

Entry points

DFHWSSR

Called by

DFHWDISP

Description

The CAVM surveillance status reader runs as a process controlled by the XRF dispatcher, DFHWDISP. It reads the status CI of the partner system from the control data set or the message data set, generates internal CAVM events, and drives the NOTIFY exit when the partner's status changes, or its surveillance signals cease. For an alternate system, it monitors and records the time-of-day clock difference when the active system is running in a different MVS image.

DFHWSSW

Entry points

DFHWSSW

Called by

DFHWDISP

Description

The CAVM surveillance status writer runs as a CAVM process controlled by the CAVM dispatcher, DFHWDISP. It writes a system's current status to its status CI in the control data set, or the message data set, to make it available to its partner and to provide a surveillance signal; generates an internal CAVM event when a status write completes; and puts the current time-of-day clock reading in the status CI to permit DFHWSSR to deduce the time-of-day clock difference when the active system and the alternate system are running in different CECs.

DFHWSTI

Entry points

DFHWSTI

Called by

DFHWDISP

Description

The CAVM surveillance tick generator and CICS status monitor runs as a CAVM process controlled by the CAVM dispatcher DFHWDISP. It issues an MVS STIMER for the surveillance interval and, when this expires, generates an internal CAVM clock-tick event, calls the inquire-CICS-status exit, and schedules the surveillance status writer processes, to cause a surveillance signal reporting this system's current status to be written to the control data set or the message data set.

DFHWSTKV

Entry points

DFHWSTKV

Called by

DFHWDISP

Description

The CAVM state management TAKEOVER request handler runs as a CAVM process controlled by the CAVM dispatcher DFHWDISP. When a new active SIGNON has been detected, it reads the state management record from the control data set and attaches an MVS subtask to invoke DFHWTI's validate-CLT CSVC service. When a TAKEOVER command has been issued, it reads the state management record, validates the TAKEOVER request, and attaches an MVS subtask to use DFHWTI's JES inquire-job-status service to determine the current state of the active system.

If the active system is still signed on to CAVM, it updates the state management record to indicate that a takeover is in progress, places the TAKEOVER message for the active system in the alternate system's status, and attaches an MVS subtask to invoke DFHWTI's TAKEOVER-initiate service.

After the active system has signed off (or terminated), it requests DFHWSSR to read the active system's final status, quiesces surveillance processing, and updates the state management record and status CIs to indicate the stage reached by takeover. It then arranges for surveillance processing to be resumed in active mode. It attaches an MVS subtask to invoke DFHWTI's process-CLT CSVC service if necessary.

When the active system has finally terminated, it updates the state management record to take its place as the new active system, generates internal CAVM events, and calls the NOTIFY exit to report the progress of the TAKEOVER request, including acceptability of the time-of-day clock reading. It terminates by returning to DFHWDISP.

DFHWSXPI

Entry points

DFHWSXPI

Called by

DFHWSTI

Description

The CAVM state management CAVM process initialization runs under the tick generator CAVM process towards the end of SIGNON. It attaches the TAKEOVER CAVM process (alternate systems only), two status writer CAVM processes, and two status reader CAVM processes, and then calls the CAVM message management initialization module.

DFHWTI

Entry points

DFHWTINA

Called by

DFHCSVC from: DFHWSSN2, DFHWSTKV, DFHZXSTS

Description

Takeover initiation is the primary function of this module, and is requested by CAVM state management at takeover to terminate the CICS active system issue commands in the CLT, and wait until the CICS active system terminates. Other XRF services provided by this module are to determine whether a job is running, to issue the operator commands for the overseer program, to issue MODIFY USERVAR to z/OS Communications Server, to validate the CLT, and to process the CLT.

DFHWTRP

Entry points

DFHWTRP

Called by

Many CAVM modules

Description

DFHWTRP makes a trace entry in the CAVM main-memory trace table.

DFHXCP

Entry points

DFHXCPNA

Called by

DFHKCP

Description

DFHXCP processes DFHKC CANCEL, CHAP, RESUME, SUSPEND, and WAIT macro calls to the transaction manager.

DFHXCPC

Entry points

DFHXCPC

Called by

DFHKCP

Description

DFHXCPC processes DFHKC ATTACH, CHANGE, DEQ, DEQALL, ENQ, and SRB macro calls to the transaction manager. It receives DFHKC INITIALIZE, REPLACE, and WAITINIT macro calls to the transaction manager and passes them on to DFHKCQ.

DFHXCP1

Entry points

DFHXCP1

Called by

DFHXCPC

Description

DFHXCP1 finds a new range of free transaction numbers when the current range has been used up.

DFHXFP

Entry points

DFHXFPNA

Called by

DFHISP, DFHMIRS

Description

The online data transformation program takes data addressed from a parameter list (command-level or DL/I), and constructs an FMH suitable for transmission to a remote ISC or MRO system; DFHXFP also performs the reverse transformation.

DFHXFQ

Entry points

DFHXFQNA

Called by

DFHXEPRH

Description

The batch data transformation program executes in an EXCI region. DFHXFQ takes data addressed from a DPL parameter list and constructs an FMH suitable for passing to the online region; DFHXFQ also performs the reverse transformation.

DFHXFX

Entry points

DFHXFXNA

Called by

DFHISP, DFHMIRS

Description

DFHXFX performs the same logical transformations of function shipping requests as DFHXFP but in a manner that is optimized for the MRO environment. It is not used for the transformation of DL/I requests; these are processed by DFHXFP.

DFHXRA

Entry points

DFHXRANA

Called by

DFHAPDM, DFHCSSC, DFHCXCU, DFHDBCR, DFHDBCT, DFHSIC1, DFHSII1, DFHSTP, DFHTCRP, DFHTDRP, DFHXRCP, DFHXRSP, DFHZNAC, DFHZOPN, DFHZSLS

Description

DFHXRA is the program that executes the DFHXR macro. It runs under the CICS TCB in AMODE(24). In general, it uses CICS macros to invoke other services. Exceptions are MVS LINK to DFHWSSON to sign on to the CAVM, and MVS LOAD and DELETE for DFHWSMS to sign off from the CAVM, and to initiate

takeover. It invokes global user exit XXRSTAT, which can lead to the abend 208.

DFHXRB

Entry points

DFHXRBNA

Called by

DFHWDSRP, DFHWMQH, DFHWMRD, DFHWSSR, DFHWSTKV

Description

DFHXRB is the XRF notify exit program. Its address is passed to the CAVM when CICS signs on to the CAVM. It runs under the CAVM TCB in AMODE(31); reacts to events detected by various CAVM modules; and creates a queue of work elements (chained from XRWECHN) to be processed by DFHXRSP.

DFHXRC

Entry points

DFHXRCNA

Called by

DFHWSSN2, DFHWSTI

Description

DFHXRC is the CICS-status exit program. Its address is passed to the CAVM when CICS signs on to the CAVM. It runs under the CAVM TCB in AMODE(31), and returns the latest CICS-status data to be written to the state management data set.

DFHXRCP

Entry points

DFHXRCNA

Description

The XRF console communication task runs under the CICS TCB in AMODE(24). It processes MODIFY commands received by CICS during initialization of the alternate system. It initiates takeover, shuts down the active system, and manages trace and dump as required.

DFHXRE

Entry points

DFHXRENA

Called by

DFHPCP

Description

The XRF startup program is the entry point for the system task attached by DFHXRA. It links to DFHXRE, whichever module was indicated by DFHXRA.

DFHXRP

Entry points

DFHXRANA

Called by

Not applicable

Description

DFHXRP consists of six object modules link-edited together:

```
DFHXRA - XRF request processor
DFHXRB - XRF NOTIFY exit program
DFHXRC - XRF inquire status exit program
DFHXRE - XRF startup program
DFHXRF - XRF CAVM sign-off interface
DFHWMS - CAVM message manager service interface.
```

It is loaded by DFHSIB1.

DFHXRSP

Entry points

DFHXRSNA

Called by

DFHXRA

Description

DFHXRSP is the XRF surveillance program, which runs as a program under a CICS transaction. It runs under the CICS TCB in AMODE(31); processes the queue of work elements created by DFHXRB; attaches the catch-up transaction CXCU, initiates takeover, and shuts down CICS as required; and can issue abends 206 and 207.

DFHXSMN

Entry points

DFHXSMNA

Called by

DFHBSTS, DFHCRNP, DFHDLIDP, DFHDLIRP, DFHEDFP, DFHEIPSE, DFHSII1, DFHSUSN, DFHSUXS, DFHTACP, DFHZSUP

Description

The security manager is invoked by the DFHSEC macro, and provides an interface to the external security manager (ESM). DFHXSMN validates the parameters passed, then calls DFHXSMX as a general-purpose subroutine to invoke the ESM.

DFHXSMX

Entry points

DFHXSMNA

Called by

DFHXSMN

Description

DFHXSMX is the subroutine used by the security manager to invoke the external security manager (ESM). For resource checking, this routine first issues the MVS RACROUTE REQUEST=FASTAUTH macro, which calls the ESM in problem state. All other security functions require the caller to be in supervisor state. For these functions, and for a failed FASTAUTH call that requires logging, the CICS SVC is issued under a general purpose subtask, entered by the DFHSK macro, to shield the main CICS task from any imbedded waits that may occur in the ESM.

DFHXSS

Entry points

DFHXSSNA

Called by

DFHCSVC

Description

DFHXSS invokes the external security manager (ESM) for all functions that need to be invoked while authorized, except for the EXTRACT functions for which it passes control to DFHXSSB.

DFHXSSB

Entry points

DFHXSSB

Called by

DFHXSS

Description

This module extracts data from the ESM's database. DFHXSSB extracts userid-related data at signon time, and session key information at LU6.2 session bind time. It uses the MVS RACROUTE REQUEST=EXTRACT macro.

DFHXSWM

Entry points

DFHXSWM

Called by

DFHXSMN

Description

DFHXSWM passes and retrieves messages to and from the XRF alternate system to see if security initialization is required in the XRF environment.

DFHXTCI

Entry points

DFHXTCI

Description

DFHXTCI is the transaction invoked when the alternate system begins a takeover. It examines the TCT to locate the terminals with XRF backup sessions, and queues these TCTTEs to DFHZSES for the SESSIONC CONTROL=SWITCH command.

DFHXTP

Entry points

DFHXTPNA

Called by

DFHTPS, DFHZTSP, DFHZXRL, DFHZXRT

Description

The terminal sharing transformation program comprises four logical modules (known as transformers 1 through 4). DFHXTP transforms routing requests into the LU type 6 format for shipping to a remote CICS address space.

DFHZABD

Entry points

DFHZABD1

Called by

TC CTYPE= requests

Description

If a TC CTYPE request is issued when ZCP has been generated without z/OS Communications Server support, DFHZABD is invoked to abend the transaction.

DFHZACT

Entry points

DFHZACT1

Called by

DFHZDSP

Description

The activate scan routine scans the four TCTTE activity queues: activate, log, wait, and NACP. DFHZACT scans the activate queue for request bits that may be set in the TCTTEs; for each request, DFHZACT calls the appropriate module. If no requests are outstanding, the TCTTE is removed from the queue. If the NACP queue is not empty, DFHZACT attaches DFHZNAC (if not already attached). Similarly, if the log queue is not empty, DFHZACT attaches DFHZRLG. DFHZACT scans the wait queue. If automatic resource definition is in the system, DFHZACT looks for any corresponding work elements. For each work element, DFHZATA is attached.

DFHZAIT

Entry points

DFHZAIT1

Called by

DFHSIF1

Description

The attach initialization tables routine initializes local tables used by the mainline task-attach routine, DFHZATT. DFHZAIT generates the page command table from information supplied by the system initialization table, modifying it for use by DFHZATT. DFHZAIT also initializes the transaction code delimiter table.

DFHZAND

Entry points

DFHZAND1

Called by

DFHZARQ

Description

The abend control block builder is used to assist in building the transaction abend block when an abend has occurred in an interconnected system. Its function is to extract the error sense bytes, and the diagnostic message sent by the other system, and to copy these into the block. As an initial step in its processing, DFHZAND acquires storage for the block itself.

DFHZARER

Entry points

DFHZARER

Called by

DFHZARL, DFHZARR, DFHZARRA

Description

DFHZARER tidies up after an LU6.2 protocol error or session failure has been detected. For some errors, it calls DFHZNAC.

DFHZARL

Entry points

DFHZARL1

Called by

DFHACP, DFHCPCBA, DFHCPCLC, DFHCRS, DFHEGL, DFHETL, DFHLUP, DFHXFP, DFHXTP, DFHZARL, DFHZARM, DFHZERH, DFHZISP, DFHZLUS, DFHZSUP, DFHZTSP, DFHZXRL, DFHZXRT

Description

DFHZARL is called via the DFHLUC macro, which passes the LU6.2 request in a parameter list mapped by the DFHLUCDS DSECT. If the request is for a remote APPC device, DFHZARL passes the parameter list to DFHZXRL for processing. (APPC is advanced program-to-program communication.) Otherwise, it examines the parameter list to determine the function required. Most functions are processed by DFHZARL. However, it calls the following modules as indicated:

```
DFHZARER - Protocol errors and exceptions
DFHZARR  - RECEIVE requests
DFHZARRA - FREE-STORE requests
DFHZERH  - Handling FMH7s and negative responses
DFHZISP  - ALLOCATE and FREE requests
DFHZRVL  - Receiving SNA indicators from z/OS Communications Server
DFHZSDL  - Sending data to z/OS Communications Server.
```

It also manages the logical receive buffer pointers TCTERBLA and TCTERBLL in a consistent manner with the physical receive buffer pointers TCTERBA and TCTERBDL, as (address, length) pairs.

DFHZARM

Entry points

DFHZARM1

Called by

DFHZARQ, DFHETL, DFHZISP

Description

DFHZARM handles DFHTC macros for LU6.2 sessions.

DFHZARQ

Entry points

DFHZARQ1

Called by

DFHETC, DFHTC macro

Description

The application request interface module analyzes the terminal control request from the application. For a z/OS Communications Server terminal, it sets the appropriate flags and calls the required module or adds the TCTTE to the activate chain.

DFHZARR

Entry points

DFHZARR

Called by

DFHZARL

Description

DFHZARR controls the receive function for LU6.2 application requests. It calls DFHZARRC to decide what to process next, or whether it is necessary to call its inline subroutine DFHZARR1 to receive more data. Then it processes the returned item, and decides whether the receive is complete. If the receive is not complete, DFHZARR loops, calling DFHZARRC and processing the returned item, until enough data has been received. DFHZARR uses the inline subroutine DFHZARR0 and the DFHZARRA module to control various receive buffers. It also uses DFHZARRF to receive FMH7s and negative responses, DFHZUSR to control the

conversation state, and the inline subroutine DFHZARR1 to handle the type of receive and how much data is to be received.

DFHZARR0 is responsible for updating the logical buffer pointers TCTERBLA and TCTERBLL, shifting up data in the LU6.2 receive buffer, and resetting associated indicators, for example, TCTECCDR in the TCTTE LUC extension.

DFHZARR1 is responsible for setting fields TCTEMINL and TCTEMAXL in the TCTTE LUC extension to inform DFHZRVL how much data to receive and whether the request is a receive immediate or a receive and wait. DFHZARR1 calls DFHZARR0 to shift up data in the LU6.2 receive buffer, and then calls DFHZRVL to receive RUs from z/OS Communications Server by placing requests on the active chain.

DFHZARRA

Entry points

DFHZARRA

Called by

DFHZARL, DFHZARR

Description

DFHZARRA controls all functions concerned with the LU6.2 application receive buffer. These include GETMAIN and FREEMAIN of buffers, copying data into a buffer, and updating the pointer to the next free slot.

DFHZARRC

Entry points

DFHZARRC

Called by

DFHZARR

Description

DFHZARRC is responsible for examining what has been received from z/OS Communications Server on a particular session (for example, data, PS headers, FMH7s, and indicators), and for deciding what should be processed next on behalf of the application. The result is returned to DFHZARR.

DFHZARRF

Entry points

DFHZARRF

Called by

DFHZARR

Description

DFHZARRF receives LU6.2 FMH7s and negative responses. It calls the DFHZARR0 subroutine to shift up data in the LU6.2 receive buffer, and then calls DFHZERH.

DFHZASX

Entry points

DFHZASX1

Called by

z/OS Communications Server

Description

The asynchronous command exit module is called by z/OS Communications Server if an asynchronous command is received. The only such commands are request shutdown, quiesce at end of chain, release quiesce, and signal. DFHZASX sets up the TCTTE appropriately and returns control to z/OS Communications Server.

DFHZATA

Entry points

DFHZATA

Called by

DFHZACT

Description

The autoinstall program runs as the CATA transaction and performs operations necessary to INSTALL autoinstallable terminals. It requests information from a user program where appropriate.

DFHZATD

Entry points

DFHZATD

Called by

DFHZACT, DFHZNAC

Description

The autoinstall delete program runs as the CATD transaction and performs operations necessary to DELETE autoinstalled terminals. It requests information from a user program where appropriate.

DFHZATDX

Entry points

DFHZATDX

Called by

DFHZATA, DFHZATD

Description

DFHZATDX is the user program for autoinstall. It is called when:

- An autoinstall INSTALL is in progress
- An autoinstall DELETE has just completed
- An autoinstall INSTALL has failed.

For INSTALL, DFHZATDX selects a model name and the corresponding TRMIDNT to be used by the terminal control builder program (DFHTBSxx). This program can be used as a model for a user program.

DFHZATI

Entry points

DFHZATI1

Called by

DFHZACT

Description

The automatic task initiation module checks for stress conditions, calls DFHZSIM if the node is not in session, acquires an RPL if necessary, and issues a conditional DFHKC TYPE=AVAIL macro. DFHZATI initiates bid protocols to decide whether the LU is available.

DFHZATMD

Entry points

DFHZATMD

Called by

DFHZATMF

Description

This program deletes all remote terminal definitions that are flagged (by DFHZATMF) for deletion.

DFHZATMF

Entry points

DFHZATMF

Called by

Description

This program flags remote terminals for Mass-deletion (by DFHZATMD). It is a part of the transaction routing component, and is started to flag all skeletons that have been unused for more than the terminal latency period for deletion.

DFHZATR

Entry points

DFHZATR

Called by

DFHZATR, DFHZXRE0

Description

The autoinstall restart program runs as the CATR transaction at CICS startup after the time period specified in the AIRDELAY parameter. DFHZATR scans all autoinstalled terminals, and causes the CATD transaction to be called to delete any autoinstalled terminals that have not been used during the AIRDELAY interval.

DFHZATS

Entry points

DFHZATS

Called by

DFHZTSP, DFHCRS

Description

The remote autoinstall program runs as the following four transactions:

CITS The remote autoinstall function that is attached by DFHZTSP.

CDTS The remote delete function that is attached by DFHZTSP or DFHCRS.

CFTS The remote reset function that flags terminals for mass deletion after a CICS restart and is attached by DFHZTSP or DFHCRS.

CMTS The mass delete function of remote terminals that is attached by DFHZATS transaction CFTS if it finds any terminals for deletion.

DFHZATT

Entry points

DFHZATT1

Called by

DFHZACT

Description

The task attach module checks for stress conditions, allocates an RPL if necessary, and determines the task to be attached either from the data, or from the TCTTE (if the previous transaction specified TRANID), or from the AID (for a 3270). DFHZATT also checks for paging commands (having been modified by DFHZAIT). Finally a conditional ATTACH is issued. The module is applicable for z/OS Communications Server, SRL, and MVS console support.

DFHZBAN

Entry points

DFHZBAN

Called by

DFHZOPN

Description

The terminal control bind analysis program checks that a bind is valid and supportable and, if requested, sets the TCTTE information that supports the session parameters.

DFHZBKT

Entry points

DFHZBKT1

Called by

DFHZSDL, DFHZSLX, DFHZRLX, DFHZLUS

Description

DFHZBKT maintains the bracket state for LU6.2.

DFHZBLX

Entry points

DFHZBLX

Called by

DFHZSCX

Description

DFHZBLX is the part of of SCIP exit which processes LU6.2 binds. It matches a TCTTE to the BIND and schedules DFHZOPN to complete the BIND process. This module returns to z/OS Communications Server.

DFHZCA

Entry points

DFHZCANA

Called by

See component submodules

Description

DFHZCA is the name of the load module created when the following modules are link-edited together:

```
DFHZACT  - Activate scan
DFHZFRE  - FREEMAIN request
DFHZGET  - GETMAIN request
DFHZQUE  - Chaining
DFHZRST  - RESETSR.
```

DFHZCB

Entry points

DFHZCBNA

Called by

See component submodules

Description

DFHZCB is the name of the load module created when the following modules are link-edited together:

DFHZATI
: Automatic task initiation

DFHZDET
: Task detach

DFHZHPSR
: HPO send/receive

DFHZLRP
: Logical record presentation

DFHZRAC
: Receive-any completion

DFHZRAS
: Receive-any slowdown processing

DFHZRVS
Receive specific
DFHZRVX
Receive specific exit
DFHZSDR
Send response
DFHZSDS
Send DFSYN
DFHZSDX
Send DFSYN data exit
DFHZSSX
Send DFSYN exit
DFHZUIX
User input exit

DFHZCC

Entry points

DFHZCCNA

Called by

See component submodules

Description

DFHZCC is the name of the load module created when the following modules are link-edited together:
DFHZARER
LU6.2 protocol error and exception handler
DFHZARL
LU6.2 application request logic
DFHZARM
LU6.2 migration logic
DFHZARR
LU6.2 application receive request logic
DFHZARRA
LU6.2 application receive buffer support
DFHZARRC
LU6.2 classify what next to receive
DFHZARRF
LU6.2 receive FMH7 and ER1
DFHZBKT
LU6.2 bracket state machine
DFHZCHS
LU6.2 chain state machine
DFHZCNT
LU6.2 contention state machine
DFHZCRT
LU6.2 RPL_B state machine
DFHZRLP
LU6.2 post-z/OS Communications Server receive logic
DFHZRLX
LU6.2 receive exit program

DFHZRVL
LU6.2 pre-z/OS Communications Server receive logic
DFHZSDL
LU6.2 send logic
DFHZSLX
LU6.2 send exit program
DFHZSTAP
MRO or LU6.2 conversation state determination
DFHZUSR
LU6.2 conversation state machine

DFHZCHS

Entry points

DFHZCHS1

Called by

DFHZRLX, DFHZSDL, DFHZSLX

Description

DFHZCHS maintains the chain state for LU6.2.

DFHZCLS

Entry points

DFHZCLS1

Called by

DFHZACT

Description

The close destination module obtains an RPL if necessary, issues CLSDST to z/OS Communications Server, and checks if it was accepted. The CLSDST exit handles the completion of the request. DFHZCLS performs a normal closedown procedure according to the LU type (for example, LU6 sends SBI and BIS). In the case of an abnormal closedown, DFHZCLS performs immediate termination, using CLSDST or TERMSESS commands. If the terminal was automatically defined, it is put out of service.

DFHZCLX

Entry points

DFHZCLX1

Called by

z/OS Communications Server

Description

The close destination exit module receives control from z/OS Communications Server when a CLSDST or TERMSESS request completes. If the CLSDST or TERMSESS was successful, DFHZCLX cleans up TCTTE and returns to z/OS Communications Server; otherwise it enqueues the TCTTE to DFHZNAC and then returns to z/OS Communications Server.

DFHZCNA

Entry points

DFHZCNA1

Called by

DFHZDSP

Description

The system console activity control program is responsible for CICS system requests. It performs the following functions:

- Shutdown—when all other access method terminals have been quiesced, quiesces console support, allowing CICS to terminate.
- Resume—resumes tasks waiting on read request when they are completed.
- Detach—releases all TIOAs associated with a completed task.
- Attach—passes the data associated with a MODIFY command (in a TIOA attached to a console TCTTE) to DFHZATT to create a task.
- ATI—determines whether a console TCTTE is available for automatic task initiation.

DFHZCNR

Entry points

DFHZCNR1

Called by

DFHZARQ

Description

The system console application request program performs READ, WRITE, and CONVERSE operations to an MVS system console that is used as a terminal.

DFHZCNT

Entry points

DFHZCNT1

Called by

DFHZLUS, DFHZRLX

Description

DFHZCNT maintains the contention state for LU6.2.

DFHZCP

Entry points

DFHZCPNA

Called by

See component submodules

Description

DFHZCP is the name of the load module created when the following modules are link-edited together:

```
DFHZARQ  - Application request handler
DFHZATT  - Attach routine
DFHZCNA  - System console activity control
DFHZDSP  - Dispatcher
DFHZISP  - Allocate/free/point routine
DFHZSUP  - Startup task
DFHZUCT  - 3270 uppercase translation.
```

DFHZCQ

Entry points

DFHZCQ

Called by

DFHAMTP, DFHCRS, DFHQRY, DFHTCRP, DFHWKP, DFHZATA, DFHZATD, DFHZTSP, DFHZXCU

Description

DFHZCQ is the control program for all requests for the dynamic add and delete of terminal control table entries. It is called by resource definition online (RDO) to:

- Cold start group lists
- Cold or warm start nonmigrated z/OS Communications Server resources
- Dynamically install using the CEDA transaction.

The main subroutines of DFHZCQ are:

```
DFHZCQCH - Catalog a TCT element
DFHZCQDL - Delete
DFHZCQIN - Initialize DFHZCQ
DFHZCQIQ - Inquire about TCTTE
DFHZCQIS - Install TCTTE
DFHZCQIT - Add macro-generated TCTTE
DFHZCQRS - Restore ZC resource.
```

DFHZCQDL

Entry points

DFHZCQDL

Called by

DFHZCQ00, DFHZNAC, RDO

Description

DFHZCQDL dynamically deletes a TCT entry when the entry is quiesced. This module is part of DFHZCQ.

DFHZCQIN

Entry points

DFHZCQIN

Called by

DFHTCRP

Description

DFHZCQIN initializes DFHZCQ for all its operations. This module is part of DFHZCQ.

DFHZCQIQ

Entry points

DFHZCQIQ

Called by

DFHZTSP

Description

DFHZCQIQ obtains the parameters for a TCT resource and is called by DFHZTSP in the terminal-owning node as part of the process of shipping a TCT definition to a remote system. This module is part of DFHZCQ.

DFHZCQIS

Entry points

DFHZCQIS

Description

DFHZCQIS installs a TCTTE. If the resource already exists, the old resource is deleted.

DFHZCQIT

Entry points

DFHZCQIT

Description

DFHZCQIT adds a macro-generated TCTTE to a CICS system.

DFHZCQRS

Entry points

DFHZCQRS

Description

During emergency restart or warm start, DFHTCRP restores terminal control resources to the state they were in before the last shutdown of CICS, using the restart data set.

DFHZCRQ

Entry points

DFHZCRQ1

Called by

TC CTYPE requests

Description

The CTYPE request module analyzes DFHTC CTYPE commands, and calls or links to the appropriate send module.

DFHZCRT

Entry points

DFHZCRT1

Called by

DFHZACT, DFHZARL, DFHZFRE, DFHZNAC, DFHZRAC, DFHZRLP, DFHZRVL, DFHZSDL, DFHZSHU, DFHZSTU, DFHZTPX

Description

DFHZCRT maintains the RPL_B state for LU6.2.

DFHZCUT

Entry points

DFHZCUT

Called by

DFHCSSC, DFHLUP, DFHSNAT, DFHTCPLR

Description

DFHZCUT manages the persistent verification signed-on-from list, also known as the local userid table (LUIT). There is one LUIT per connection supporting persistent verification.

DFHZCW

Entry points

DFHZCWNA

Called by

See component submodules

Description

DFHZCW is the name of the load module created when the following modules are link-edited together:

```
DFHZERH  - LU6.2 error program
DFHZEV1  - LU6.2 BIND security
DFHZEV2  - LU6.2 BIND security
DFHZLUS  - LU6.2 session management program.
```

DFHZCX

Entry points

DFHZCXNA

Called by

See component submodules

Description

DFHZCX is the name of the load module created when the following modules are link-edited together:

```
DFHZABD  - Abend routine for incorrect requests
DFHZAND  - Build TACB before issuing PC abends
DFHZCNR  - System console application request
DFHZIS1  - ISC or IRC syncpoint
DFHZIS2  - IRC internal requests
DFHZLOC  - Locate TCTTE and ATI requests
DFHZSTU  - Terminal control status change.
```

DFHZCXR

Entry points

DFHZCXRA

Called by

See component submodules

Description

DFHZCXR is the generic name allocated to a composite module that is not called by any other code. It includes the following transaction-routing related modules:

```
DFHZTSP  - Terminal-sharing program
DFHZXRL  - Routes LU6.2 commands to TOR
DFHZXRT  - Receives LU6.2 commands from AOR.
```

DFHZCY

Entry points

DFHZCYNA

Called by

See component submodules

Description

DFHZCY is the name of the load module created when the following modules are link-edited together:

DFHZASX
: DFASY exit

DFHZDST
: SNA-ASCII translation

DFHZLEX
: LERAD exit

DFHZLGX
: LOGON exit

DFHZLTX
: LOSTERM exit

DFHZNSP
: Network services exit

DFHZOPA
: Open z/OS Communications Server ACB

DFHZRRX
: Release request exit

DFHZRSY
: Resynchronization

DFHZSAX
: Send synchronous command exit

DFHZSCX
: SESSION control input exit

DFHZSDA
: Send synchronous command

DFHZSES
SESSIONC
DFHZSEX
SESSIONC exit
DFHZSHU
Shutdown z/OS Communications Server
DFHZSIM
SIMLOGON
DFHZSIX
SIMLOGON exit
DFHZSKR
Send response to command
DFHZSLS
Set logon start
DFHZSYN
Handle CTYPE=SYNC or CTYPE=RECOVER request
DFHZSYX
SYNAD exit
DFHZTPX
TPEND exit
DFHZTRA
Create ZCP or VIO trace requests
DFHZXRC
XRF session state data analysis

DFHZCZ

Entry points

DFHZCZNA

Called by

See component submodules

Description

DFHZCZ is the name of the load module created when the following modules are link-edited together:

```
DFHZCLS  - CLSDST
DFHZCLX  - CLSDST exit
DFHZCRQ  - Command request
DFHZEMW  - Error message writer
DFHZOPN  - OPNDST
DFHZOPX  - OPNDST exit
DFHZRAQ  - Read-ahead queuing
DFHZRAR  - Read-ahead retrieval
DFHZTAX  - Turnaround exit.
```

DFHZDET

Entry points

DFHZDET1

Called by

DFHZACT, DFHZISP

Description

The task detach module receives control when a detach request is issued by DFHZISP. If a WRITE is pending (deferred write or any write), the SEND routine is called. If the SEND cannot complete, the DETACH request stays on the activate queue. If requests are queued then DFHZACT drives DFHZDET when the operation is complete. If the node is in between bracket state, an end bracket is sent.

DFHZDSP

Entry points

DFHZDSP1

Called by

DFHSII1

Description

The dispatcher module handles the dispatching of modules for execution, and gives control to z/OS Communications Server modules of ZCP using DFHZACT.

DFHZDST

Entry points

DFHZDST1

Called by

DFHZRVX, DFHZSDS

Description

The data stream translator module translates data between EBCDIC and ASCII code while that data is being sent and received on z/OS Communications Server sessions.

DFHZEMW

Entry points

DFHZEMW1

Called by

DFHACP, DFHZDET, DFHZNAC, DFHZRAC

Description

The error message writer module handles all requests for error messages on z/OS Communications Server supported terminals/LUs. According to the request flags, it:

- Sends a negative response
- Purges unprocessed inbound data until EOC or CANCEL is received
- Sends an error message.

DFHZERH

Entry points

DFHZERH1

Called by

DFHZARL, DFHZARRF

Description

DFHZERH handles the sending and receiving of LU6.2 FMH7s and negative responses. It also manages the logical receive buffer pointers TCTERBLA and TCTERBLL in a consistent manner with the physical receive buffer pointers TCTERBA and TCTERBDL, as (address, length) pairs.

DFHZEV1

Entry points

DFHZEV11

Description

DFHZEV1 is the LU6.2 bind-time security encryption validation program, part 1.

DFHZEV2

Entry points

DFHZEV21

Description

DFHZEV2 is the LU6.2 bind-time security encryption validation program, part 2.

DFHZFRE

Entry points

DFHZFRE1

Called by

DFHZACT, DFHZEMW, DFHZCLS, DFHZCLX

Description

The FREEMAIN module is used to free storage (RPLs, NIBs, bind areas, TIOAs, buffer lists, LUC send/receive buffers, and extract logon data) acquired by ZC modules. Some storage is also freed by other ZC modules.

DFHZGET

Entry points

DFHZGET1

Called by

DFHZACT, DFHZARL, DFHZATI, DFHZATT, DFHZCLS, DFHZISP, DFHZOPN, DFHZRAC, DFHZRST, DFHZRSY, DFHZRVL, DFHZRVS, DFHZSDA, DFHZSDL, DFHZSDR, DFHZSDS, DFHZSES, DFHZSKR

Description

The GETMAIN module is used to acquire an RPL, NIB, bind area, TIOA, buffer list, or LUC send/receive buffer. DFHZGET also sets up the dynamic NIB using the information in the NIB descriptor block. Normally, when a ZC module requires some of this storage, it invokes DFHZGET to obtain the storage; if this is unsuccessful, it may queue the request, and then DFHZACT calls DFHZGET on behalf of the caller.

DFHZHPRX

Entry points

DFHZHPNA

Called by

DFHKCSP (via DFHZHPSR and DFHKCP)

Description

In authorized path SRB mode, DFHZHPRX issues z/OS Communications Server EXECRPL.

DFHZHPSR

Entry points

DFHZHPS1

Called by

DFHZRVS, DFHZSDS

Description

DFHZHPSR is the SEND and RECEIVE module for the HPO environment.

DFHZISP

Entry points

DFHZISP1

Called by

DFHISP, DFHKCP

Description

The intersystem program services ISC requests to free, or point to, a particular TCTTE within a specified system, or to allocate a TCTTE within a specified system. DFHZISP also handles ATI requests, and checks for a terminal timeout.

DFHZIS1

Entry points

DFHZIS11

Description

DFHZIS1 handles the transmissions control CTYPE requests of Prepare, Syncpoint Request (SPR), Commit, and Abort. Each request is translated into the appropriate ISC/IRC action and is transmitted to the connected system.

DFHZIS2

Entry points

DFHZIS21

Called by

DFHZARQ, DFHZIS1

Description

The intersystem program provides services for CICS system code that wants to use intersystem or interregion (IRC) function requests:

RECEIVE
Is invoked when DFHCRNP gets input data as a result of a 'switch first' SVC request.

IOR The IRC input/output routine. This interfaces with the IRC SVC in order to send data to the other end of the connection, or await data from there.

GETDATA
Is used to fetch input data into a TIOA.

DISCONNECT
Disconnects a given IRC link.

STOP Quiesces interregion activity, either for connections to a given system, or for the whole of IRC.

LOGOFF
Issues a logoff request to the IRC SVC. This completes IRC activity for this CICS system.

OPERATIVE
Allows connections to be made to a given system.
RECABRT
processes input abend FMHs (FMH07).

DFHZLEX

Entry points

DFHZLEX1

Called by

z/OS Communications Server

Description

The logical error address (LERAD) exit module receives control from z/OS Communications Server when a logical error is detected. Logical errors are usually the result of an incorrectly defined terminal table.

DFHZLGX

Entry points

DFHZLGX1

Called by

z/OS Communications Server

Description

The logon exit module receives control from z/OS Communications Server when a terminal logs on to the network. DFHZLGX scans the CICS NIBs and, if a match is found, sets an OPNDST request in the corresponding TCTTE and places it on the activate queue. If no match is found, DFHZLGX defines a terminal automatically, if possible, by allocating an autodefine work element which holds the CINIT_RU. The work element is then queued for activate scan processing. Otherwise, a dummy TCTTE is placed on the NACP queue to write an error message.

DFHZLOC

Entry points

DFHZLOC1

Called by

DFHTC CTYPE=LOCATE

Description

The locate module provides two functions:

- Locates specific TCTTEs, TCTSEs, and SESSIONs in the TCT

- Locates LDC information.

DFHZLRP

Entry points

DFHZLRP1

Called by

DFHZARQ, DFHZSUP

Description

The logical record presentation module handles deblocking of input data. The delimiters that are recognized are new line (NL), interchange record separator (IRS), and transparent (TRN). One logical record is returned for each DFHTC TYPE=READ request.

DFHZLTX

Entry points

DFHZLTX1

Called by

z/OS Communications Server

Description

The lost terminal (LOSTERM) exit module receives control when z/OS Communications Server detects a loss of contact with a node. There are three possible return codes set by z/OS Communications Server on entry to this routine:

node lost, recovery in progress
: The terminal is placed out of service with no further action taken.

node lost, recovery successful
: The TCTTE is queued to the NACP queue with a ‘successful' error code set; NACP issues a CLSDST, schedules a SIMLOGON, and issues an information message.

node lost, no recovery or unsuccessful recovery
: The TCTTE is queued to the NACP queue with an ‘unsuccessful' error code set; NACP issues a CLSDST and also the appropriate message.

DFHZLUS

Entry points

DFHZLUS1

Description

DFHZLUS handles session management for LU6.2 sessions.

DFHZNAC

Entry points

DFHZNANA

Called by

DFHZACT

Description

The node abnormal condition program is attached by DFHZACT when an error in communication with a logical unit occurs. DFHZNAC performs the following functions:

- Analyzes abnormal conditions
- Sends appropriate messages to the CSNE transient data destination
- Invokes the user-supplied (or sample) node error program
- Takes the appropriate actions resulting from the defaults which may have been modified by the node error program.

DFHZNAC consists of the following copybooks:

```
DFHZNCA  - Primary error action table and exits
DFHZNCE  - Take action routine
DFHZNCS  - Sense decode routine
DFHZNCV  - z/OS Communications Server return code routine.
```

DFHZNEP

Entry points

DFHZNENA

Called by

DFHZNAC

Description

The user-replaceable node error program provides:

- A general environment within which it is easy for users to add their own error processors
- Fundamental error recovery actions for a z/OS Communications Server 3270 network
- The default NEP where the user selects a NEP at system initialization.

DFHZNSP

Entry points

DFHZNSP1

Called by

z/OS Communications Server

Description

The network service program is invoked when z/OS Communications Server detects a network service error; for example, when attempting to connect two nodes together, or when the link between two nodes is broken unexpectedly. This module receives control from the z/OS Communications Server NSEXIT.

DFHZOPA

Entry points

DFHZOPA1

Called by

DFHEIQVT

Description

The open VTAM ACB module is invoked by DFHEIQVT when the master terminal command **SET VTAM OPEN** is issued. The ACB is opened and DFHZSLS is called to accept logon requests.

VTAM is now known as z/OS Communications Server.

DFHZOPN

Entry points

DFHZOPN1

Called by

DFHZACT

Description

The open destination module acquires storage for an RPL and NIB and BIND areas if the TCTTE does not have these resources already, and sets up the BIND image if required. DFHZOPN then issues a z/OS Communications Server OPNDST macro (or OPNSEC macro if secondary, to respond to an incoming BIND) to establish a session between CICS and the remote LU.

DFHZOPX

Entry points

DFHZOPX1

Called by

z/OS Communications Server

Description

The open destination exit module receives control from z/OS Communications Server on completion of the OPNDST macro in DFHZOPN. If the OPNDST was successful, it indicates in the TCTTE that SDT (start data transfer) is to be sent and checks whether a “good morning” message should be triggered. It then returns to z/OS Communications Server.

DFHZQUE

Entry points

DFHZQUE1

Called by

All ZCP exits called by z/OS Communications Server, DFHTCQUE macro

Description

The queue manipulation module processes all requests to add or remove a TCTTE to or from a ZCP activate queue. Additions to the activate queue made by mainline modules use compare-and-swap (CS), because an exit routine may also be adding to the queue asynchronously.

DFHZRAC

Entry points

DFHZRAC1

Called by

DFHZDSP

Description

The receive-any completion module processes the completion of receive-any requests, sets up the TIOA to be passed to attach, and reissues the RECEIVE_ANY macro.

DFHZRAQ

Entry points

DFHZRAQ1

Called by

DFHZARQ, DFHZSYN

Description

The read-ahead queuing module is used to save the inbound data stream in temporary storage when an interlock is caused by both the host and the terminal wanting to send data at the same time.

DFHZRAR

Entry points

DFHZRAR1

Called by

DFHZARQ

Description

The read-ahead retrieval module is called to retrieve data previously saved in temporary storage by DFHZRAQ.

DFHZRAS

Entry points

DFHZRAS1

Called by

DFHZRAC

Description

The receive-any slowdown processing module issues RECEIVE SPEC NQs on LU6.2 sessions for connections and modegroups for which there are ALLOCATE requests queued. This is only done on sessions considered most likely to lead to freeing a "flooding" situation that occurred when LU6.2 connections were reestablished after a failure.

DFHZRLG

Entry points

DFHZRLNA

Called by

DFHZACT

Description

The response logger program logs responses received for protected data sent to an APB. DFHZRLG processes TCTTEs on the log queue when attached by DFHZACT.

DFHZRLP

Entry points

DFHZRLP1

Called by

DFHZDSP

Description

DFHZRLP handles the completion of LU6.2 RECEIVE requests, using the receive RPL addressed by field TCTERPLB in the TCTTE LUC extension. It also manages the logical receive buffer pointers TCTERBLA and TCTERBLL in a consistent manner with the physical receive buffer pointers TCTERBA and TCTERBDL, as (address, length) pairs.

DFHZRLX

Entry points

DFHZRLX1

Called by

z/OS Communications Server

Description

DFHZRLX is a z/OS Communications Server exit routine that queues the completed RPL for (post-z/OS Communications Server) processing by DFHZRLP.

DFHZRRX

Entry points

DFHZRRX1

Called by

z/OS Communications Server

Description

The release request exit module receives control from z/OS Communications Server when another application program has requested connection to a terminal currently connected to CICS. If the terminal is not busy, a CLSDST request is queued to the activate chain. Otherwise the release request indicator is set and the request is processed later by module DFHZDET.

DFHZRSP

Entry points

DFHZRSNA

Description

The resynchronization send program performs 3614-dependent actions and is also used to retransmit committed output messages. The message is retrieved from temporary storage if necessary.

DFHZRST

Entry points

DFHZRST1

Called by

DFHZACT, DFHZATI, DFHZCRQ, DFHZDET, DFHZEMW, DFHZERH, DFHZNAC, DFHZRAC, DFHZRSY, DFHZSTU

Description

The RESETSR module changes the mode of a session with a terminal and cancels unsatisfied RECEIVE requests. The mode that is set can be Continue Any (CA) or Continue Specific (CS) and RTYPE=DFSYN, DFASY, or RESP.

DFHZRSY

Entry points

DFHZRSY1

Called by

DFHZACT

Description

The resynchronize module resynchronizes CICS and other nodes of the network. DFHZRSY checks whether inbound and outbound sequence numbers are valid.

DFHZRVL

Entry points

DFHZRVL1

Called by

DFHZARL, DFHZARRL

Description

DFHZRVL processes RECEIVE commands for LU6.2 sessions, using the receive RPL (RPL_B) addressed by field TCTERPLB in the TCTTE LUC extension. The processing state of the receive RPL is held in the RPL_B state machine field TCTERPBS, also in the TCTTE LUC extension.

DFHZRVS

Entry points

DFHZRVS1

Called by

DFHZACT

Description

The `receive specific` module initiates a `DFSYN receive specific` to obtain the next logical record from a node when a user application issues a RECEIVE command.

DFHZRVX

Entry points

DFHZRVX1

Called by

z/OS Communications Server

Description

The receive specific exit module receives control from z/OS Communications Server when a receive specific is completed. If the data received is too long for the TIOA provided, the overlength data flag is turned on in the TCTTE and the TCTTE is put back on the activate chain. Otherwise, the response is checked and marked in the TCTTE. The data length is set in the TIOA and the FMH is removed.

DFHZSAX

Entry points

DFHZSAX1

Called by

z/OS Communications Server

Description

The send DFASY exit module receives control from z/OS Communications Server when an asynchronous command has completed. It places the TCTTE on the NACP queue if recovery is needed.

DFHZSCX

Entry points

DFHZSCX1

Called by

z/OS Communications Server

Description

The SCIP exit module is entered whenever the following asynchronous commands are received:

- Non-LU6.2 BIND (as secondary)
- UNBIND (as secondary)
- STSN (as secondary)
- Clear (as secondary)
- SDT (as secondary)
- Request recovery (as primary).

The module correlates BINDs to a TCTTE and schedules DFHZOPN to complete the BIND process. For the other commands, it takes appropriate action and then schedules DFHZNAC using the NACP queue. This module calls DFHZBLX to process LU6.2 binds.

DFHZSDA

Entry points

DFHZSDA1

Called by

DFHZACT, DFHZSDS

Description

The send data flow asynchronous module handles asynchronous command requests. It ensures that an RPL is allocated, primes the RPL for the requested command, and issues the z/OS Communications Server asynchronous send macro.

DFHZSDL

Entry points

DFHZSDL1

Called by

DFHZARL

Description

DFHZSDL processes SEND commands for LU6.2 sessions, using the RPL addressed by field TCTERPLA in the TCTTE.

DFHZSDR

Entry points

DFHZSDR1

Called by

DFHZACT, DFHZCRQ, DFHZDET, DFHZRVS, DFHZSDA, DFHZSDS

Description

The send response module sends responses to nodes when a synchronization request for a terminal is made and a response is outstanding from a previous operation. If errors occur during task initiation, this module is responsible for the negative response.

DFHZSDS

Entry points

DFHZSDS1

Called by

DFHZACT, DFHZARQ, DFHZATI, DFHZATT, DFHZDET

Description

The send data synchronous module sets up and issues the appropriate z/OS Communications Server send macro for requests of “send data” or an SNA synchronous command.

DFHZSDX

Entry points

DFHZSDX1

Called by

z/OS Communications Server

Description

The send data synchronous exit module receives control from z/OS Communications Server when a SEND request is complete. It checks the RPL for successful completion of the message sent and takes appropriate action.

DFHZSES

Entry points

DFHZSES1

Called by

DFHZACT, DFHZRSY

Description

The session control module is entered whenever a session control command is requested by CICS. It sets up and issues the z/OS Communications Server SESSIONC command.

DFHZSEX

Entry points

DFHZSEX1

Called by

z/OS Communications Server

Description

The SESSIONC exit module receives control from z/OS Communications Server when a SESSIONC command has completed. If the command was successful, it turns off the corresponding flags and enqueues the TCTTE on the activate chain. If the completion was not successful, the TCTTE is placed on the NACP queue for recovery processing.

DFHZSHU

Entry points

DFHZSHU1

Called by

DFHZDSP

Description

The close z/OS Communications Server ACB module is invoked whenever CICS and z/OS Communications Server are being uncoupled. This may be as a result of DFHZTPX being driven as the result of a z/OS Communications Server halt command or the issue of the master terminal command SET VTAM,CLOSE|IMMCLOSE. The status of all sessions is checked and, when all are inactive, the ACB is closed.

DFHZSIM

Entry points

DFHZSIM1

Called by

DFHZACT

Description

The simulate logon module is entered to issue a z/OS Communications Server SIMLOGON or REQSESS (if secondary) request to place a node in session without the operator having to logon. LU6.2 can be selected by mode name.

DFHZSIX

Entry points

DFHZSIX1

Called by

z/OS Communications Server

Description

Whenever a SIMLOGON or REQSESS command has been completed, this exit routine is scheduled by z/OS Communications Server. On successful completion, it turns off the SIMLOGON requested flag and enqueues the TCTTE or TCTME on the activate chain or, if NACP is required, for NACP processing.

DFHZSKR

Entry points

DFHZSKR1

Called by

DFHZACT

Description

The send command response module sends responses to z/OS Communications Server commands including response to BIND, STSN, and SDT. A positive or negative response can be sent. The module is for secondary LU support only.

DFHZSLS

Entry points

DFHZSLS1

Called by

DFHZDSP, DFHZOPA

Description

The SETLOGON start module issues SETLOGON to cause z/OS Communications Server to accept automatic logon requests, and issues the initial RECEIVE ANYs for RPLs in the receive-any pool. DFHZSLS also examines the SIT to determine whether autodefine is used. If it is, the appropriate system initialization parameters are copied to the TCT prefix.

DFHZSLX

Entry points

DFHZSLX1

Called by

z/OS Communications Server

Description

DFHZSLX is a z/OS Communications Server exit routine that handles the completion of LU6.2 SEND requests.

DFHZSSX

Entry points

DFHZSSX1

Called by

z/OS Communications Server

Description

The send data flow synchronous exit module receives control when the send of a DFSYN command has been completed.

DFHZSTAP

Entry points

DFHZSTA1

Called by

DFHEGL, DFHETC, DFHETL

Description

DFHZSTAP determines the state of an MRO or LU6.2 conversation from an application viewpoint.

DFHZSTU

Entry points

DFHZSTU1

Called by

DFHTC CTYPE=STATUS, DFHEIQMT, DFHEIQSC, DFHEIQST

Description

DFHZSTU changes the status of TCTTEs and TCTSEs. It can change the following statuses:

- Inservice
- Outservice
- Intlog | No intlog
- Page | Autopage
- ATI | NATI.

DFHZSUP

Entry points

DFHZSUP1

Called by

DFHKCP

Description

The startup task module is the entry point for all terminal-related tasks. DFHZSUP performs the following functions:

- Sets up the TCTTE status
- Performs security checking
- Performs logging of the TCTTE status and input TIOA
- Performs option checking for transaction definition
- Passes control to transaction program, for example, user application, DFHACP, DFHAPRT.

DFHZSYN

Entry points

DFHZSYN1

Called by

DFHDBP

Description

DFHZSYN handles CTYPE=SYNC and RECOVER requests. For protected message support, DFHSPP issues CTYPE=SYNC to clear protected messages. For RECOVER requests, DFHZSYN ensures that no further I/O is issued to that session, and that UNBIND flows.

DFHZSYX

Entry points

DFHZSYX1

Called by

z/OS Communications Server

Description

The SYNAD exit module receives control from z/OS Communications Server when a catastrophic error is encountered. DFHZSYX determines the type of error and the appropriate action to be taken, and schedules NACP using the NACP queue to complete the recovery processing.

DFHZTAX

Entry points

DFHZTAX1

Called by

z/OS Communications Server

Description

The turnaround exit module is called by z/OS Communications Server on completion of the SEND operation initiated by DFHZRVS in order to perform a turnaround in flip-flop protocol.

DFHZTPX

Entry points

DFHZTPX1

Called by

z/OS Communications Server

Description

The TPEND exit module receives control when z/OS Communications Server is terminating. It schedules a CLSDST for each active session if quick shutdown is required, and sets bits in the TCT prefix so that DFHZSHU is invoked.

DFHZTRA

Entry points

DFHZTRA1

Called by

DFHZACT, DFHZDET, DFHZRAC, DFHZRLP, DFHZRVS, DFHZSDL, DFHZSDR, DFHZSDS

Description

DFHZTRA creates VIO trace entries.

DFHZTSP

Entry points

DFHZTSP1

Called by

DFHAPRT, DFHISP, DFHRTE, DFHTPS, DFHZARQ, DFHZCQ, DFHZSUP

Description

The terminal sharing program acquires a TCTTE for a link to a remote CICS address space, and transfers request data to that space. DFHZTSP also receives requests from the remote address space.

DFHZUCT

Entry points

DFHZUCT1

Called by

DFHAPRT, DFHZARQ, DFHZCNA, DFHZRAC, DFHZRVX, DFHZSUP

Description

The uppercase translate module converts a z/OS Communications Server 3270 data stream into uppercase.

DFHZUIX

Entry points

DFHZUIX1

Called by

DFHZACT, DFHZRAC, DFHZRVX

Description

The user input exit module is called directly (by DFHZRAC) or indirectly (by DFHZRVX via DFHZACT) to link to the user's XZCIN exit.

DFHZUSR

Entry points

DFHZUSR1

Called by

DFHACP, DFHETL, DFHZARER, DFHZARL, DFHZARM, DFHZARR, DFHZARRF, DFHZERH, DFHZOPX, DFHZSTAP, DFHZSUP, DFHZUSR, DFHZXRL, DFHZXRT

Description

DFHZUSR maintains the conversation state for LU6.2.

DFHZXCU

Entry points

DFHZXCU

Description

The z/OS Communications Server XRF catch-up program is used to send messages that allow a new alternate system to catch up with the current state of the active system for:

- TCT contents
- Bound/unbound state of sessions.

The program is invoked when a new alternate system signs on.

DFHZXQO

Entry points

DFHZXQO

Called by

DFHTCRP, DFHZXST

Description

The XRF ZCP tracking queue organizer allows pending XRF tracking activity to be stored in a way that honors interdependencies, while allowing such requests to be met as soon as all their prerequisites are fulfilled. This component consists of a data structure and accessing program that uses the CICS catalog key structure to identify all the actions for a single resource and the dependencies between them. Actions are put into the structure on receipt in DFHTCRP, and removed by DFHTCRP and at the end of DFHZNAC processing for standby BIND and CLSDST completion. The structure is freed at the end of DFHTCRP tracking.

DFHZXRC

Entry points

DFHZXRC1

Called by

DFHZACT

Description

DFHZXRC analyzes the data received in response to the SESSIONC CONTROL=SWITCH command. It determines the state of the session at the point when it was switched, and initiates the necessary action to clean up and recover the session.

DFHZXRE0

Entry points

DFHZXRE0

Called by

System

Description

DFHZXRE0 runs the CXRE transaction to perform autoconnect and XRF reconnect processing. It also starts the acquire process for terminals with flag TCTEXRE set.

DFHZXRL

Entry points

DFHZXRL1

Called by

DFHZARL, DFHZISP

Description

DFHZXRL is executed in an application-owning region. It routes LU6.2 commands to the terminal-owning region.

DFHZXRT

Entry points

DFHZXRT1

Called by

DFHZTSP

Description

DFHZXRT executes in a terminal-owning region. It receives LU6.2 commands from the application-owning region, and issues them to an APPC device.

DFHZXST

Entry points

DFHZXST

Called by

DFHETC, DFHSIJ1, DFHTCRP, DFHTCRPS, DFHZNAC, DFHZOPA, DFHZXCU

Description

XRF ZCP session-state tracking is called by:

- DFHZNAC for BIND/UNBIND completion in the active system, and for standby-BIND and UNBIND in the alternate system
- DFHETC for logon data freed in the active system
- DFHTCRPS to handle a tracking message
- DFHTCRP to terminate session tracking
- DFHZXCU for BIND/UNBIND catch-up in the active system
- DFHSIJ1 and DFHZOPA to issue a SETLOGON START command.

Notices

This information was developed for products and services offered in the U.S.A. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property rights may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

© Copyright IBM Corp. 1997, 2017

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- RACF security overview
- Developing for external interfaces
- Reference: application developmenth
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- Reference: diagnostics

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 4 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services
- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- Supplied Transactions
- CICSPlex SM Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS Transaction Server for z/OS, Version 5 Release 4 , but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below:

For the CICSPlex® SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user’s user name and other personally identifiable information for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other personally identifiable information for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect no personally identifiable information. These cookies cannot be disabled.

For CICS Explorer®:
Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user’s user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled “Cookies, Web Beacons and Other Technologies” and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

A

ABAB gate
  CREATE_ABEND_RECORD function 12
  INQUIRE_ABEND_RECORD function 14
  START_ABEND function 17
  TAKE_TRANSACTION_DUMP function 19
  UPDATE_ABEND_RECORD function 19
ABEND_TERMINATE function, XMAC gate 1398
ABEND_TRANSACTION function, XMER gate 1355
abnormal termination
  system recovery program (SRP) 1787
  transaction failure program (TFP) 1842
ABNORMALLY_TERMINATE_TASK function, KEDS gate 592
ACB (access control block) 1703
ACB (access method control block), VSAM 1603
ACB (access method control block), z/OS Communications Server 1828
ACCEPT function, SOCK gate 1105
access control block (ACB) 1703
access method control block (ACB), VSAM 1603
access method control block (ACB), z/OS Communications Server 1828
access methods, terminal control 1819
ACCUMULATE_RMI_TIME function, MNMN gate 726
ACP (abnormal condition program) 1842
  node 1750
ACQUIRE_ACTIVITY function, BAAC gate 335
ACQUIRE_CONNECTION function, ISCO gate 546
ACQUIRE_PROCESS function, BAPR gate 354
ACQUIRE_PROGRAM function, LDLD gate 624
ACQUIRE_SURROGATE function, APRS gate 44
ACTIVATE_DEBUG_PROFILE function, DPFM gate 419
ACTIVATE_MODE function, DSIT gate 472
ACTIVATE_TRAP function, TRSR gate 1186
adapter, FEPI 1697
  logic flow 1699
ADD function, CCCC gate 367
ADD_ACTIVITY function, BAAC gate 335
ADD_ATOMSERVICE function, W2AT gate 1323
ADD_CHANNEL function, PGCH gate 786
ADD_CRITICAL_MODULE function, KEDS gate 592
ADD_CRITICAL_WINDOW function, KEDS gate 593
ADD_DOMAIN function, DMDM gate 409
ADD_DOMAIN function, KEDD gate 583
ADD_ENTRY function, DDDI gate 381
ADD_EPADAPTER function, EPAD gate 533
ADD_FILE function, FCMT gate 236
ADD_GATE function, KEDD gate 584
ADD_IPCONN function, ISIC gate 550
ADD_LINK function, RMLN gate 951
ADD_LOCK function, LMLM gate 683
ADD_PENDING_REQUEST function, SHPR gate 1034
ADD_PIPELINE function, PIPL gate 889
ADD_POLICY function, MPDD gate 741
ADD_POLICYSCOPE function, MPDD gate 741
ADD_POOL function, TSSH gate 1214
ADD_PROCESS function, BAPR gate 354
ADD_REATTACH_ACQUIRED function, BAAC gate 336
ADD_REPL_ROLE_FOR_METHOD function, XSEJ gate 1418
ADD_REPL_TERM_MODEL, AITM format 1466
ADD_REPLACE_ATOMSERVICE function, W2AT gate 1325
ADD_REPLACE_DOCTEMPLATE function, DHTM gate 399
ADD_REPLACE_ENQMODEL function, NQRN gate 765
ADD_REPLACE_LIBRARY function, LDLB gate 612
ADD_REPLACE_PROCESSTYPE function, BATT gate 359
ADD_REPLACE_TCLASS function, XMCL gate 1348
ADD_REPLACE_TCPIPSERVICE function, SOAD gate 8, 9, 1103
ADD_REPLACE_TDQDEF function, TDTM gate 295
ADD_REPLACE_TRANDEF function, XMXD gate 1379
ADD_REPLACE_TSMODEL function, TSAD gate 1193
ADD_REPLACE_URIMAP function, WBUR gate 1294
ADD_SUBEVENT function, EMEM gate 525
ADD_SUBORDINATE function, OTSU gate 776
ADD_SUBPOOL function, S2AD gate 1088
ADD_SUBPOOL function, SMAD gate 1064
ADD_SUSPEND function, DSSR gate 481
ADD_SYMBOL_LIST function, DHSL gate 397
ADD_SYSTEM_DUMPCODE function, DUDT gate 493
ADD_TCB function, DSIT gate 474
ADD_TCLASS function, XMCL gate 1349
ADD_TIMER_REQUEST function, BAAC gate 336
ADD_TRAN_DUMPCODE function, DUDT gate 494
ADD_TRANSACTION_SECURITY function, XSXM gate 1449
ADD_TRANSACTION_USER function, USXM gate 1245
ADD_TRUSTED_EXTENSION, SMK7 gate 1073
ADD_USER_VIA_ICRX function, USAD gate 1239
ADD_USER_VIA_ICRX function, XSAD gate 1412
ADD_USER_WITH_PASSWORD function, USAD gate 1229
ADD_USER_WITH_PASSWORD function, XSAD gate 1404
ADD_USER_WITHOUT_PASSWORD function, USAD gate 1231
ADD_USER_WITHOUT_PASSWORD function, XSAD gate 1406
address space modules 1729
ADJUST_STCK_TO_LOCAL function, KETI gate 606
ADOPT_PROGRAM function, PGDD gate 812
advanced program-to-program communication (APPC) 1458, 1847
AIIN format
  COMPLETE_INIT function 1464
  START_INIT function 1464
AIIQ format
  END_BROWSE function 1466
  GET_NEXT function 1466
  INQUIRE_TERM_MODEL function 1465
  LOCATE_TERM_MODEL function 1464
  START_BROWSE function 1465
  UNLOCK_TERM_MODEL function 1464
AIRDELAY 1897
AITM format
  ADD_REPL_TERM_MODEL 1466
  DELETE_TERM_MODEL 1467
AITM manager 1463
ALLOCATE function, TFAL gate 309
ALLOCATE processing in application-owning region 1851

© Copyright IBM Corp. 1997, 2017

ALLOCATE processing in
terminal-owning region 1856
ALLOCATE_SEND function, ISIS
gate 563
ALLOCATE_SET_STORAGE function,
TSQR gate 1207
ALLOCATE_TRANSACTION_STG
function, SMAR gate 1067
allocation of TCTTE, function
shipping 1719
allocation program
undelivered messages clean up
program (TPQ) 1490
AMDUSREF 2019
AOR (application-owning region) 1458,
1847
ALLOCATE processing in 1851
APPC command processing in 1853
ATTACH processing in 1849
DETACH processing in 1851
FREE processing in 1853
LU6.2 command processing in 1853
AP (Application Manager Domain)
domain 10
AP (application) domain 10
AP 00E1 trace point ID 1583
AP A500 trace point ID 1583
AP A501 trace point ID 1583
AP E110 trace point ID 1583
AP E111 trace point ID 1583
APAC gate
REPORT_CONDITION function 22
APAP gate
TRANSFER_SIT function 23
APCR gate
ESTIMATE_ALL function 23
ESTIMATE_CHANGED function 24
EXPORT_ALL function 24
EXPORT_CHANGED function 25
IMPORT_ALL function 26
IMPORT_CHANGED function 27
APEX gate
INVOKE_USER_EXIT function 28
APID gate
PROFILE function 28
QUERY_NETNAME function 29
APIQ gate
INQ_APPLICATION_DATA
function 29
INQ_SIT_PARM function 30
APJC gate
WRITE_JOURNAL_DATA
function 30
APLI gate
ESTABLISH_LANGUAGE
function 31
PIPI_CALL_SUB function 35
PIPI_INIT_SUB_DP function 36
PIPI_TERM function 37
START_PROGRAM function 33
APLX gate
NOTIFY_REFRESH function 37
APPC
command processing in
application-owning region 1853
command processing in
terminal-owning region 1857
APPC *(continued)*
daisy chaining 1854
transaction routing 1865
z/OS Communications Server 1885
APPC (advanced program-to-program
communication) 1458, 1847
APPC autoinstall
call of builders 1517
APPC connections, autoinstall 1453
APPC control blocks 1848
APPC devices, autoinstall disconnection
flow 1458
APPC devices, autoinstall logon
flow 1454
APPC devices, LU6.2
transaction routing for 1865
APPEND function, RMRE gate 973
application (AP) domain 10
Application Manager Domain (AP)
domain 10
application programming commands,
FEPI
logic flow 1697
application programming functions with
function shipping 1707
application programs
mapping control program
(MCP) 1480
application services
basic mapping support (BMS) 1468
built-in functions 1519
command interpreter 1532
data interchange program (DIP) 1546
SAA Communications interface 1764
SAA Resource Recovery
interface 1764
temporary-storage browse
transaction 1583
application-owning region (AOR) 1458
APRA gate
RELAY_TERMINAL_REQUEST
function 38
REMOTE_ATTACH function 38
REMOTE_DETACH function 38
APRD gate
END_ATOMS function 39
INITIALISE function 39
PRE_INITIALISE function 40
APRL gate
CREATE_BUNDLE_BY_ID_VER
function 40
CREATE_RESOURCE function 41
DISCARD_BUNDLE_BY_ID_VER
function 42
SET_BUNDLE_BY_ID_VER
function 43
APRR gate
IPIC_ROUTE_TRANSACTION
function 44
APRS gate
ACQUIRE_SURROGATEfunction 44
RELEASE_SURROGATE function 44
APRT gate
ROUTE_TRANSACTION function 44
APRX gate
FLATTEN_REQUEST function 45
FLATTEN_RESPONSE function 46
APRX gate *(continued)*
UNFLATTEN_REQUEST function 46
UNFLATTEN_RESPONSE
function 46
APTC gate
CANCEL function 47
CLOSE function 47
EXTRACT_PROCESS function 47
LISTEN function 48
OPEN function 48
RECEIVE function 49
SEND function 49
SET_SESSION function 50
APTD gate
DELETE_TRANSIENT_DATA
function 50
INITIALISE_TRANSIENT_DATA
function 51
READ_TRANSIENT_DATA
function 52
RESET_TRIGGER_LEVEL
function 53
WRITE_TRANSIENT_DATA
function 53
APUE gate
SET_EXIT_STATUS function 334
APXM gate
BIND_XM_CLIENT function 55
INIT_XM_CLIENT function 55
RELEASE_XM_CLIENT function 55
RMI_START_OF_TASK function 55
AS domain
modules 10
Asynchronous services Domain (AS)
domain 8
ATI (automatic transaction
initiation) 1733, 1814, 1867
atom (resource definition) 1497
ATTACH function, BRAT gate 55
ATTACH function, DSAT gate 457
ATTACH function, XMAT gate 1336,
1398
ATTACH processing in
application-owning region 1849
ATTACH processing in terminal-owning
region 1855
audit trail 1510
autoinstall
APPC call of builders 1517
APPC connections 1453
diagnosing problems 1460
logoff call of builders 1517
logon call of builders 1517
rejection of BIND parameters 1460
rejection of logon request 1460
terminals 1453
autoinstall disconnection flow,
LU-initiated 1456
autoinstall logon flow
APPC devices 1454
autoinstall of a generic resource
connection 1455
autoinstall of consoles install flow 1455
autoinstall terminal model
manager 1763
autoinstall terminal model manager
(AITM) 1463

AUTOINSTALL_IPCONN function, ISIC gate 552
autoinstalled consoles deletion
  autoinstalled consoles deletion 1458
automatic journaling 1591
automatic logging 1591
automatic transaction initiation (ATI) 1733, 1814, 1867
AVAIL function, RMRE gate 975

B

BA (Business application manager domain) domain 335
BAAC gate
  ACQUIRE_ACTIVITY function 335
  ADD_ACTIVITY function 335
  ADD_REATTACH_ACQUIRED function 336
  ADD_TIMER_REQUEST function 336
  CANCEL_ACTIVITY function 336
  CHECK_ACTIVITY function 337
  DELETE_ACTIVITY function 338
  LINK_ACTIVITY function 338
  RESET_ACTIVITY function 339
  RESUME_ACTIVITY function 339
  RETURN_END_ACTIVITY function 340
  RUN_ACTIVITY function 340
  SUSPEND_ACTIVITY function 341
BABR gate
  COMMIT_BROWSE function 341
  ENDBR_ACTIVITY function 341
  ENDBR_CONTAINER function 342
  ENDBR_PROCESS function 342
  GETNEXT_ACTIVITY function 342
  GETNEXT_CONTAINER function 343
  GETNEXT_PROCESS function 343
  INQUIRE_ACTIVATION function 344
  INQUIRE_ACTIVITY function 344
  INQUIRE_CONTAINER function 346
  INQUIRE_PROCESS function 347
  STARTBR_ACTIVITY function 347
  STARTBR_CONTAINER function 348
  STARTBR_PROCESS function 348
BACKOUT function, FCCU gate 131
BACKOUT function, FCDU gate 147
backout logging 1591
BACKOUT_REPLICATION function, FCLJ gate 227
BACKOUT_UOW function, RMUW gate 980
backup while open (BWO) 1614, 1617, 1633, 1644
BACR gate
  COPY_CONTAINER function 349
  DELETE_CONTAINER function 350
  GET_CONTAINER_INTO function 350
  GET_CONTAINER_LENGTH function 351
  GET_CONTAINER_SET function 352
  MOVE_CONTAINER function 352
  PUT_CONTAINER function 353
BAPR gate
  ACQUIRE_PROCESS function 354
  ADD_PROCESS function 354
  CANCEL_PROCESS function 355
  CHECK_PROCESS function 355
  LINK_PROCESS function 356
  RESET_PROCESS function 357
  RESUME_PROCESS function 357
  RUN_PROCESS function 358
  SUSPEND_PROCESS function 358
basic direct access method (BDAM) 1588
BATT gate
  ADD_REPLACE_PROCESSTYPE function 359
  COMMIT_PROCESSTYPE_TABLE function 360
  DISCARD_PROCESSTYPE function 360
  END_BROWSE_PROCESSTYPE function 360
  GET_NEXT_PROCESSTYPE function 360
  INQUIRE_PROCESSTYPE function 361
  SET_PROCESSTYPE function 362
  START_BROWSE_PROCESSTYPE function 362
BAXM gate
  BIND_ACTIVITY_REQUEST function 363
  INIT_ACTIVITY_REQUEST function 363
BDAM
  ENDBR request processing 1600
  READ request processing 1594
  READNEXT request processing 1600
  RESETBR request processing 1599
  REWRITE request processing 1597
  STARTBR request processing 1599
  UNLOCK request processing 1598
  WRITE request processing 1595
BDAM (basic direct access method) 1588
BDAM request processor, file control (DFHFCBD) 1615
BEGIN_TRAN function, OTTR gate 778
BIND function, SOCK gate 1107
BIND_ACTIVITY_REQUEST function, BAXM gate 363
BIND_CHANNEL function, PGCH gate 787
BIND_FACILITY function, TFBF gate 326
BIND_LDAP function, DDAP gate 374
BIND_RECEIVER function, ISIS gate 564
BIND_SECONDARY_FACILITY function, TDXM gate 309
BIND_UOW_TO_TXN function, RMUW gate 981
BIND_XM_CLIENT function, APXM gate 55
BIND_XM_CLIENT function, DPXM gate 453
BIND_XM_CLIENT function, MPXM gate 750
BMS (basic mapping support) 1468
  3270 mapping (M32) 1483
BMS (basic mapping support) *(continued)*
  control blocks, illustrated 1472
  data stream build (DSB) 1477
  full version, modules used 1475
  LU1 printer with extended attributes mapping program (ML1) 1482
  mapping control program (MCP) 1479
  message switching 1749
  minimum version, modules used 1475
  modules 1474
  modules and routines, organization 1475
  non-3270 input mapping (IIP) 1478
  page and text build (PBP) 1484
  partition handling program (PHP) 1486
  route list resolution program (RLR) 1487
  standard version, modules used 1475
  terminal page processor (TPP) 1488
  terminal page retrieval program (TPR) 1491
  terminal page scheduling program (TPS) 1492
  undelivered messages clean up program (TPQ) 1490
BPS (builder parameter set) 1494
BRAT gate
  ATTACH function 55
BREAK_PARTNERSHIP function, PTTW gate 917
BRIQ gate
  INQUIRE_CONTEXT function 56
BROWSE function, FCRF gate 264
browse token 7
browse token, table manager 1797
BROWSE_ALL_GET_NEXT function, LGBA gate 646
BROWSE_CHAINS_GET_NEXT function, LGCC gate 648
BSAM (basic sequential access method) 1814
  and testing facility 1817
BUILD_CONTENT_TYPE function, PIMM gate 879
BUILD_MIME_HEADERS function, PIMM gate 880
BUILD_MIME_MESSAGE function, PIMM gate 881
BUILD_MULTIPART_RELATED function, PIMM gate 881
BUILD_XOP function, PIMM gate 915
build/delete terminals 1516
builder parameter list 1516
builder parameter set (BPS) 1494
builders 1494
  description 1493
  purpose 1502
builders for 3277 remote terminal
  calling sequence 1514
built-in functions
  description 1519
  field edit 1519
  phonetic conversion 1519

Business application manager domain (BA) domain 335
BWO (backup while open) 1614, 1617, 1633, 1644
BWO_BITS_DISABLED function, FCAT gate 100
BWO_BITS_ENABLED function, FCAT gate 101

C

CALL macro
  DL/I interface 1560
CALLDLI macro
  DL/I interface 1560
calling sequence builders for 3277 remote terminal 1514
CANCEL function, APTC gate 47
CANCEL function, SOCK gate 1108
CANCEL function, TISR gate 1180
CANCEL_ACTIVITY function, BAAC gate 336
CANCEL_AID function, TFAL gate 310
CANCEL_AIDS_FOR_CONNECTION function, TFAL gate 310
CANCEL_AIDS_FOR_TERMINAL function, TFAL gate 311
CANCEL_CLOSE_FILE function, FCFS gate 221
CANCEL_PROCESS function, BAPR gate 355
CANCEL_SPECIFIC_AID function, TFAL gate 312
CANCEL_TASK function, DSAT gate 458
CATA transaction 1495, 1499, 2046
catalog manager, file control (DFHFCAT) 1614
CATALOG_DSNB function, FCDN gate 136
CATALOG_PROGRAMS function, LDLD gate 626
CATD transaction 2046
CATR transaction 2048
CATS transaction 2048
CC (CICS catalog domain) domain 367
CCB (connection control block) 1726
CCCC gate
  ADD function 367
  DELETE function 368
  END_BROWSE function 368
  END_WRITE function 368
  GET function 369
  GET_NEXT function 369
  GET_UPDATE function 369
  PUT_REPLACE function 370
  START_BROWSE function 370
  START_WRITE function 370
  STARTUP_CLOSE function 371
  STARTUP_OPEN function 371
  TYPE_PURGE function 371
  WRITE function 371
  WRITE_NEXT function 372
CCE (console control element) 1823
CCNV gate
  CONVERT_ADS function 58
  CONVERT_DATA function 60
CCNV gate *(continued)*
  CREATE_CONVERSION_TOKEN function 62
  EXTRACT_ADS function 63
  FREE_CONVERSION_TOKEN function 65
  GET_CONVERSION_TOKEN function 66
  INITIALISE function 67
  INQUIRE_CONVERSION_SIZE function 69
  VERIFY_CGCSGID function 70
  VERIFY_CICS_CCSID function 72
  VERIFY_IANA_CCSID function 73
  VERIFY_IBM_CCSID function 74
CEBR transaction 1583
CECI transaction 1532
CECS transaction 1532
CEDA install 1517
CEDA transaction 1761
CEDB transaction 1761
CEDC transaction 1761
CEMT transaction 1746, 1768
CEOT transaction 1746
CEST transaction 1746
CFDT load program, file control (DFHFCDL) 1617
CFDT open/close program, file control (DFHFCDO) 1619
CFDT request processor, file control (DFHFCDR) 1619
CFDT resynchronization program, file control (DFHFCDY) 1620
CFDT RMC program, file control (DFHFCDW) 1620
CFDT UOW calls program, file control (DFHFCDU) 1620
CHAIN_BROWSE_GET_NEXT function, LGCB gate 647
CHANGE_MODE function, DSAT gate 459
CHANGE_PRIORITY function, DSAT gate 461
CHECK function, FCCA gate 103
CHECK_ACTIVITY function, BAAC gate 337
CHECK_CALLER_IN_ROLE function, XSEJ gate 1418
CHECK_CICS_COMMAND function, XSRC gate 1442
CHECK_CICS_RESOURCE function, XSRC gate 1445
CHECK_EJB_METHOD function, XSEJ gate 1419
CHECK_NON_CICS_RESOURCE function, XSRC gate 1446
CHECK_PREFIX function, TSBR gate 1195
CHECK_PROCESS function, BAPR gate 355
CHECK_STORAGE function, SMCK gate 1068
CHECK_SURROGATE_USER function, XSRC gate 1447
CHECK_TIMER function, EMEM gate 525
CHECK_TRANID_IN_USE function, TFAL gate 312
checkpoint and restart 1815
CIB (command input buffer) 1824
CICS business logic interface 1528
CICS catalog domain (CC) domain 367
CICS Web support 1528
CICS_RESYNC function, ISRE gate 576
CICS-DB2
  task-related user exit 1522
CICS-DB2 Attachment facility 1520
CICS-DB2 Attachment Facility 1526
CICS-DB2 DB2ENTRY block (D2ENT) 1526
CICS-DB2 DB2TRAN block (D2TRN) 1526
CICS-DB2 Global block (D2GLB) 1526
CICS-DB2 global work area (D2GWA) 1526
CICS-DB2 life of task block (D2LOT) 1527
CICS-DB2 packageset block (D2PKGSET) 1527
CICS-DB2 static storage (D2SS) 1526
CICS-DB2 subtask block (D2CSB) 1526
CICS-DB2 support 1520
CICS-DBCTL interface 1545
class of service, LU6.2 1885
CLEAR_ENVIRONMENT function, FCFR gate 160
CLEAR_LABELS function, PGHM gate 819
CLEAR_MATCH function, DSAT gate 462
CLEAR_PENDING function, RMNM gate 970
close destination program, DFHZCLS 1456
CLOSE function, APTC gate 47
CLOSE function, FCCT gate 124
CLOSE function, LGGL gate 656
CLOSE function, SOCK gate 1109
CLOSE_ALL_EXTRA_TD_QUEUES function, TDOC gate 293
CLOSE_FILE function, FCFS gate 221
CLOSE_MVS_CIB_QUEUE function, CQCQ gate 75
CLOSE_SESSION function, WBCL gate 1262
CLOSE_SESSION function, WBSV gate 1293
CLOSE_SOCKET_IN_POOL function, SOPL gate 1150
CLOSE_TRANSIENT_DATA function, TDOC gate 293
CMPX transaction 1715
CMSG transaction 1747, 1749
cold start 1516
COLD_START_RLS function, FCCA gate 104
COLLECT_FILE_STATISTICS function, FCST gate 281
COLLECT_POOL_STATISTICS function, FCST gate 282
COLLECT_RESOURCE_STATS function, STST gate 1165, 1172

COLLECT_STATISTICS function, STST gate 1166, 1173
command input buffer (CIB) 1824
command-language translator 1759
COMMIT function, FCCU gate 132
COMMIT function, FCDU gate 148
COMMIT function, OTTR gate 778
COMMIT function, RMOT gate 972
commit process
  single-phase 1809
  two-phase 1808
COMMIT_BROWSE function, BABR gate 341
COMMIT_DSNREFS function, FCDN gate 137
COMMIT_ENQMODEL function, NQRN gate 767
COMMIT_EVENT function, EPEV gate 536
COMMIT_FILES function, FCMT gate 241
COMMIT_ONE_PHASE function, OTTR gate 779
COMMIT_POOLS function, FCRL gate 276
COMMIT_PROCESSTYPE_TABLE function, BATT gate 360
COMMIT_REPLICATION function, FCLJ gate 227
COMMIT_TDQDEFS function, TDTM gate 299
COMMIT_UOW function, RMUW gate 982
common programming interface (CPI) 1764
communication with remote system 1709
Communications interface, SAA 1764
Communications Server
  exits
    DFASY 1703
    LOGON 1703
    LOSTERM 1703
    NSEXIT 1703
    SCIP 1703
    TPEND 1703
COMPLETE_INIT function, AIIN format 1464
COMPLETE_INIT function, CPIN format 1767
COMPLETE_JVMSERVER function, SJJS gate 1047
COMPLETE_PIPELINE function, PIPL gate 890
COMPLETE_QUIESCE function, FCQI gate 259
components of CICS
  organization 1
concurrency control 1589
concurrent tasks 1591
CONNECT function, LGLB gate 666
CONNECT function, LGST gate 674
CONNECT function, SOCK gate 1110
CONNECT_DSNB function, FCDN gate 137
connection control block (CCB) 1726
console control element (CCE) 1823
console message handling 1774
contention update model 1590
control blocks
  for BMS, illustrated 1472
  for file control, illustrated 1601
  for interregion communication (IRC), illustrated 1723
  for relay transaction, illustrated 1863
  for subsystem interface, illustrated 1775
  for table manager, illustrated 1799
  for terminal control, illustrated 1831
  for user exit interface, illustrated 1875
conversation
  session recovery 1730
CONVERSE function, ISIS gate 564
CONVERSE function, MEM4 gate 696
CONVERSE function, MEME gate 688
CONVERSE function, PITG gate 905
CONVERT_ADS function, CCNV gate 58
CONVERT_CID_TO_CONTENT_ID function, PIMM gate 882
CONVERT_CONTENT_ID_TO_CID function, PIMM gate 883
CONVERT_DATA function, CCNV gate 60
CONVERT_DATA_TO_JSON function, PIJG gate 877
CONVERT_NAME function, LDLD gate 626
CONVERT_TIME function, TIMF gate 1175
CONVERT_TO_DECIMAL_TIME function, KETI gate 606
CONVERT_TO_STCK_FORMAT function, KETI gate 607
COPY_CHANNEL function, PGCH gate 787
COPY_CONTAINER function, BACR gate 349
COPY_CONTAINER function, PGCR gate 795
COPY_CONTAINER_POOL function, PGCP gate 794
coupling facility data table 1590
coupling facility data tables server 1592
CPI (common programming interface) 1764
CPIN format
  COMPLETE_INIT function 1767
  START_INIT function 1767
CPMI transaction 1708
CPSP format
  SYNCPOINT_REQUEST 1767
CQCQ gate
  CLOSE_MVS_CIB_QUEUE function 75
  DEFER_CIB function 76
  GET_CIB function 76
  GET_PROCESSED_CIB function 77
  INITIALIZE function 77
  MERGE_CIB_QUEUES function 77
  PUT_CIB function 78
  PUT_PROCESSED_CIB function 78
  TRACE_PUT_CQ function 78
CRB (cross-region block) 1723
CREATE function, RLCB gate 942, 946
CREATE function, RZSO gate 1025
CREATE function, SOCK gate 1112
CREATE_ABEND_RECORD function, ABAB gate 12
CREATE_BUNDLE_BY_ID_VER function, APRL gate 40
CREATE_CHAIN_TOKEN function, LGCC gate 649
CREATE_CHANNEL function, PGCH gate 787
CREATE_CONTAINER_POOL function, PGCP gate 794
CREATE_CONTEXT function, PIAT gate 869
CREATE_CONTEXT_RESP function, PIAT gate 869
CREATE_CONVERSION_TOKEN function, CCNV gate 62
CREATE_DIRECTORY function, DDDI gate 382
CREATE_DOCUMENT function, DHDH gate 384
CREATE_ENQUEUE_POOL function, NQNQ gate 759
CREATE_ICE_USER_EXTENSION, ICUS gate 285
CREATE_JVMSERVER function, SJJS gate 1046
CREATE_LE_ENCLAVE function, LEPT gate 288
CREATE_NETWORK_UOWID function, RMUW gate 982
CREATE_NON_TERMINAL_MSG function, PIAT gate 870
CREATE_PARTNERSHIP function, PTTW gate 919
CREATE_PASSTICKET function, XSPW gate 1438
CREATE_POOL function, PTTW gate 919
CREATE_PTHREAD function, LEPT gate 288
CREATE_REGISTER_REQUEST function, PIAT gate 870
CREATE_REGISTER_RESP function, PIAT gate 871
CREATE_RESOURCE function, APRL gate 41
CREATE_SOCKET_POOL function, SOPL gate 1148
CREATE_TASK function, KEDS gate 593
CREATE_TCB function, KEDS gate 594
CREATE_TERMINAL_MSG function, PIAT gate 871
CREATE_UOW function, RMUW gate 983
CREATE_USER_EXTENSION, ICUS gate 331
CREATE_WEBSERVICE function, PIWR gate 907
create, EXEC CICS 1517
CREATED function, RLRO gate 936
creation/deletion state machine 1505
CROSS_SYSTEM_DUMP_AVAIL function, DUSR gate 509

cross-region block (CRB) 1723
cross-system coupling facility (XCF)
  used for interregion communication 1723
CSD utility program (DFHCSDUP) 1533
  commands 1533
CSFE transaction 1588
CSGM transaction 1722
CSM1 transaction 1708
CSM2 transaction 1708
CSM3 transaction 1708
CSM5 transaction 1708
CSMI transaction 1708
CSNC transaction 1729
  delay-queue 1730
  quiesce of interregion facility 1730
  suspension 1730
  termination 1730
CSNE transaction 1750
CSPG transaction 1491
CSPQ transaction 1481, 1490
CSPS transaction 1481
CSZI transaction 1697
CVMI transaction 1708
CXRT transaction 1856

D

D2CSB (CICS-DB2 subtask block) 1526
D2ENT (CICS-DB2 DB2ENTRY block) 1526
D2GLB (CICS-DB2 global block) 1526
D2GWA (CICS-DB2 global work area) 1526
D2LOT (CICS-DB2 life of task block) 1527
D2PKGSET (CICS-DB2 packageset block) 1527
D2SS (CICS-DB2 static storage) 1526
D2TRN (CICS-DB2 DB2TRAN block) 1526
daisy chaining
  APPC 1854
  LU6.2 1854
data control block (DCB) 1603
data event control block (DECB) 1819
data for function shipping, formatting 1711
data format
  transaction-routed 1861
data interchange block (DIB) 1547
data services
  transient 1868
data set name block (DSNB) 1604
data streams for transaction routing 1860
data table request processor, file control (DFHFCDTS) 1619
data tables, processing using 1592
database resource adapter (DRA) 1535
database support 1520, 1535, 1559, 1760
DATASET_COPY function, FCLJ gate 228
DB2 1520
DBCTL (database control) 1535
  PSB scheduling 1544
  PSB termination 1544
DBCTL (database control) *(continued)*
  system definition 1544
DBCTL call processor 1540
DBCTL global block (DGB) 1545
DBCTL scheduling block (DSB) 1545
DBCTL user-replaceable program 1540
DCB (data control block) 1603
DD (directory manager) domain 374
DDAP gate
  BIND_LDAP function 374
  END_BROWSE_RESULTS function 375
  FLUSH_LDAP_CACHE function 375
  FREE_SEARCH_RESULTS function 375
  GET_ATTRIBUTE_VALUE function 376
  GET_NEXT_ATTRIBUTE function 377
  GET_NEXT_ENTRY function 377
  SEARCH_LDAP function 378
  START_BROWSE_RESULTS function 379
  UNBIND_LDAP function 379
DDBR gate
  END_BROWSE function 380
  GET_NEXT_ENTRY function 380
  START_BROWSE function 381
DDDI gate
  ADD_ENTRY function 381
  CREATE_DIRECTORY function 382
  DELETE_DIRECTORY function 382
  DELETE_ENTRY function 382
  REPLACE_DATA function 383
DDLO gate
  LOCATE function 383
DEACTIVATE function, NQNQ gate 762
DEACTIVATE_TRAP function, TRSR gate 1186
DEBKEY option
  READ request processing 1595
  READNEXT request processing 1600
  RESETBR request 1599
  STARTBR request 1599
deblocking
  DEBKEY option 1595
  DEBREC option 1595
  READ request processing 1595
  RESETBR request 1599
  STARTBR request 1599
deblocking for BDAM data sets 1589
DEBREC option
  READ request processing 1595
  READNEXT request processing 1600
  RESETBR request 1599
  STARTBR request 1599
Debugging profile domain (DP) 419
DECB (data event control block) 1819
DECODE_ODR function, MNOD gate 729
DECREMENT_USE_COUNT function, PIWR gate 908
DEFER_CIB function, CQCQ gate 76
deferred work element (DWE) 1783
DEFINE_ATOMIC_EVENT function, EMEM gate 526
DEFINE_COMPOSITE_EVENT function, EMEM gate 526
DEFINE_PROGRAM function, LDLD gate 627
DEFINE_PROGRAM function, PGDD gate 812
DEFINE_TIMER function, EMEM gate 527
DEL_TRANSACTION_SECURITY function, XSXM gate 1449
DELETE function, CCCC gate 368
DELETE function, DMEN gate 411
DELETE function, FCCR gate 114
DELETE function, FCCT gate 125
DELETE function, FCFR gate 161
DELETE function, FCRF gate 266
DELETE function, TSQR gate 1207
DELETE function, TSSH gate 1214
delete lock 1599
DELETE_ACTIVITY function, BAAC gate 338
DELETE_ALL function, LGCC gate 649
DELETE_ALL_OPEN_TCBS function, DSIT gate 474
DELETE_ATOMSERVICE function, W2AT gate 1326
DELETE_ATTACHMENTS function, PIMM gate 883
DELETE_BEAN_SECURITY function, XSEJ gate 1420
DELETE_BOOKMARK function, DHDH gate 386
DELETE_CERTIFICATE_DATA function, SOIS gate 1130
DELETE_CHANNEL function, PGCH gate 788
DELETE_CONTAINER function, BACR gate 350
DELETE_CONTAINER function, PGCR gate 796
DELETE_CONTAINER_POOL function, PGCP gate 794
DELETE_DATA function, DHDH gate 386
DELETE_DEBUG_PROFILE function, DPFM gate 420
DELETE_DIRECTORY function, DDDI gate 382
DELETE_DOCTEMPLATE function, DHTM gate 401
DELETE_DOCUMENT function, DHDH gate 387
DELETE_DSNB function, FCDN gate 138
DELETE_ENTRY function, DDDI gate 382
DELETE_EVENT function, EMEM gate 528
DELETE_FILE function, FCMT gate 241
DELETE_GATE function, KEDD gate 584
DELETE_HFS_FILE function, DHFS gate 392
DELETE_HISTORY function, LGCC gate 650
DELETE_ICE_USER_EXTENSION, ICUS gate 286

DELETE_LINK function, RMLN gate 954
DELETE_LOCK function, LMLM gate 683
DELETE_MULTIPLE function, FCCR gate 116
DELETE_OPEN_TCB function, DSIT gate 475
DELETE_OWNED_CHANNELS function, PGCH gate 789
DELETE_PENDING_REQUEST function, SHPR gate 1035
DELETE_PROGRAM function, LDLD gate 629
DELETE_PROGRAM function, PGDD gate 817
DELETE_SOCKET_POOL function, SOPL gate 1149
DELETE_SUBPOOL function, S2AD gate 1090
DELETE_SUBPOOL function, SMAD gate 1065
DELETE_SUBSPACE_TCBS function, DSAT gate 462
DELETE_SUSPEND function, DSSR gate 481
DELETE_SYSTEM_DUMPCODE function, DUDT gate 495
DELETE_TCB function, DSIT gate 475
DELETE_TCLASS function, XMCL gate 1349
DELETE_TCPIPSERVICE function, SOAD gate 1105
DELETE_TERM_MODEL, AITM format 1467
DELETE_THREADED_TCB function, SJDS gate 1046
DELETE_TIMER function, EMEM gate 528
DELETE_TRAN_DUMPCODE function, DUDT gate 495
DELETE_TRANDEF function, XMDD gate 1355
DELETE_TRANSACTION_USER function, USXM gate 1246
DELETE_TRANSIENT_DATA function, APTD gate 50
DELETE_TRUSTED_EXTENSION, SMK7 gate 1074
DELETE_TSMODEL function, TSAD gate 1194
DELETE_URIMAP function, WBUR gate 1297
DELETE_USER function, USAD gate 1233
DELETE_USER_EXTENSION, ICUS gate 332
DELETE_USER_SECURITY function, XSAD gate 1408
DELETE_USER_TOKEN, ICUS gate 286
deletion of autoinstalled APPC devices 1457
deletion of autoinstalled consoles 1458
DELIVER_BACKOUT_DATA function, RMRO gate 997
DELIVER_FORGET function, RMDE gate 1001
DELIVER_IC_RECOVERY_DATA function, TSRM gate 1226
DELIVER_RECOVERY function, RMDE gate 1001
DEQUEUE function, NQED gate 752
DEQUEUE_TASK function, NQNQ gate 762
DEREGISTER function, DUFT gate 507
DEREGISTER function, KEAR gate 582
DEREGISTER function, RLRO gate 936
DEREGISTER function, SORD gate 1151
DEREGISTER_INTEREST function, RSSR gate 1014
DEREGISTER_POLICY function, MPMO gate 749
DEREGISTER_TCLASS_USAGE function, XMCL gate 1350
DESTROY_PARTNERSHIP function, PTTW gate 920
DESTROY_POOL function, PTTW gate 921
DETACH processing in application-owning region 1851
DETACH_CHANNEL function, PGCH gate 789
DETACH_TERMINATED_OWN_TCBS function, KEDS gate 595
device independence 1471
DFH$ISAI 581
DFH£STXA 1770, 1771
DFH0ISAI 581
DFH0STXC 1770, 1771
DFH0STXR 1770, 1771
DFH99BC 1567
DFH99CC 1567
DFH99DY 1567
DFH99FP 1567
DFH99GI 1567
DFH99KC 1567
DFH99KH 1567
DFH99KO 1567
DFH99KR 1567
DFH99LK 1567
DFH99M 1565
DFH99ML 1567
DFH99MM 1567
DFH99MP 1567
DFH99MT 1567
DFH99RP 1567
DFH99T 1567
DFH99TK 1567
DFH99TX 1567
DFH99VH 1567
DFHACP 1842, 1844, 1913
DFHAICBP 1913
DFHAIDUF 1467, 1785
DFHAIIN1 1467
DFHAIIN2 1467
DFHAIIQ 1468
DFHAIRP 1468
DFHAITM 1468
DFHALP 1914
DFHAMP 1761, 1762, 1830, 1914
DFHAMPIL 1830
DFHAMTP 1495, 1827, 1830
DFHAMXM 1763
DFHAPAC 1844
DFHAPEX 1875, 1876
DFHAPJC 1914
DFHAPRC 1784
DFHAPRDR 1496, 1498, 1830
DFHAPRT 1846, 1856, 1865
DFHAPRX 1915
DFHAPSIP 1915
DFHAPST 1916
DFHAPTD 1916
DFHAPTI 1917
DFHAPTIM 1734, 1917
DFHAPTIX 1734, 1917
DFHAPTR0 1785, 1841
DFHAPTR2 1785, 1841
DFHAPTR4 1785, 1841
DFHAPTR5 1785, 1841
DFHAPTR6 1785, 1841
DFHAPTR7 1785, 1841
DFHAPTR8 1785, 1841
DFHAPTR9 1785, 1841
DFHAPTRA 1785, 1841
DFHAPTRB 1785, 1841
DFHAPTRC 1785, 1841
DFHAPTRD 1785, 1841
DFHAPTRE 1785, 1841
DFHAPTRF 1768, 1785, 1841
DFHAPTRG 1785, 1841
DFHAPTRI 1785, 1841
DFHAPTRJ 1785, 1841
DFHAPTRL 1785, 1841
DFHAPTRN 1468, 1785, 1841
DFHAPTRO 1785, 1841
DFHAPTRP 1785, 1841
DFHAPTRR 1785, 1841
DFHAPTRS 1785, 1841
DFHAPTRV 1785, 1841
DFHAPTRW 1785, 1841
DFHAPXME 1844
DFHASAS 10
DFHASDM 10
DFHASDUF 10
DFHASRM 10
DFHASTRI 10
DFHASV 1918
DFHASXM 10
DFHBAA10 365
DFHBAA11 365
DFHBAA12 365
DFHBAAC 365
DFHBAAC0 365
DFHBAAC1 365
DFHBAAC2 365
DFHBAAC3 365
DFHBAAC4 365
DFHBAAC5 365
DFHBAAC6 365
DFHBAAR1 365
DFHBAAR2 365
DFHBABR 365
DFHBABU1 365
DFHBACO1 365
DFHBACR 365
DFHBADM 366
DFHBADU1 366
DFHBADUF 366
DFHBALR2 366
DFHBALR3 366

DFHBALR4 366
DFHBALR5 366
DFHBALR6 366
DFHBALR7 366
DFHBALR8 366
DFHBALR9 366
DFHBAOFI 366
DFHBAPR 366
DFHBAPR0 366
DFHBAPT1 366
DFHBAPT2 366
DFHBAPT3 366
DFHBARUC 366
DFHBARUD 366
DFHBARUP 366
DFHBASP 366
DFHBATRI 366
DFHBATT 367
DFHBAUE 367
DFHBAVP1 367
DFHBAXM 367
DFHBMSCA 1492
DFHBMSX 1473
DFHBS* builder programs 1495, 1828
DFHBSIB3 1918
DFHBSIZ1 1918
DFHBSIZ3 1918
DFHBSM61 1919
DFHBSM62 1919
DFHBSMIR 1919
DFHBSMPP 1919
DFHBSS 1920
DFHBSSA 1920
DFHBSSF 1920
DFHBSSS 1920
DFHBSSZ 1921
DFHBSSZ6 1923
DFHBSSZB 1921
DFHBSSZG 1921
DFHBSSZI 1922
DFHBSSZL 1922
DFHBSSZM 1922
DFHBSSZP 1922
DFHBSSZR 1923
DFHBSSZS 1923
DFHBST 1923
DFHBSTB 1924
DFHBSTB3 1924
DFHBSTBL 1924
DFHBSTC 1924
DFHBSTD 1925
DFHBSTE 1925
DFHBSTH 1925
DFHBSTI 1925
DFHBSTM 1926
DFHBSTO 1926
DFHBSTP3 1926
DFHBSTS 1927
DFHBSTT 1927
DFHBSTZ 1927
DFHBSTZ1 1931
DFHBSTZ2 1931
DFHBSTZ3 1931
DFHBSTZA 1927
DFHBSTZB 1928
DFHBSTZC 1928
DFHBSTZE 1928
DFHBSTZH 1928
DFHBSTZL 1929
DFHBSTZO 1929
DFHBSTZP 1929
DFHBSTZR 1929
DFHBSTZS 1930
DFHBSTZV 1930
DFHBSTZZ 1930
DFHBSXGS 1931
DFHBSZZ 1932
DFHBSZZS 1932
DFHBSZZV 1932
DFHCAPB 1932
DFHCCCC 373
DFHCCDM 373
DFHCCDUF 373, 1785
DFHCCNV 1933
DFHCCTRI 373, 1785, 1842
DFHCCUTL 373
DFHCDCON 1841
DFHCLS3 1885, 1892
DFHCMAC 702
DFHCMP 1933
DFHCPARH 1768
DFHCPCxx 1768
DFHCPDUF 1768, 1785
DFHCPI 1768
DFHCPIN1 1768
DFHCPIN2 1768
DFHCPIR 1768
DFHCPLC 1768
DFHCPLRR 1768
DFHCPSRH 1768
DFHCPY 1933
DFHCRC 1731, 1934
DFHCRNP 1729, 1934
DFHCRQ 1934
DFHCRR 1731, 1934
DFHCRS 1935
DFHCRSP 1729, 1935
DFHCRT 1856, 1865, 1935
DFHCSA 1936
DFHCSDUF 1785
DFHCSDUP 1534, 1763, 1936
DFHCSSC 1936
DFHCSVC 1937
DFHCSVCU 1937
DFHCTRI 1459
DFHCUCAB 1938
DFHCUCB 1938
DFHCUCCB 1938
DFHCUCDB 1939
DFHCWTO 1939
DFHD2CC 1527
DFHD2CM0 1527
DFHD2CM1 1527
DFHD2CM2 1527
DFHD2CM3 1527
DFHD2CO 1527
DFHD2D2 1527
DFHD2EDF 1527
DFHD2EX1 1522, 1527
DFHD2EX2 1527
DFHD2EX3 1527
DFHD2IN1 1527
DFHD2IN2 1527
DFHD2INI 1527
DFHD2MSB 1527
DFHD2RP 1527
DFHD2ST 1527
DFHD2STP 1527
DFHD2STR 1527
DFHD2TM 1527
DFHDBAT 1545, 1939
DFHDBCON 1545, 1940
DFHDBCR 1940
DFHDBCT 1545, 1940
DFHDBCTX 1545, 1941
DFHDBDI 1545, 1941
DFHDBDSC 1545, 1941
DFHDBDUF 1785
DFHDBIE 1545
DFHDBIQ 1545, 1942
DFHDBME 1545, 1942
DFHDBMOX 1545, 1942
DFHDBNE 1545
DFHDBP 1784, 1942
DFHDBREX 1545, 1943
DFHDBSPX 1545, 1943
DFHDBSSX 1545, 1943
DFHDBSTX 1545, 1944
DFHDBTOX 1545, 1944
DFHDBUEX 1545, 1944
DFHDCP 1945
DFHDDDUF 1785
DFHDDTRI 1785, 1842
DFHDES 1945
DFHDHDH 408
DFHDHDM 408
DFHDHDUF 408
DFHDHPB 408
DFHDHPD 408
DFHDHPM 408
DFHDHPR 408
DFHDHPS 408
DFHDHPT 408
DFHDHPU 408
DFHDHPX 408
DFHDHRM 408
DFHDHSL 409
DFHDHTM 409
DFHDHTRI 409
DFHDHUE 409
DFHDIP 1546, 1945
DFHDLI 1545, 1560, 1562, 1946
DFHDLIAI 1946
DFHDLIDP 1545, 1562, 1946
DFHDLIRP 1562, 1760, 1946
DFHDLXDF 518
DFHDMDM 419
DFHDMDS 419
DFHDMDUF 419, 1785
DFHDMEN 419
DFHDMENF 419
DFHDMIQ 419
DFHDMP 1761, 1762, 1947
DFHDMPBA 1534
DFHDMSVC 419
DFHDMTRI 419, 1785, 1842
DFHDMWQ 419
DFHDRPG 1947
DFHDSAT 492
DFHDSB 1473, 1477
DFHDSB1$ 1948

DFHDSBA$ 1948
DFHDSBR 492
DFHDSDM 492
DFHDSDUF 1785
DFHDSIT 492
DFHDSKE 492
DFHDSSM 492
DFHDSSR 492
DFHDSST 492
DFHDSTRI 1785, 1842
DFHDSUE 492
DFHDU710 1841, 1948
DFHDUDM 518
DFHDUDT 518
DFHDUDU 518
DFHDUDUF 1785
DFHDUF 1785
DFHDUFUT 1785
DFHDUIO 518
DFHDUPH 518, 1564
DFHDUPM 518, 1565
DFHDUPMC 1565
DFHDUPP 518, 1564
DFHDUPPC 1565
DFHDUPR 518, 1564
DFHDUPS 518, 1564
DFHDUPSC 1565
DFHDUSR 518
DFHDUSU 518
DFHDUSVC 518
DFHDUTM 518
DFHDUTRI 1785, 1842
DFHDUXD 518
DFHDUXW 518
DFHDXACH 1948
DFHDXSTM 1949
DFHDYP 1949
DFHEAG 1949
DFHEAG0 1949
DFHEAI 1950
DFHEAI0 1950
DFHEAP1$ 1759, 1951
DFHEBF 1519, 1580, 1951
DFHEBU 1951
DFHECI 1952
DFHECID 1532, 1952
DFHECIP 1532, 1952
DFHECP1$ 1759, 1953
DFHECSP 1532
DFHEDAD 1762, 1953
DFHEDAP 1762, 1953
DFHEDC 1580, 1954
DFHEDCP 1582
DFHEDFBR 1584, 1954
DFHEDFD 1584, 1954
DFHEDFM 1584, 1955
DFHEDFP 1584, 1955
DFHEDFR 1584, 1955
DFHEDFX 1584, 1738, 1955
DFHEDI 1580, 1956
DFHEDP 1956
DFHEDP1$ 1759, 1956
DFHEEI 1578, 1957
DFHEEX 1957
DFHEFRM 1957
DFHEGL 1551, 1580, 1958
DFHEIBLK 1571
DFHEICDS 1572
DFHEIDTI 1581, 1958
DFHEIFC 1613, 1707, 1959
DFHEIG 1958
DFHEIIC 1958
DFHEIP 1549, 1570, 1738, 1794, 1959
DFHEIPA 1570, 1959
DFHEIPRT 1582
DFHEIPSE 1582
DFHEIPSH 1582
DFHEIQDN 1582
DFHEIQDS 1581
DFHEIQDU 1582
DFHEIQIR 1582
DFHEIQMS 1582
DFHEIQPF 1581
DFHEIQPN 1581
DFHEIQSA 1581
DFHEIQSC 1581, 1827
DFHEIQSJ 1582
DFHEIQSK 1582
DFHEIQSM 1581
DFHEIQSP 1581
DFHEIQSQ 1582
DFHEIQST 1581, 1827
DFHEIQSX 1581
DFHEIQSZ, FEPI EXEC stub 1697
DFHEIQTM 1581
DFHEIQTR 1582
DFHEIQVT 1582
DFHEISC 1579, 1960
DFHEISDS 1572
DFHEISP 1960
DFHEISR 1960
DFHEITAB 1532
DFHEITBS 1532
DFHEITMT 1746
DFHEITOT 1746
DFHEITST 1746
DFHEJC 1580, 1961
DFHEJDU 520
DFHEKC 1580, 1961
DFHELII 1961
DFHEMBA 532
DFHEMBR 532
DFHEMDM 533
DFHEMDUF 533
DFHEMEM 533
DFHEMS 1473, 1580, 1961
DFHEMTA 1962
DFHEMTD 1746, 1962
DFHEMTP 1746, 1962
DFHEMTRI 533
DFHEOP 1582
DFHEOTP 1746, 1962
DFHEPAS 543
DFHEPC 1549, 1756, 1963
  EXEC CICS LINK command 1579
DFHEPDM 543
DFHEPDS 543
DFHEPDUF 543
DFHEPEV 543
DFHEPI 1963
DFHEPIS 543
DFHEPP1$ 1759, 1963
DFHEPS 1581, 1794, 1964
DFHEPSS 543
DFHEPSY 543
DFHEPTRI 543
DFHEPXM 543
DFHERDUF 1785
DFHERM 1807, 1964
DFHERMRS 1782, 1810
DFHESE 1582
DFHESN 1582
DFHESP 1580
DFHESTP 1746, 1964
DFHESZ, FEPI EXEC stub 1697
DFHETC 1550, 1552, 1578, 1964
DFHETD 1579, 1871, 1965
DFHETL 1550, 1552, 1965
DFHETR 1580, 1965
DFHETRX 1581
DFHETS 1579, 1966
DFHEXI 1966
DFHFCAT 1614, 1966
DFHFCBD 1615, 1966
DFHFCCA 1616
DFHFCDL 1617
DFHFCDN 1617, 1967
DFHFCDO 1619
DFHFCDR 1619
DFHFCDTS 1619, 1967
DFHFCDTX 1620
DFHFCDU 1620
DFHFCDUF 1785
DFHFCDW 1620
DFHFCDY 1620
DFHFCES 1620
DFHFCFL 1620
DFHFCFR 1620, 1967
DFHFCFS 1622, 1968
DFHFCIN1 1625
DFHFCIN2 1626
DFHFCIR 1627
DFHFCL 1627, 1968
DFHFCLF 1629
DFHFCLJ 1629
DFHFCM 1968
DFHFCMT 1629, 1969
DFHFCN 1632, 1969
DFHFCNQ 1636
DFHFCOR 1637
DFHFCQI 1637
DFHFCQR 1637
DFHFCQS 1637
DFHFCQT 1637
DFHFCQU 1638
DFHFCRC 1638
DFHFCRD 1640
DFHFCRF 1641
DFHFCRL 1638, 1642, 1969
DFHFCRO 1643
DFHFCRP 1643, 1970
DFHFCRR 1645
DFHFCRS 1645
DFHFCRV 1645
DFHFCSD 1646, 1970
DFHFCST 1646, 1970
DFHFCU 1970
DFHFCVR 1591, 1648, 1971
DFHFCVS 1649, 1971
DFHFCXDF 518
DFHFDP 1971

DFHFEP 1588, 1972
DFHFRDUF 1785
DFHGMM 1722, 1972
DFHHPSVC 1972
DFHICDUF 1785
DFHICP 1734, 1973
DFHICRC 1734
DFHIEDM 546
DFHIEIE 546
DFHIIP 1474, 1478
DFHIIP1$ 1973
DFHIIPA$ 1973
DFHIPCSP 1785
DFHIPDUF 1785
DFHIR3762 message 1730
DFHIRP 1709, 1728, 1974
DFHIRW10 1974
DFHISAIP 581
DFHISAL 581
DFHISBU 581
DFHISCIP 581
DFHISCO 581
DFHISCOP 581
DFHISCU 581
DFHISDIP 581
DFHISDM 582
DFHISDUF 582
DFHISEM 582
DFHISFS 582, 1975
DFHISIC 582
DFHISIF 582
DFHISIS 582
DFHISJU 582
DFHISP 1549, 1707, 1974
DFHISPIP 581
DFHISRE 582
DFHISRE1 582
DFHISREX 582
DFHISRR 582
DFHISRRP 582
DFHISSR 582
DFHISTRI 582
DFHISUE 582
DFHISXF 582, 1975
DFHISXFT 582
DFHISXM 582
DFHISZA 582
DFHJCP 1975
DFHJUP 1976
DFHKCP 1976
DFHKCQ 1976
DFHKCRP 1977
DFHKCSC 1977
DFHKCSP 1977
DFHKEAR 611
DFHKEDCL 611
DFHKEDD 611
DFHKEDRT 611
DFHKEDS 611
DFHKEDUF 611, 1785
DFHKEEDA 611
DFHKEGD 611
DFHKEIN 611
DFHKELCL 611
DFHKELOC 611, 1785
DFHKELRT 611
DFHKERCD 611
DFHKERER 611
DFHKERET 611
DFHKERKE 611
DFHKERPC 611
DFHKERRI 611
DFHKERRQ 611
DFHKERRU 612
DFHKERRX 612
DFHKESCL 612
DFHKESFM 612
DFHKESGM 612
DFHKESIP 612
DFHKESRT 612
DFHKESTX 612
DFHKESVC 612
DFHKETA 612
DFHKETCB 612
DFHKETI 612
DFHKETIX 612
DFHKETRI 612, 1786, 1842
DFHKETXR 612
DFHKEXM 612
DFHL2BA 681
DFHL2BL1 681
DFHL2BL2 681
DFHL2BS1 681
DFHL2BS2 681
DFHL2BS3 681
DFHL2BS4 681
DFHL2CB 681
DFHL2CC 681
DFHL2CH1 681
DFHL2CH2 681
DFHL2CH3 681
DFHL2CH4 681
DFHL2CH5 681
DFHL2CHA 681
DFHL2CHE 682
DFHL2CHG 681
DFHL2CHH 681
DFHL2CHI 681
DFHL2CHL 681
DFHL2CHM 682
DFHL2CHN 681
DFHL2CHR 682
DFHL2CHS 682
DFHL2DM 681
DFHL2HS2 682
DFHL2HS3 682
DFHL2HS4 682
DFHL2HS5 682
DFHL2HS6 682
DFHL2HS7 682
DFHL2HS8 682
DFHL2HS9 682
DFHL2HSG 682
DFHL2HSJ 682
DFHL2LB 681
DFHL2MV 681
DFHL2OFI 682
DFHL2SL1 682
DFHL2SLE 682
DFHL2SLN 682
DFHL2SR 681
DFHL2SR1 682
DFHL2SR2 682
DFHL2SR3 682
DFHL2SR4 682
DFHL2SR5 682
DFHL2TR 681
DFHL2TRI 1842
DFHL2VPX 682
DFHL2WF 681
DFHLDDM 644
DFHLDDMI 644
DFHLDDUF 644, 1786
DFHLDLB 645
DFHLDLB2 645
DFHLDLB3 645
DFHLDLD 644
DFHLDLD1 644
DFHLDLD2 644
DFHLDLD3 645
DFHLDNT 645
DFHLDST 645
DFHLDSVC 645
DFHLDTRI 645, 1786, 1842
DFHLGDM 680
DFHLGDUF 680
DFHLGGL 680
DFHLGHB 680
DFHLGICV 681
DFHLGIGT 681
DFHLGILA 681
DFHLGIMS 681
DFHLGIPA 681
DFHLGIPI 681
DFHLGISM 681
DFHLGJN 681
DFHLGLD 681
DFHLGPA 681
DFHLGSC 681
DFHLGSSI 681
DFHLGST 681
DFHLGTRI 681, 1842
DFHLMDM 685
DFHLMDS 685
DFHLMDUF 685, 1786
DFHLMLM 685
DFHLMTRI 686, 1786, 1842
DFHLUC requests 1853
DFHLUP 1978
DFHM32 1474, 1483
DFHM321$ 1982
DFHM32A$ 1982
DFHMAPDS 1471
DFHMCAD 1471
DFHMCBDS 1471
DFHMCP 1474, 1479
DFHMCP1$ 1978
DFHMCPA$ 1978
DFHMCPE 1493
DFHMCPE$ 1978
DFHMCPIN 1493
DFHMCRDS 1471
DFHMCX 1474, 1978
DFHME64 702
DFHMEBM 702
DFHMEBU 702
DFHMEDM 702
DFHMEDUF 702, 1786
DFHMEFO 702
DFHMEIN 702
DFHMEME 702

DFHMERSx 1020
DFHMESR 702
DFHMETRI 702, 1786, 1842
DFHMEWS 702
DFHMEWT 702
DFHMGP 1746, 1747, 1979
DFHMGT 1747, 1979
DFHMIN 1493
DFHMIRS 1549, 1707, 1979
DFHML1 1482, 1980
DFHMLDM 716
DFHMLDUF 716
DFHMLPC 716
DFHMLTF 716
DFHMLTRI 716
DFHMLXT 716
DFHMNAC 739
DFHMNAD 739
DFHMNDM 739
DFHMNDUF 739, 1786
DFHMNMN 739
DFHMNNT 739
DFHMNOD 739
DFHMNSM 739
DFHMNSR 739
DFHMNST 739
DFHMNSU 740
DFHMNSVC 740
DFHMNTD 740
DFHMNTI 740
DFHMNTRI 740, 1786, 1842
DFHMNUE 740
DFHMNXM 740
DFHMPAC 752
DFHMPDD 752
DFHMPDM 752
DFHMPDUF 752
DFHMPIB 752
DFHMPMO 752
DFHMPPP 752
DFHMPRL 752
DFHMPTRI 752
DFHMPXM 752
DFHMRDUF 1786
DFHMROQP 1981
DFHMSP 1747, 1749, 1981
DFHMSRCA 1493
DFHMXP 1982
DFHNQDM 772
DFHNQDUF 772
DFHNQED 772
DFHNQEDI 772
DFHNQIB 772
DFHNQIE 772
DFHNQNQ 772
DFHNQRN 772
DFHNQST 772
DFHNQTRI 772, 1842
DFHNXDUF 1786
DFHOSPWA 1472
DFHOTCO 780
DFHOTDM 780
DFHOTDUF 780
DFHOTRM 780
DFHOTSU 780
DFHOTTR 780
DFHOTTRI 780
DFHP3270 1986
DFHPADM 784
DFHPADUF 784, 1786
DFHPAGP 784
DFHPAIO 784
DFHPASY 784
DFHPATRI 784, 1786, 1842
DFHPBP 1474, 1484
DFHPBP1$ 1982
DFHPBPA$ 1982
DFHPCP 1755
DFHPCXDF 518
DFHPD510 1784
DFHPD710 1983
DFHPDKW 1786
DFHPDX1 1786
DFHPEP 1758, 1983
DFHPGADS 1472
DFHPGAI 867
DFHPGAQ 867
DFHPGDD 867, 1763
DFHPGDM 867
DFHPGDUF 867, 1786
DFHPGEX 867
DFHPGHM 867
DFHPGIS 868
DFHPGLD 868
DFHPGLE 868
DFHPGLK 868
DFHPGLU 868
DFHPGPG 868
DFHPGRE 868
DFHPGRP 868
DFHPGST 868
DFHPGTRI 868, 1786, 1842
DFHPGUE 868
DFHPGXE 868
DFHPGXM 868
DFHPHN 1519
DFHPHP 1474, 1486, 1984
DFHPIA1 916
DFHPIAD 916
DFHPIAP 916
DFHPIAT 916
DFHPICA 916
DFHPICC 916
DFHPIDM 916
DFHPIDSH 916
DFHPIDUF 916
DFHPIII 916
DFHPIIT 916
DFHPIIW 916
DFHPILN 916
DFHPIMM 916
DFHPIPA 917
DFHPIPL 917
DFHPIPM 917
DFHPIRL 917
DFHPIRT 917
DFHPISF 917
DFHPISN 917
DFHPISN1 917
DFHPISN2 917
DFHPIST 917
DFHPITC 917
DFHPITH 917
DFHPITL 917
DFHPITP 917
DFHPITQ 917
DFHPITQ1 917
DFHPITRI 917
DFHPITS 917
DFHPIWR 917
DFHPIWT 917
DFHPIXI 917
DFHPIXO 917
DFHPIXS 917
DFHPL1OI 1984
DFHPRDUF 1786
DFHPRK 1984
DFHPSDDS 1472
DFHPSP 1794, 1985
DFHPSPCK 1794
DFHPSPDW 1794, 1985
DFHPSPSS 1794, 1985
DFHPSPST 1794, 1986
DFHPSSVC 1794, 1986
DFHPTDM 929
DFHPTDUF 1786
DFHPTTW 930
DFHPUP 1762, 1986
DFHQRY 1827, 1831, 1987
DFHRCEX 1987
DFHREST 1845, 1988
DFHRKB 1987
DFHRLCB 947
DFHRLDM 947
DFHRLDUF 947
DFHRLMF 947
DFHRLPK 947
DFHRLPM 947
DFHRLR 1474, 1487
DFHRLR1$ 1988
DFHRLRA$ 1988
DFHRLRG 947
DFHRLRO 947
DFHRLRP 947
DFHRLRS 947
DFHRLSC 947
DFHRLST 948
DFHRLTRI 948
DFHRLVP1 948
DFHRLXM 948
DFHRMCAL 1802
DFHRMCD 1009
DFHRMCD1 1009
DFHRMCD2 1009
DFHRMCI2 1009
DFHRMCI3 1009
DFHRMCI4 1010
DFHRMDM 1010
DFHRMDU0 1010
DFHRMDU2 1010
DFHRMDU3 1010
DFHRMDU4 1010
DFHRMDUF 1786
DFHRML1D 1010
DFHRMLK1 1010
DFHRMLK2 1010
DFHRMLK3 1010
DFHRMLK4 1010
DFHRMLK5 1010
DFHRMLKQ 1010
DFHRMLN 1010

DFHRMLSD 1010
DFHRMLSF 1010
DFHRMLSO 1010
DFHRMLSP 1010
DFHRMLSS 1010
DFHRMLSU 1010
DFHRMNM 1010
DFHRMNM1 1010
DFHRMNS1 1010
DFHRMNS2 1010
DFHRMOFI 1010
DFHRMR1D 1010
DFHRMR1E 1010
DFHRMR1K 1010
DFHRMR1S 1010
DFHRMRO 1010
DFHRMRO1 1010
DFHRMRO2 1010
DFHRMRO3 1010
DFHRMRO4 1010
DFHRMROO 1010
DFHRMROS 1010
DFHRMROU 1010
DFHRMROV 1010
DFHRMSL 1010
DFHRMSL1 1011
DFHRMSL2 1011
DFHRMSL3 1011
DFHRMSL4 1011
DFHRMSL5 1011
DFHRMSL6 1011
DFHRMSL7 1011
DFHRMSLF 1011
DFHRMSLJ 1011
DFHRMSLL 1011
DFHRMSLO 1011
DFHRMSLV 1011
DFHRMSLW 1011
DFHRMST 1011
DFHRMST1 1011
DFHRMSY 1782, 1810, 1988
DFHRMTRI 1011, 1842
DFHRMU1C 1011
DFHRMU1D 1011
DFHRMU1E 1011
DFHRMU1F 1011
DFHRMU1J 1012
DFHRMU1K 1012
DFHRMU1L 1012
DFHRMU1N 1012
DFHRMU1Q 1012
DFHRMU1R 1012
DFHRMU1S 1012
DFHRMU1U 1012
DFHRMU1V 1012
DFHRMU1W 1012
DFHRMUC 1011
DFHRMUO 1011
DFHRMUTL 1011
DFHRMUW 1011
DFHRMUW0 1011
DFHRMUW1 1011
DFHRMUW2 1011
DFHRMUW3 1011
DFHRMUWB 1011
DFHRMUWE 1011
DFHRMUWF 1011
DFHRMUWH 1011
DFHRMUWJ 1011
DFHRMUWL 1011
DFHRMUWN 1011
DFHRMUWP 1011
DFHRMUWQ 1011
DFHRMUWS 1011
DFHRMUWU 1011
DFHRMUWV 1011
DFHRMUWW 1011
DFHRMVP1 1012
DFHRMXN2 1012
DFHRMXN3 1012
DFHRMXN4 1012
DFHRMXN5 1012
DFHRMXNE 1012
DFHRSDM 1020
DFHRSDU 1020
DFHRSDUF 1020
DFHRSFD 1020
DFHRSSM 1020
DFHRSSR 1020
DFHRSXM 1020
DFHRSXRI 1020
DFHRTC 1989
DFHRTE 1989
DFHRTSU 1865
DFHRXDM 1024
DFHRXDUF 1024
DFHRXSVC 1024
DFHRXTRI 1024
DFHRXUW 1024
DFHRXXRG 1024
DFHRXXRM 1024
DFHRZDUF 1033
DFHRZIX 1033
DFHRZJN 1033
DFHRZLN 1033
DFHRZNR2 1033
DFHRZOFI 1033
DFHRZRG2 1033
DFHRZRJ 1033
DFHRZRM 1033
DFHRZRS1 1033
DFHRZRT 1033
DFHRZRT1 1033
DFHRZRT2 1033
DFHRZSO 1033
DFHRZSO1 1033
DFHRZTA 1033
DFHRZTCX 1033
DFHRZTR1 1034
DFHRZTRI 1033
DFHRZVP1 1034
DFHRZXM 1034
DFHS22RX 1771
DFHS2AD 1102
DFHS2GF 1102
DFHS2MC2 1102
DFHS2MF 1102
DFHS2MG 1102
DFHS2PP 1102
DFHS2SQ 1102
DFHS2SR 1102
DFHSABDS 1775
DFHSAXDF 518
DFHSFP 1989
DFHSHDM 1040
DFHSHDUF 1040
DFHSHOFI 1040
DFHSHPR 1040
DFHSHRE1 1040
DFHSHRM 1040
DFHSHRQ 1040
DFHSHRQ1 1040
DFHSHRR 1041
DFHSHRRP 1041
DFHSHRSP 1041
DFHSHRT 1041
DFHSHRT1 1041
DFHSHRT2 1041
DFHSHSY 1041
DFHSHTI 1041
DFHSHTRI 1041
DFHSHVP1 1041
DFHSHXM 1041
DFHSIA1 1990
DFHSIB1 1990
DFHSIC1 1990
DFHSID1 1990
DFHSIF1 1991
DFHSIG1 1991
DFHSIH1 1991
DFHSII1 1762, 1992
DFHSIJ1 1738, 1992
DFHSIP 1993
DFHSJBD 1063
DFHSJCS 1063
DFHSJDM 1063
DFHSJDS 1063
DFHSJIN 1063
DFHSJIS 1063
DFHSJJS 1063
DFHSJNT 1063
DFHSJPJP 1063
DFHSJSM 1063
DFHSJTH 1063
DFHSKC 1778, 1780
DFHSKE 1779, 1780
DFHSKM 1778, 1780
DFHSKP 1777, 1993
DFHSMAD 1101
DFHSMAR 1101
DFHSMCK 1101
DFHSMDM 1101
DFHSMDUF 1101, 1786
DFHSMGF 1101
DFHSMMC2 1101
DFHSMMCI 1101
DFHSMMF 1101
DFHSMMG 1101
DFHSMNT 1101
DFHSMSCP 1773, 1994
DFHSMSR 1102
DFHSMST 1102
DFHSMSVC 1102
DFHSMSY 1102
DFHSMTRI 1102, 1786, 1842
DFHSMVN 1102
DFHSMVP 1102
DFHSMXDF 518
DFHSNAT 1994
DFHSNEP 1754
DFHSNNFY 1994

DFHSNP 1995
DFHSNSN 1995
DFHSNTRI 1786, 1842
DFHSNVCL 1995
DFHSNVID 1996
DFHSNVPR 1996
DFHSNVTO 1996
DFHSOAD 1163
DFHSOCK 1164
DFHSODM 1164
DFHSODUF 1164
DFHSOIS 1164
DFHSONT 1164
DFHSOPL 1164
DFHSORD 1164
DFHSOSE 1164
DFHSOSM 1164
DFHSOTB 1165
DFHSOTRI 1165
DFHSPP 1781, 1782, 1783, 1887, 1996
DFHSR1 1792
DFHSRLI 1792, 1997
DFHSRP 1787, 1792, 1997
DFHSSDUF 1786
DFHSSEN 1776, 1998
DFHSSGC 1777, 1998
DFHSSIN 1773, 1776, 1998
DFHSSMGP 1777, 1999
DFHSSMGT 1777, 1999
DFHSSWT 1777, 1999
DFHSSWTF 1777, 1999
DFHSSWTO 1777, 2000
DFHST03X 1771
DFHST04X 1771
DFHST06X 1771
DFHST08X 1771
DFHST09X 1771
DFHST14X 1771
DFHST16X 1771
DFHST17X 1771
DFHST21X 1771
DFHSTD2X 1771
DFHSTDBX 1174, 1771
DFHSTDSX 1771
DFHSTDT 2000
DFHSTDUF 1174, 1786
DFHSTDUX 1771
DFHSTE15 1771
DFHSTE35 1771
DFHSTEJX 1771
DFHSTFC 2000
DFHSTIB 2001
DFHSTIIX 1771
DFHSTIN 1771
DFHSTISX 1771
DFHSTJC 2001
DFHSTLDX 1771
DFHSTLGX 1771
DFHSTLK 2001
DFHSTLS 2001
DFHSTMNX 1771
DFHSTMQX 1771
DFHSTOT 1771
DFHSTP 1738, 2002
DFHSTPGX 1771
DFHSTRD 1771
DFHSTRMX 1771
DFHSTSJX 1771
DFHSTSMX 1771
DFHSTSOX 1771
DFHSTST 1174
DFHSTSTX 1771
DFHSTSZ 2002
DFHSTTD 2003
DFHSTTI 1174
DFHSTTM 2003
DFHSTTQX 1771
DFHSTTR 2003
DFHSTTRI 1174, 1786, 1842
DFHSTTS 2004
DFHSTTSX 1771
DFHSTU03 1771
DFHSTU04 1771
DFHSTU06 1771
DFHSTU08 1771
DFHSTU09 1771
DFHSTU14 1771
DFHSTU16 1771
DFHSTU17 1771
DFHSTU21 1771
DFHSTU22 1771
DFHSTUD2 1771
DFHSTUDB 1771
DFHSTUDS 1772
DFHSTUDU 1772
DFHSTUE 1174
DFHSTUEJ 1772
DFHSTUII 1772
DFHSTUIS 1772
DFHSTULD 1772
DFHSTULG 1772
DFHSTUMN 1772
DFHSTUMQ 1772
DFHSTUP 1768
DFHSTUP1 1772
DFHSTUPG 1772
DFHSTURM 1772
DFHSTURS 1772
DFHSTURX 1772
DFHSTUSJ 1772
DFHSTUSM 1772
DFHSTUSO 1772
DFHSTUST 1772
DFHSTUTQ 1772
DFHSTUTS 1772
DFHSTUXM 1772
DFHSTWR 1772
DFHSTXMX 1772
DFHSUDUF 1786
DFHSUEX 1875, 1876
DFHSUSN 2004
DFHSUSX 2004
DFHSUTRI 1786
DFHSUWT 1878, 2005
DFHSUZX 2005
DFHSZ2CP 1706
DFHSZ2DX 1706
DFHSZ2ID 1706
DFHSZ2IX 1706
DFHSZ2OA 1706
DFHSZ2OD 1706
DFHSZ2OR 1706
DFHSZ2OX 1706
DFHSZ2OY 1706
DFHSZ2QS 1706
DFHSZ2QX 1706
DFHSZ2SB 1706
DFHSZ2SC 1706
DFHSZ2SD 1706
DFHSZ2SH 1706
DFHSZ2SQ 1706
DFHSZ2SR 1706
DFHSZ2SX 1706
DFHSZ2TE 1706
DFHSZATC 1703
DFHSZATR 1703
DFHSZBCL 1703
DFHSZBCS 1703
DFHSZBFT 1703
DFHSZBLO 1703
DFHSZBRS 1703
DFHSZBSI 1703
DFHSZBST 1704
DFHSZBUN 1704
DFHSZBUS 1704
DFHSZDUF 1704, 1786
DFHSZFRD 1704
DFHSZFSD 1704
DFHSZIDX 1704
DFHSZPCP 1704
DFHSZPDX 1704
DFHSZPID 1704
DFHSZPIX 1704
DFHSZPOA 1704
DFHSZPOD 1704
DFHSZPOR 1704
DFHSZPOX 1704
DFHSZPOY 1704
DFHSZPQS 1704
DFHSZPQX 1704
DFHSZPSB 1704
DFHSZPSC 1704
DFHSZPSD 1704
DFHSZPSH 1704
DFHSZPSQ 1704
DFHSZPSR 1704
DFHSZPSS 1704
DFHSZPSX 1704
DFHSZPTE 1704
DFHSZRCA 1704
DFHSZRCT 1704
DFHSZRDC 1704
DFHSZRDG 1704
DFHSZRDN 1704
DFHSZRDP 1704
DFHSZRDS 1704
DFHSZRDT 1704
DFHSZREQ 1704
DFHSZRFC 1704
DFHSZRGR 1704
DFHSZRIA 1704
DFHSZRIC 1704
DFHSZRID 1704
DFHSZRIF 1705
DFHSZRII 1705
DFHSZRIN 1705
DFHSZRIO 1705
DFHSZRIP 1705
DFHSZRIQ 1705
DFHSZRIS 1705
DFHSZRIT 1705

DFHSZRIW 1705
DFHSZRNC 1705
DFHSZRNO 1705
DFHSZRPM 1705
DFHSZRPW 1705
DFHSZRQR 1705
DFHSZRQW 1705
DFHSZRRD 1705
DFHSZRRT 1705
DFHSZRSC 1705
DFHSZRSE 1705
DFHSZRST 1705
DFHSZRTM 1705
DFHSZRXD 1705
DFHSZRZZ 1705
DFHSZSIP 1705
DFHSZVBN 1705
DFHSZVGF 1705
DFHSZVQS 1705
DFHSZVRA 1705
DFHSZVRI 1705
DFHSZVSC 1705
DFHSZVSL 1705
DFHSZVSQ 1705
DFHSZVSR 1705
DFHSZVSY 1705
DFHSZWSL 1705
DFHSZXDA 1705
DFHSZXFR 1705
DFHSZXLG 1705
DFHSZXLT 1705
DFHSZXNS 1705
DFHSZXPM 1705
DFHSZXRA 1706
DFHSZXSC 1706
DFHSZXTP 1706
DFHSZYLG 1706
DFHSZYQR 1706
DFHSZYRI 1706
DFHSZYSC 1706
DFHSZYSR 1706
DFHSZYSY 1706
DFHSZZAG 1706
DFHSZZFR 1706
DFHSZZNG 1706
DFHSZZRG 1706
DFHTACP 1811, 1813, 2005
DFHTAJP 1733, 1734, 2006
DFHTBS 1496, 1498
DFHTBS00 2009
DFHTBSB 2006
DFHTBSBP 1497, 2006
DFHTBSD 2007
DFHTBSDP 2007
DFHTBSL 2007
DFHTBSLP 2007
DFHTBSQ 2008
DFHTBSQP 2008
DFHTBSR 2008
DFHTBSRP 2008
DFHTBSS 1496, 1498, 1511, 1827, 1830
DFHTBSSP 2009
DFHTC macro 1853
DFHTCBP 2009
DFHTCDUF 1786
DFHTCP 1817, 1833, 2009
DFHTCRP 1495, 1830, 2010
DFHTCRPC 2010
DFHTCRPL 2011
DFHTCRPS 2011
DFHTCRPU 2011
DFHTCT 1826
DFHTCXDF 518
DFHTDA 1871, 2011
DFHTDB 1871, 2012
DFHTDDUF 1786
DFHTDEXC 1871
DFHTDEXL 2012
DFHTDOC 1871
DFHTDP 1871, 2013
DFHTDQ 2013
DFHTDRM 1871, 2013
DFHTDRP 2014
DFHTDSUC 1871
DFHTDTM 1871, 2014
DFHTDTRI 1786
DFHTDX 2014
DFHTEP 1836, 2015
DFHTFP 1842, 1844
DFHTIDM 1184
DFHTIDUF 1184, 1786
DFHTIEM 1807
DFHTISR 1184
DFHTITRI 1184, 1786, 1842
DFHTKDUF 1786
DFHTMDUF 1786
DFHTMP 1800, 1829, 2015
DFHTOAxx 1831
DFHTOBPS 1831
DFHTON 2016
DFHTONR 1496, 1499
DFHTOR 1762, 1831, 2016
DFHTORP 2016
DFHTPE 1472
DFHTPP 1474, 1488
DFHTPP1$ 2017
DFHTPPA$ 2017
DFHTPQ 1474, 1490, 2017
DFHTPR 1474, 1491, 2017
DFHTPS 1474, 1492, 2018
DFHTR710 2019
DFHTRAO 1192
DFHTRAP 1192, 2019
DFHTRDM 1192
DFHTRDUF 1786, 1841
DFHTRFFD 1786, 1841
DFHTRFFE 1786, 1841
DFHTRFPB 1786, 1841
DFHTRFPP 1786, 1841
DFHTRIB 1786, 1841
DFHTRP 1837, 2020
DFHTRPRA 1841
DFHTRPRG 1841
DFHTRPT 1192
DFHTRPX 1192
DFHTRSR 1192
DFHTRSU 1192
DFHTRTRI 1786, 1842
DFHTRXDF 518
DFHTRZCP 2020
DFHTRZIP 2020
DFHTRZPP 2020
DFHTRZxP 1831
DFHTRZXP 2021
DFHTRZYP 2021
DFHTRZZP 2021
DFHTSBR 1227
DFHTSCL 1227
DFHTSDM 1228
DFHTSDUF 1786
DFHTSITR 1228, 1842
DFHTSP 1228, 2021
DFHTSPT 1228
DFHTSQR 1228
DFHTSRM 1228
DFHTSSH 1228
DFHTSSQ 1229
DFHTSSR 1229
DFHTSST 1229
DFHTT710 2022
DFHTTPDS 1472
DFHTU710 1839
DFHUCNV 2022
DFHUEDUF 1786
DFHUEH 1874, 1876, 2022
DFHUEM 1580, 1802, 1806, 1873, 1876, 2023
DFHUSAD 1251
DFHUSBP 2023
DFHUSDM 1252
DFHUSDUF 1252, 1786
DFHUSFL 1252
DFHUSIS 1252
DFHUSST 1252
DFHUSTI 1252
DFHUSTRI 1252, 1786, 1842
DFHUSXM 1252
DFHW2A 1334
DFHW2AC 1334
DFHW2AS 1334
DFHW2AT 1335
DFHW2DM 1335
DFHW2DUF 1335
DFHW2FD 1335
DFHW2FI 1335
DFHW2FR 1335
DFHW2RP 1335
DFHW2SD 1335
DFHW2ST 1335
DFHW2TRI 1335
DFHW2TS 1335
DFHW2TT 1335
DFHW2UE 1335
DFHW2W2 1335
DFHWBA 1530
DFHWBA1 1531
DFHWBAAX 1530
DFHWBADX 1530
DFHWBAP 1321
DFHWBAPF 1321
DFHWBBLI 1531
DFHWBCL 1321, 1531
DFHWBDM 1321
DFHWBERX 1530
DFHWBGB 1531
DFHWBIP 1530
DFHWBLT 1531
DFHWBQM 1321
DFHWBRP 1321
DFHWBSR 1321
DFHWBST 1531

DFHWBTC 1531
DFHWBTTA 1531
DFHWBTTB 1531
DFHWBTTC 1531
DFHWBUR 1321
DFHWBXM 1322
DFHWBXN 1530
DFHWCCS 2023
DFHWCGNT 2024
DFHWDATT 2024
DFHWDINA 2024
DFHWDISP 2025
DFHWDSRP 2025
DFHWDWAT 2025
DFHWKP 1828, 2026
DFHWLFRE 2026
DFHWLGET 2026
DFHWMG1 2026
DFHWMI 2027
DFHWMMT 2027
DFHWMP1 2028
DFHWMPG 2027
DFHWMQG 2028
DFHWMQH 2028
DFHWMQP 2028
DFHWMQS 2029
DFHWMRD 2029
DFHWMS 2029
DFHWMS20 2030
DFHWMWR 2030
DFHWOS 2030
DFHWOSA 2030
DFHWOSB 2031
DFHWSRTR 2031
DFHWSSN1 2031
DFHWSSN2 2032
DFHWSSN3 2032
DFHWSSOF 2032
DFHWSSR 2033
DFHWSSW 2033
DFHWSTI 2033
DFHWSTKV 2034
DFHWSXPI 2035
DFHWTI 2035
DFHWTO 1878
DFHWTRP 2035
DFHXCALL 1586
DFHXCDMP 1586
DFHXCEIP 1586
DFHXCO 1586
DFHXCOPT 1586
DFHXCP 2036
DFHXCP1 2036
DFHXCPC 2036
DFHXCPLD 1587
DFHXCPLH 1587
DFHXCPLL 1587
DFHXCPLO 1587
DFHXCPRH 1587
DFHXCRCD 1587
DFHXCRCH 1587
DFHXCRCL 1587
DFHXCRCO 1587
DFHXCSTB 1586
DFHXCSVC 1587
DFHXCTAB 1587
DFHXCTRA 1587
DFHXCTRD 1587
DFHXCTRI 1587
DFHXCTRP 1587
DFHXCURM 1587
DFHXDXDF 518
DFHXFP 1707, 2036
DFHXFQ 2037
DFHXFX 1707, 2037
DFHXMAB 1402
DFHXMAT 1402
DFHXMBD 1402
DFHXMCL 1403
DFHXMDD 1403
DFHXMDM 1403
DFHXMDUF 1403, 1786
DFHXMER 1403
DFHXMFD 1403
DFHXMIQ 1403
DFHXMLD 1403
DFHXMQC 1403
DFHXMQD 1403
DFHXMRP 1404
DFHXMSR 1404
DFHXMST 1404
DFHXMTRI 1404, 1786, 1842
DFHXMXD 1404
DFHXMXE 1404
DFHXRA 2037
DFHXRB 2038
DFHXRC 2038
DFHXRCP 2038
DFHXRDUF 1786
DFHXRE 2038
DFHXRP 2039
DFHXRSP 2039
DFHXRXDF 519
DFHXSAD 1450
DFHXSDM 1451
DFHXSDUF 1451, 1786
DFHXSFL 1451
DFHXSIS 1451
DFHXSLU 1451
DFHXSMN 2039
DFHXSMX 2040
DFHXSPW 1451
DFHXSRC 1451
DFHXSS 2040
DFHXSSA 1451
DFHXSSB 1451, 2040
DFHXSSC 1451
DFHXSSD 1451
DFHXSSI 1451
DFHXSTRI 1451, 1842
DFHXSWM 2041
DFHXTCI 2041
DFHXTP 1859, 1865, 2041
DFHZABD 2041
DFHZACT 1750, 2042
DFHZAIT 2042
DFHZAND 2042
DFHZARER 1556, 2043
DFHZARL 1550, 1551, 1553, 1556, 2043
DFHZARM 1550, 1554, 2044
DFHZARQ 1550, 1556, 1732, 2044
DFHZARR 1550, 1551, 1556, 2044
DFHZARR0 1556, 2045
DFHZARR1 1557, 2045
DFHZARRA 1556, 2045
DFHZARRC 1556, 1557, 2045
DFHZARRF 1556, 2045
DFHZASX 2046
DFHZATA 1459, 1495, 1827, 2046
DFHZATA2 1459
DFHZATD 1459, 1827, 2046
DFHZATDX 1459, 1460, 2047
DFHZATDY 1459
DFHZATI 2047
DFHZATMD 2047
DFHZATMF 2048
DFHZATR 1459, 2048
DFHZATS 1459, 1827, 2048
DFHZATT 2049
DFHZBAN 2049
DFHZBKT 1887, 2049
DFHZBLX 1881, 2049
DFHZCA 2050
DFHZCB 2050
DFHZCC 2051
DFHZCGRP 1903
DFHZCHS 1887, 2052
DFHZCLS 2052
DFHZCLS, close destination program 1456
DFHZCLX 1457, 2052
DFHZCNA 1824, 2053
DFHZCNR 1824, 2053
DFHZCNT 1887, 2053
DFHZCP 1732, 1817, 1833, 2054
DFHZCQ 1495, 1512, 1763, 1827, 2054
DFHZCQDL 2055
DFHZCQIN 2055
DFHZCQIQ 2055
DFHZCQIS 2055
DFHZCQIT 2056
DFHZCQRS 2056
DFHZCQRT 1497
DFHZCRQ 2056
DFHZCRT 1887, 2056
DFHZCUT 2057
DFHZCW 2057
DFHZCX 1731, 2057
DFHZCXR 2058
DFHZCY 2058
DFHZCZ 2059
DFHZDET 2059
DFHZDSP 2060
DFHZDST 2060
DFHZEMW 2060
DFHZERH 1550, 1551, 1556, 1558, 2061
DFHZEV1 2061
DFHZEV2 2061
DFHZFRE 2061
DFHZGAI 1459
DFHZGCA 1895, 1903
DFHZGCC 1903
DFHZGCH 1882
DFHZGCN 1893, 1903
DFHZGDA 1904
DFHZGET 2062
DFHZGIN 1882
DFHZGPC 1904
DFHZGPR 1904
DFHZGRP 1904
DFHZGSL 1904

DFHZGTA 1496, 1499
DFHZGTI (terminal location) 1828
DFHZGUB 1905
DFHZHPRX 2062
DFHZHPSR 2062
DFHZIS1 1731, 2063
DFHZIS2 1731, 2063
DFHZISP 1550, 1551, 1558, 1732, 2063
DFHZLEX 2064
DFHZLGX 2064
DFHZLOC 2064
DFHZLOC (terminal location) 1828
DFHZLRP 2065
DFHZLS1 1893
DFHZLTX 2065
DFHZLUS 2065
DFHZMJM 1753
DFHZNAC 1750, 1753, 1820, 2066
DFHZNCA 1457, 1753
DFHZNCE 1753
DFHZNCM 1753
DFHZNCS 1753
DFHZNCV 1753
DFHZNEP 1754, 2066
DFHZNSP 2066
DFHZOPA 2067
DFHZOPN 2067
DFHZOPX 2067
DFHZQUE 2068
DFHZRAC 2068
DFHZRAQ 2068
DFHZRAR 2069
DFHZRAS 2069
DFHZRLG 2069
DFHZRLP 1889, 2069
DFHZRLX 1887, 1892, 2070
DFHZRRX 2070
DFHZRSP 2070
DFHZRST 2071
DFHZRSY 2071
DFHZRVL 1887, 1889, 2071
DFHZRVS 2071
DFHZRVX 2072
DFHZSAX 2072
DFHZSCX 2072
DFHZSDA 2073
DFHZSDL 1887, 1891, 2073
DFHZSDR 2073
DFHZSDS 2074
DFHZSDX 2074
DFHZSES 2074
DFHZSEX 2075
DFHZSHU 2075
DFHZSIM 2075
DFHZSIX 2076
DFHZSKR 2076
DFHZSLS 2076
DFHZSLX 1887, 1892, 2077
DFHZSSX 2077
DFHZSTAP 1550, 1551, 1558, 2077
DFHZSTU 2077
DFHZSUP 1732, 1850, 1855, 2078
DFHZSYN 2078
DFHZSYX 2078
DFHZTAX 2079
DFHZTPX 2079
DFHZTRA 2079
DFHZTSP 1495, 1827, 1851, 1856, 1857, 1865, 2080
DFHZUCT 2080
DFHZUIX 2080
DFHZUSR 2080
DFHZXCU 2081
DFHZXDUF 1786
DFHZXPS 1905
DFHZXQO 2081
DFHZXRC 1905, 2081
DFHZXRE0 2082
DFHZXRL 1849, 1851, 1853, 1865, 2082
DFHZXRT 1865, 2082
DFHZXST 2083
DFXUSTRI 1786
DGB (DBCTL global block) 1545
DH (Document handler domain)
  domain 384
DHDH gate
  CREATE_DOCUMENT function 384
  DELETE_BOOKMARK function 386
  DELETE_DATA function 386
  DELETE_DOCUMENT function 387
  INQUIRE_DOCUMENT function 387
  INSERT_BOOKMARK function 388
  INSERT_DATA function 388
  REPLACE_DATA function 389
  RETRIEVE_WITH_CTLINFO function 391
  RETRIEVE_WITHOUT_CTLINFO function 391
  SET_PARAMETERS function 392
DHFS gate
  DELETE_HFS_FILE function 392
  END_BROWSE_DIRECTORY function 393
  GET_NEXT_IN_DIRECTORY function 393
  INQUIRE_HFS_FILE function 393
  MAKE_HFS_DIRECTORY function 394
  READ_HFS_FILE function 394
  START_BROWSE_DIRECTORY function 395
  WRITE_HFS_FILE function 396
DHSL gate
  ADD_SYMBOL_LIST function 397
  EXPORT_SYMBOL_LIST function 397
  IMPORT_SYMBOL_LIST function 398
  SET_SYMBOL_VALUE_BY_API function 398
  SET_SYMBOL_VALUE_BY_SSI function 399
DHTM gate
  ADD_REPLACE_DOCTEMPLATE function 399
  DELETE_DOCTEMPLATE function 401
  END_BROWSE function 401
  GET_NEXT function 402
  INITIALIZE_DOCTEMPLATES function 403
  INQUIRE_DOCTEMPLATE function 403
DHTM gate *(continued)*
  INQUIRE_TEMPLATE_STATUS function 405
  READ_TEMPLATE function 405
  START_BROWSE function 406
DIB (data interchange block) 1547
DIB (DL/I interface block) 1561
DIP (data interchange program) 1546
  storage control 1547
  temporary-storage control 1547
  terminal control 1547
  trace control 1547
Directory manager domain 374
DISABLE routine of DFHUEM 1874
DISABLE_FILE function, FCFS gate 223
DISABLE_OSGIBUNDLE function, SJTH gate 1052
DISABLE_STATISTICS function, STST gate 1167
DISCARD function, LGJN gate 660
DISCARD function, LGLD gate 669
DISCARD function, RLCB gate 943
DISCARD_AIDS function, TFAL gate 312
DISCARD_BUNDLE function, RLPM gate 930
DISCARD_BUNDLE_BY_ID_VER function, APRL gate 42
DISCARD_ENQMODEL function, NQRN gate 767
DISCARD_EPADAPTER function, EPAD gate 534
DISCARD_IPCONN function, ISIC gate 553
DISCARD_JSONTRANSFRM function, MLJS gate 703
DISCARD_JVMSERVER function, SJJS gate 1048
DISCARD_LIBRARY function, LDLB gate 615
DISCARD_PIPELINE function, PIPL gate 890
DISCARD_POLICY function, MPDD gate 742
DISCARD_POLICYSCOPE function, MPDD gate 742
DISCARD_PROCESSTYPE function, BATT gate 360
DISCARD_TDQDEF function, TDTM gate 299
DISCARD_WEBSERVICE function, PIWR gate 908
DISCARD_XMLTRANSFORM function, MLXT gate 712
discard, EXEC CICS 1517
DISCARDED function, RLRO gate 937
DISCONNECT for IRC 1729
DISCONNECT function, LGLB gate 667
DISCONNECT function, LGST gate 675
DISCONNECT_ALL function, LGLB gate 668
DISCONNECT_CFDT_POOLS function, FCDS gate 146
DISCONNECT_DSNB function, FCDN gate 138
disconnection flow for terminals, LU-initiated 1456

DISPATCH_EVENT function, EPEV gate 536
Dispatcher 454, 455, 456, 457
Dispatcher domain (DS) domain 454
distributed program link (DPL) 1547
distributed transaction processing (DTP)
  logical unit type 6.1 (LU6.1) protocol 1549
  session failures 1550
  system failures 1550
DL/I
  PSB scheduling 1544, 1760
  PSB termination 1544, 1760
  remote 1760
  system definition 1760
DL/I database support 1535, 1559, 1760
DL/I interface
  CALL macro 1560
  CALLDLI macro 1560
  EXEC DLI command 1560
  IMS service modules 1560
  program specification block (PSB) 1560
DL/I interface block (DIB) 1561
DL/I interface parameter list (DLP) 1545, 1561
DL/I interface program 1559
DL/I request handling, function shipping 1717
DL/I support 1535, 1559, 1760
DLP (DL/I interface parameter list) 1545, 1561
DM (Domain manager domain) domain 409
DMDM gate
  ADD_DOMAIN function 409
  INITIALISE_DOMAIN function 416
  PRE_INITIALISE function 417
  QUIESCE_DOMAIN function 417
  QUIESCE_SYSTEM function 410
  SET_PHASE function 410
  TERMINATE_DOMAIN function 417
  WAIT_PHASE function 411
DMEN gate
  DELETE function 411
  LISTEN function 412
  PROCESS_ENF_EVENT function 418
DMIQ gate
  END_BROWSE function 413
  GET_NEXT function 413
  INQ_DOMAIN_BY_ID function 414
  INQ_DOMAIN_BY_NAME function 414
  INQ_DOMAIN_BY_TOKEN function 415
  START_BROWSE function 415
Document handler domain (DH) domain 384
domain
  Directory manager (DD) 374
domain calls 1
  formats 6
  generic 6
  specific 6
  tokens 6
domain gates 5
Domain manager domain (DM) domain 409
domains 1
  application (AP) 10
  Application Manager Domain (AP) 10
  Asynchronous services Domain (AS) 8
  Business application manager domain (BA) 335
  CICS catalog domain (CC) 367
  debugging profile (DP) 419
  Dispatcher domain (DS) 454
  Document handler domain (DH) 384
  Domain manager domain (DM) 409
  Dump Domain (DU) 493
  Enqueue domain (NQ) 752
  Enterprise Java domain (EJ) 519
  Event manager domain (EM) 521
  Event processing domain (EP) 533
  inter-system 546
  IP ECI (IE) 543
  IS 546
  JVM domain (SJ) 1041
  Kernel domain (KE) 582
  Loader domain (LD) 612
  Lock manager domain (LM) 683
  Logger Domain (LG) 645
  Managed platform domain (MP) 740
  Markup language domain (ML) 703
  Message domain (ME) 686
  Monitoring domain (MN) 716
  object transaction service (OT) 773
  Parameter manager domain (PA) 781
  Partner management domain (PT) 917
  Pipeline manager domain (PI) 868
  Program manager domain (PG) 784
  Recovery manager domain (RM) 948
  Region status domain (RS) 1012
  Request streams domain (RZ) 1024
  Resource life-cycle domain (RL) 930
  RRMS (RX) 1020
  Scheduler services domain (SH) 1034
  Security domain (XS) 1404
  Sockets domain (SO) 1103
  Statistics domain (ST) 1165
  Storage manager domain (SM) 1063
  Temporary Storage Domain (TS) 1193
  Timer domain (TI) 1175
  Trace domain (TR) 1184
  transaction manager (XM) 1335
  User domain (US) 1229
  Web 2.0 Domain (W2) 1322
  Web domain (WB) 1252
DP (debugging profile) domain 419
DPFM gate
  ACTIVATE_DEBUG_PROFILE function 419
  DELETE_DEBUG_PROFILE function 420
  END_PM_BROWSE function 421
  GET_DEBUG_PROFILE function 421
  INACTIVATE_DEBUG_PROFILE function 423
DPFM gate *(continued)*
  READNEXT_PM_PROFILE function 424
  REPLACE_DEBUG_PROFILE function 426
  SAVE_DEBUG_PROFILE function 428
  START_PM_BROWSE function 431
DPIQ gate
  INQUIRE_DEBUG_TASK function 431
  INQUIRE_PARAMETERS function 432
  SET_DEBUG_PROFILE function 432
  SET_DEBUGGING function 433
  SET_PARAMETERS function 433
DPLM gate
  ENDBR_DEBUG_PROFILES function 433
  READNEXT_DEBUG_PROFILE function 434
  READNEXT_INPUT function 436
  RESTARTBR_DEBUG_PROFILES function 438
  STARTBR_DEBUG_PROFILES function 439
  UPDATE_PROFILE_IN_LIST function 440
DPPM gate
  PATTERN_MATCH_PROFILE function 440
  PATTERN_MATCH_TASK function 443
DPUM gate
  GET_USER_DEFAULTS function 443
  SAVE_USER_DEFAULTS function 445
DPWD gate
  PROCESS_PAGE function 448
  PROCESS_SUBMIT function 449
DPWE gate
  PROCESS_PAGE function 449
  PROCESS_SUBMIT function 450
DPWJ gate
  PROCESS_PAGE function 450
  PROCESS_SUBMIT function 451
DPWL gate
  PROCESS_PAGE function 451
  PROCESS_SUBMIT function 452
DPXM gate
  BIND_XM_CLIENT function 453
  INIT_XM_CLIENT function 453
  RELEASE_XM_CLIENT function 453
DRA (database resource adapter) 1535
DRAIN_CONTROL_ACB function, FCCA gate 105
DRAIN_EVENT_ADAPTER function, EPEV gate 536
DRIVE_PENDING function, RLRO gate 937
DS (Dispatcher domain) domain 454
DSAC gate
  ASSOCIATE function 454, 455, 456, 457
DSAT gate
  ATTACH function 457
  CANCEL_TASK function 458

DSAT gate *(continued)*
CHANGE_MODE function 459
CHANGE_PRIORITY function 461
CLEAR_MATCH function 462
DELETE_SUBSPACE_TCBS function 462
FORCE_PURGE_INHIBIT_QUERY function 491
FREE_SUBSPACE_TCBS function 463
NOTIFY_DELETE_TCB function 492
PURGE_INHIBIT_QUERY function 491
RELEASE_OPEN_TCB function 463
SET_PRIORITY function 463
SET_TRANSACTION_TOKEN function 464
TASK_REPLY function 490
TCB_POOL_MANAGEMENT function 465
DSB (data stream build) 1477
interfaces, illustrated 1477
page and text build (PBP) 1478
terminal page processor (TPP) 1478
DSB (DBCTL scheduling block) 1545
DSBR gate
END_BROWSE function 465
GET_NEXT function 465
INQUIRE_TASK function 467
INQUIRE_TCB function 469
SET_TASK function 470
SET_TCB function 471
START_BROWSE function 471
DSIT gate
ACTIVATE_MODE function 472
ADD_TCB function 474
DELETE_ALL_OPEN_TCBS function 474
DELETE_OPEN_TCB function 475
DELETE_TCB function 475
FREE_TCB function 475
INQUIRE_DISPATCHER function 476
PROCESS_DEAD_TCBS function 477
SET_DISPATCHER function 477
DSMT gate
END_BROWSE_MVSTCB function 478
GET_NEXT_MVSTCB function 479
INQUIRE_MVSTCB function 479
SNAPSHOT_MVSTCBS function 480
START_BROWSE_MVSTCB function 480
DSNAME block manager, file control (DFHFCDN) 1617
DSNB (data set name block) 1604
DSNCUEXT 1527
DSSR gate
ADD_SUSPEND function 481
DELETE_SUSPEND function 481
RESUME function 481
SUSPEND function 482
WAIT_MVS function 484
WAIT_OLDC function 486
WAIT_OLDW function 487
DU (Dump Domain) domain 493
DUDT gate
ADD_SYSTEM_DUMPCODE function 493
ADD_TRAN_DUMPCODE function 494
DELETE_SYSTEM_DUMPCODE function 495
DELETE_TRAN_DUMPCODE function 495
ENDBR_SYSTEM_DUMPCODE function 495
ENDBR_TRAN_DUMPCODE function 496
GETNEXT_SYSTEM_DUMPCODE function 496
GETNEXT_TRAN_DUMPCODE function 497
INQUIRE_SYSTEM_DUMPCODE function 498
INQUIRE_TRAN_DUMPCODE function 499
SET_SYSTEM_DUMPCODE function 500
SET_TRAN_DUMPCODE function 501
STARTBR_SYSTEM_DUMPCODE function 503
STARTBR_TRAN_DUMPCODE function 503
DUDU gate
SYSTEM_DUMP function 503
TRANSACTION_DUMP function 504
DUFT gate
DEREGISTER function 507
INQUIRE_FEATURE function 507
REGISTER function 508
UPDATE_FEATURE function 509
Dump Domain (DU) domain 493
dump utility program (DFHDUxxx) 1563
DUMP_DATA function, EJDU gate 519
DUMP_STACK function, EJDU gate 519
DUMPDS_CLOSE function, DUSR gate 510
DUMPDS_OPEN function, DUSR gate 510
DUMPDS_SWITCH function, DUSR gate 510
dumps
finding in dump 1798
finding table entries in 1798
dumps, FEPI
interpreted areas 1701
DUSR gate
CROSS_SYSTEM_DUMP_AVAIL function 509
DUMPDS_CLOSE function 510
DUMPDS_OPEN function 510
DUMPDS_SWITCH function 510
INQUIRE_CURRENT_DUMPDS function 510
INQUIRE_DUMPDS_AUTOSWITCH function 511
INQUIRE_DUMPDS_OPEN_STATUS function 511
DUSR gate *(continued)*
INQUIRE_INITIAL_DUMPDS function 511
INQUIRE_RETRY_TIME function 512
INQUIRE_SYSTEM_DUMP function 512
SET_DUMPDS_AUTOSWITCH function 512
SET_DUMPTABLE_DEFAULTS function 513
SET_INITIAL_DUMPDS function 513
SET_RETRY_TIME function 514
SET_SYSTEM_DUMP function 514
SET_TRANTABLESIZE function 514
SET_TRANTABLETYPE function 515
DWE (deferred work element) 1783
DYN_CREATE_WEBSERVICE function, PISC gate 898
dynamic allocation sample program (DYNALLOC) 1565
dynamic log
as used by file control 1591
for restartable transactions 1844
dynamic transaction backout
READ UPDATE request 1594
WRITE request 1595
dynamic transaction backout (DTB)
transaction restart 1844

E

ECIS gate
END_BROWSE function 79
END_BROWSE_CAPTURESPEC function 79
END_BROWSE_DATA_PRED function 79
END_BROWSE_INFO_SOURCE function 80
END_BROWSE_OPTION_PRED function 80
GET_NEXT_CAPTURESPEC function 80
GET_NEXT_DATA_PRED function 83
GET_NEXT_ENTRY function 84
GET_NEXT_INFO_SOURCE function 84
GET_NEXT_OPTION_PRED function 85
INQ_CAPTURESPEC function 86
INQ_EVENTBINDING function 88
INQ_EVENTPROCESS function 89
SET_EVENTBINDING function 90
SET_EVENTPROCESS function 89
START_BROWSE_CAPTURESPEC function 90
START_BROWSE_DATA_PRED function 90
START_BROWSE_EVENTBINDING function 91
START_BROWSE_INFO_SOURCE function 91
START_BROWSE_OPTION_PRED function 92

ECSC gate
SYSTEM_EVENT_STATUS function 95
SYSTEM_EVENT_THRESHOLD function 92, 93, 96
ECSE gate
SIGNAL_EVENT function 97
EDF (execution diagnostic facility) 1583
EIB (EXEC interface block) 1571
EIC (EXEC interface communication area) 1572
EIP (EXEC interface program) 1570
EIS (EXEC interface storage) 1572
EJ (Enterprise Java domain) domain 519
EJDU gate
DUMP_DATA function 519
DUMP_STACK function 519
INQUIRE_TRACE_FLAGS function 520
EM (Event manager domain) domain 521
EMBR gate
END_BROWSE_EVENT function 521
END_BROWSE_TIMER function 521
GET_NEXT_EVENT function 521
GET_NEXT_TIMER function 522
INQUIRE_EVENT function 523
INQUIRE_TIMER function 523
START_BROWSE_EVENT function 524
START_BROWSE_TIMER function 525
EMEM gate
ADD_SUBEVENT function 525
CHECK_TIMER function 525
DEFINE_ATOMIC_EVENT function 526
DEFINE_COMPOSITE_EVENT function 526
DEFINE_TIMER function 527
DELETE_EVENT function 528
DELETE_TIMER function 528
FIRE_EVENT function 529
FORCE_TIMER function 529
INQUIRE_STATUS function 530
REMOVE_SUBEVENT function 530
RETRIEVE_REATTACH_EVENT function 531
RETRIEVE_SUBEVENT function 531
TEST_EVENT function 532
emergency restart 1516
after 1517
ENABLE routine of DFHUEM 1873
ENABLE_FILE function, FCFS gate 224
ENABLE_OSGIBUNDLE function, SJTH gate 1053
ENCODE_ODR function, MNOD gate 730
END_ATOMS function, APRD gate 39
END_BACKOUT function, RMRO gate 998
END_BROWSE function, AIIQ format 1466
END_BROWSE function, CCCC gate 368
END_BROWSE function, DDBR gate 380
END_BROWSE function, DHTM gate 401
END_BROWSE function, DMIQ gate 413
END_BROWSE function, DSBR gate 465
END_BROWSE function, ECIS gate 79
END_BROWSE function, EPIS gate 538
END_BROWSE function, FCFR gate 163
END_BROWSE function, FCRF gate 267
END_BROWSE function, LDLD gate 630
END_BROWSE function, LGJN gate 660
END_BROWSE function, LGLD gate 670
END_BROWSE function, LGST gate 675
END_BROWSE function, SOTB gate 1156
END_BROWSE function, TSBR gate 1195
END_BROWSE function, TSMB gate 1198
END_BROWSE function, TSSH gate 1215
END_BROWSE function, WBAP gate 1252
END_BROWSE_ALL function, LGBA gate 646
END_BROWSE_APPLICATION function, PGIS gate 828
END_BROWSE_ATOMSERVICE function, W2AT gate 1327
END_BROWSE_BUNDLE function, RLPM gate 931
END_BROWSE_BUNDLERES function, RLRO gate 937
END_BROWSE_CAPTURESPEC function, ECIS gate 79
END_BROWSE_CHAINS function, LGCC gate 650
END_BROWSE_DATA_PRED function, ECIS gate 79
END_BROWSE_DIRECTORY function, DHFS gate 393
END_BROWSE_ENQMODEL function, NQRN gate 767
END_BROWSE_ENQUEUE function, NQIB gate 755
END_BROWSE_EVENT function, EMBR gate 521
END_BROWSE_FILE function, FCMT gate 242
END_BROWSE_HEADERS function, WBCL gate 1263
END_BROWSE_HOST function, WBUR gate 1298
END_BROWSE_INFO_SOURCE function, ECIS gate 80
END_BROWSE_JVMSERVER function, SJJS gate 1049
END_BROWSE_LIBRARY function, LDLB gate 616
END_BROWSE_MVSTCB function, DSMT gate 478
END_BROWSE_OPTION_PRED function, ECIS gate 80
END_BROWSE_OSGIBUNDLE function, SJBD gate 1041
END_BROWSE_OSGISERVICE function, SJBD gate 1042
END_BROWSE_OSGISERVICE function, SJTH gate 1053
END_BROWSE_PIPELINE function, PIPL gate 891
END_BROWSE_PROCESSTYPE function, BATT gate 360
END_BROWSE_PROGRAM function, PGIS gate 828
END_BROWSE_RESULTS function, DDAP gate 375
END_BROWSE_TCLASS function, XMCL gate 1350
END_BROWSE_TDQDEF function, TDTM gate 299
END_BROWSE_TIMER function, EMBR gate 521
END_BROWSE_TRANDEF function, XMBD gate 1343
END_BROWSE_TRANSACTION function, XMIQ gate 1360
END_BROWSE_TXN_TOKEN function, XMIQ gate 1360
END_BROWSE_URIMAP function, WBUR gate 1298
END_BROWSE_WEBSERVICE function, PIWR gate 909
END_BROWSE_XMLTRANSFORM function, MLXT gate 716
END_CHAIN_BROWSE function, LGCB gate 648
END_DELIVERY function, RMDE gate 1002
END_DRAIN_EVENT_ADAPTER function, EPEV gate 536
END_DSNB_BROWSE function, FCDN gate 139
END_FILE_IN_POOL_BROWSE function, FCST gate 283
END_LINK_BROWSE function, RMLN gate 954
END_PM_BROWSE function, DPFM gate 421
END_POOL_BROWSE function, PTTW gate 921
END_SUBPOOL_BROWSE function, S2AD gate 1090
END_SUBPOOL_BROWSE function, SMAD gate 1066
END_SYSTEM_DUMP function, RSDU gate 1012
END_TASK function, KEDS gate 596
END_TRANSACTION function, RSXM gate 1019
END_TRANSACTION function, USXM gate 1246
END_TRANSACTION function, XSXM gate 1450
END_TRANSACTION_DUMP function, RSDU gate 1013
END_TSPOOL_BROWSE function, TSSH gate 1215
END_UOW_BROWSE function, RMUW gate 984
END_UOWDSN_BROWSE function, FCFL gate 155
END_WORK_TOKEN_BROWSE function, RMWT gate 993

END_WRITE function, CCCC gate 368
ENDBR_ACTIVITY function, BABR gate 341
ENDBR_CONTAINER function, BABR gate 342
ENDBR_CONTAINER function, PGCR gate 797
ENDBR_DEBUG_PROFILES function, DPLM gate 433
ENDBR_PROCESS function, BABR gate 342
ENDBR_SYSTEM_DUMPCODE function, DUDT gate 495
ENDBR_TRAN_DUMPCODE function, DUDT gate 496
ENDBROWSE_IPCONN function, ISIC gate 553
ENF servicer, file control (DFHFCES) 1620
Enqueue domain (NQ) domain 752
ENQUEUE function, NQED gate 753
Enterprise Java domain (EJ) domain 519
environment, function shipping 1710
EP (Event processing) domain 533
EP domain
  modules 543
EPAD gate
  ADD_EPADAPTER function 533
  DISCARD_EPADAPTER function 534
  FREE_EPADAPTER function 534
  RECOVER_EPADAPTERS function 534
  SET_EPADAPTER function 534
EPAS gate
  FORMAT_EVENT function 535
EPB (exit program block) 1800, 1805, 1872
EPEV gate
  COMMIT_EVENT function 536
  DISPATCH_EVENT function 536
  DRAIN_EVENT_ADAPTER function 536
  END_DRAIN_EVENT_ADAPTER function 536
  PUT_EVENT function 537
  QUEUE_EVENT function 537
EPIS gate
  END_BROWSE function 538
  GET_NEXT_ENTRY function 538
  INQ_EPADAPTER function 540
  SET_EPADAPTER function 542
  SET_EVENT_PROCESSING function 542
  START_BROWSE_EPADAPTER function 542
EPL (exit program link) 1872
ERROR function, LGGL gate 680
ESDS (entry-sequenced data set)
  invalid DELETE request 1598
  WRITE request processing 1595
ESTABLISH function, SOCK gate 1113
ESTABLISH_LANGUAGE function, APLI gate 31
ESTABLISH_PIPELINE function, PIPL gate 891
ESTAE exit routine 1728, 1780, 1781
ESTIMATE_ALL function, APCR gate 23
ESTIMATE_CHANGED function, APCR gate 24
Event manager domain (EM) domain 521
Event processing (EP) domain 533
EXCEPTION_DATA_PUT function, MNMN gate 727
exchange log name (XLN) 1887
EXCI (external CICS interface) 1584
  design overview 1584
  EXEC CALL interface 1585
  EXEC CICS interface 1585
  programming interfaces 1585
EXEC CALL interface of EXCI 1585
EXEC CICS CREATE 1517
EXEC CICS DELETE command 1598
EXEC CICS DISCARD 1517
EXEC CICS ENDBR command 1600
EXEC CICS interface of EXCI 1585
EXEC CICS READ command 1594
EXEC CICS READNEXT command 1599
EXEC CICS READPREV command 1599
EXEC CICS RESETBR command 1599
EXEC CICS REWRITE command 1597
EXEC CICS STARTBR command 1599
EXEC CICS UNLOCK command 1598
EXEC CICS WRITE command 1595
EXEC DLI command
  DL/I interface 1560
EXEC interface block (EIB) 1571
EXEC interface communication area (EIC) 1572
EXEC interface module, file control (DFHEIFC) 1613
EXEC interface program (EIP) 1570
EXEC interface storage (EIS) 1572
execution diagnostic facility (EDF) 1583
exit interface, user
  control blocks, illustrated 1875
exit program block (EPB) 1800, 1805, 1872
exit program link (EPL) 1872
exits
  VTAM 1462
  z/OS Communications Server 1911
EXPLICIT_OPEN function, LGJN gate 661
EXPORT_ALL function, APCR gate 24
EXPORT_CERTIFICATE_DATA function, SOIS gate 1131
EXPORT_CHANGED function, APCR gate 25
EXPORT_SYMBOL_LIST function, DHSL gate 397
external CICS interface (EXCI) 1584
  design overview 1584
  EXEC CALL interface 1585
  EXEC CICS interface 1585
  programming interfaces 1585
extract statistics reporting function 1770
EXTRACT_ADS function, CCNV gate 63
EXTRACT_CFDT_STAT function, FCDS gate 146
EXTRACT_PROCESS function, APTC gate 47
EXTRACT_STATISTICS function, FCCT gate 126
EXTRACT-EXIT routine of DFHUEM 1874
extrapartition transient data queues 1867
  interfaces, illustrated 1870
  referencing using indirect queues 1867

F

facility control area associated address
  task control area (TCA)
    terminal control 1818
fast file locate element (FFLE) 1601
FBWA (file browse work area)
  FREEMAIN in ENDBR request processing 1600
  RESETBR request 1599
  STARTBR request 1599
FC static (file control static storage) 1605
FCAT gate
  INQ_BASEDSNAME function 98
  INQ_CATALOG_QUIESCESTATE function 99
  INQ_CATALOG_RECOV_REQD function 99
  INQ_DATASET_STATE function 100
  SET_BWO_BITS_DISABLED function 100
  SET_BWO_BITS_ENABLED function 101
  SET_CATALOG_RECOV_POINT function 101
  SET_CATALOG_RECOV_REQD function 102
  SET_CATALOG_RECOVERED function 103
FCBU gate
  TIDY_UP_FILE function 103
FCCA gate
  CHECK function 103
  COLD_START_RLS function 104
  DRAIN_CONTROL_ACB function 105
  INQUIRE_RECOVERY function 105
  LOST_LOCKS_COMPLETE function 106
  QUIESCE_COMPLETE function 107
  QUIESCE_REQUEST function 107
  REGISTER_CONTROL_ACB function 108
  RELEASE_LOCKS function 109
  RESET_NONRLS_BATCH function 110
  RETAIN_DATASET_LOCKS function 110
  RETAIN_UOW_LOCKS function 111
  UNREGISTER_CONTROL_ACB function 112
FCCI gate
  INQUIRE function 112
FCCR DELETE
  file control 1658
  file control, FCCR DELETE function 1658

FCCR DELETE_MULTIPLE
file control 1659
file control, FCCR DELETE_MULTIPLE function 1659
FCCR gate
DELETE function 114
DELETE_MULTIPLE function 116
HIGHEST function 117
LOAD function 118
POINT function 119
READ function 119
READ_DELETE function 121
REWRITE function 121
UNLOCK function 122
WRITE function 123
FCCR HIGHEST
file control 1651
file control, FCCR HIGHEST function 1651
FCCR LOAD
file control 1654
file control, FCCR LOAD function 1654
FCCR POINT
file control 1651
file control, FCCR POINT function 1651
FCCR READ
file control 1652
file control, FCCR READ function 1652
FCCR READ_DELETE
file control 1653
file control, FCCR READ_DELETE function 1653
FCCR REWRITE
file control 1656
file control, FCCR REWRITE function 1656
FCCR UNLOCK
file control 1654
file control, FCCR UNLOCK function 1654
FCCR WRITE
file control 1655
file control, FCCR WRITE function 1655
FCCT CLOSE
file control 1662
file control, FCCT CLOSE function 1662
FCCT DELETE
file control 1663
file control, FCCT DELETE function 1663
FCCT EXTRACT_STATISTICS
file control 1664
file control, FCCT EXTRACT_STATISTICS function 1664
FCCT gate
CLOSE function 124
DELETE function 125
EXTRACT_STATISTICS function 126
OPEN function 127
SET function 130
FCCT OPEN
file control 1660
file control, FCCT OPEN function 1660
FCCT SET
file control 1663
file control, FCCT SET function 1663
FCCU BACKOUT
file control 1667
file control, FCCU BACKOUT function 1667
FCCU COMMIT
file control 1666
file control, FCCU COMMIT function 1666
FCCU gate
BACKOUT function 131
COMMIT function 132
INQUIRE function 133
PREPARE function 134
RESTART function 135
RETAIN function 135
FCCU INQUIRE
file control 1667
file control, FCCU INQUIRE function 1667
FCCU PREPARE
file control 1665
file control, FCCU PREPARE function 1665
FCCU RESTART
file control 1669
file control, FCCU RESTART function 1669
FCCU RETAIN
file control 1666
file control, FCCU RETAIN function 1666
FCDN gate
CATALOG_DSNB function 136
COMMIT_DSNREFS function 137
CONNECT_DSNB function 137
DELETE_DSNB function 138
DISCONNECT_DSNB function 138
END_DSNB_BROWSE function 139
GET_NEXT_DSNB function 140
INQUIRE_DSNB function 141
RESET_ALL_QUIESCE_STATUS function 143
SET_CATALOG_RECOVERED function 144
SET_DSNB function 144
START_DSNB_BROWSE function 145
UPDATE_RECOVERY_POINTS function 145
FCDS DISCONNECT_CFDT_POOLS
file control 1670
file control, FCDS DISCONNECT_CFDT_POOLS function 1670
FCDS EXTRACT_CFDT_STATS
file control 1669
file control, FCDS EXTRACT_CFDT_STATS function 1669
FCDS gate
DISCONNECT_CFDT_POOLS function 146
EXTRACT_CFDT_STATS function 146
FCDU BACKOUT
file control 1673
file control, FCDU BACKOUT function 1673
FCDU COMMIT
file control 1672
file control, FCDU COMMIT function 1672
FCDU gate
BACKOUT function 147
COMMIT function 148
INQUIRE function 149
PREPARE function 151
RESTART function 152
RETAIN function 153
FCDU INQUIRE
file control 1674
file control, FCDU INQUIRE function 1674
FCDU PREPARE
file control 1671
file control, FCDU PREPARE function 1671
FCDU RESTART
file control 1675
file control, FCDU RESTART function 1675
FCDU RETAIN
file control 1671
file control, FCDU RETAIN function 1671
FCDY gate
RESYNC_CFDT_LINK function 153
RESYNC_CFDT_POOL function 154
RETURN_CFDT_ENTRY_POINTS function 155
FCDY RESYNC_CFDT_LINK
file control 1676
file control, FCDY RESYNC_CFDT_LINK function 1676
FCDY RESYNC_CFDT_POOL
file control 1676
file control, FCDY RESYNC_CFDT_POOL function 1676
FCDY RETURN_CFDT_ENTRY_POINTS
file control 1677
file control, FCDY RETURN_CFDT_ENTRY_POINTS function 1677
FCFL END_UOWDSN_BROWSE
file control 1677
file control, FCFL END_UOWDSN_BROWSE function 1677
FCFL FIND_RETAINED
file control 1678
file control, FCFL FIND_RETAINED function 1678
FCFL FORCE_INDOUBTS
file control 1678

FCFL FORCE_INDOUBTS *(continued)*
file control, FCFL FORCE_INDOUBTS function 1678
FCFL gate
END_UOWDSN_BROWSE function 155
FIND_RETAINED function 156
FORCE_INDOUBTS function 156
GET_NEXT_UOWDSN function 157
RESET_BFAILS function 158
RETRY function 159
START_UOWDSN_BROWSE function 159
TEST_USER function 160
FCFL GET_NEXT_UOWDSN
file control 1679
file control, FCFL GET_NEXT_UOWDSN function 1679
FCFL RESET_BFAILS
file control 1680
file control, FCFL RESET_BFAILS function 1680
FCFL RETRY
file control 1680
file control, FCFL RETRY function 1680
FCFL START_UOWDSN_BROWSE
file control 1680
file control, FCFL START_UOWDSN_BROWSE function 1680
FCFL TEST_USER
file control 1681
file control, FCFL TEST_USER function 1681
FCFR gate
CLEAR_ENVIRONMENT function 160
DELETE function 161
END_BROWSE function 163
FREE_UNUSED_BUFFERS function 165
PREPARE_FILE_REQUEST function 165
PREPARE_TO_BACKOUT function 166
READ_INTO function 166
READ_NEXT_INTO function 170
READ_NEXT_SET function 173
READ_NEXT_UPDATE_INTO function 175
READ_NEXT_UPDATE_SET function 178
READ_PREVIOUS_INTO function 181
READ_PREVIOUS_SET function 184
READ_PREVIOUS_UPDATE_INTO function 186
READ_PREVIOUS_UPDATE_SET function 189
READ_SET function 192
READ_UPDATE_INTO function 195
READ_UPDATE_SET function 198
REPLACE function 202
REPLACE_DELETE function 204
RESET_BROWSE function 206
FCFR gate *(continued)*
RESTART_FILE_CONTROL function 208
REWRITE function 209
REWRITE_DELETE function 211
START_BROWSE function 213
TEST_FILE_USER function 216
UNLOCK function 216
WRITE function 218
FCFS gate
CANCEL_CLOSE_FILE function 221
CLOSE_FILE function 221
DISABLE_FILE function 223
ENABLE_FILE function 224
OPEN_FILE function 224
FCIN gate
INITIALISE_FILE_CONTROL function 226
WAIT_FOR_FILE_CONTROL function 226
FCLJ DATASET_COPY
file control 1688
file control, FCLJ DATASET_COPY function 1688
FCLJ FILE_CLOSE
file control 1682
file control, FCLJ FILE_CLOSE function 1682
FCLJ FILE_OPEN
file control 1681
file control, FCLJ FILE_OPEN function 1681
FCLJ gate
BACKOUT_REPLICATION function 227
COMMIT_REPLICATION function 227
DATASET_COPY function 228
FILE_CLOSE function 228
FILE_OPEN function 229
READ_ONLY function 229
READ_UPDATE function 230
SYNCHRONISE_READ_UPDATE function 231
TAKE_KEYPOINT function 232
WRITE_ADD_COMPLETE function 232, 233
WRITE_DELETE function 235
WRITE_UPDATE function 235
FCLJ READ_ONLY
file control 1682
file control, FCLJ READ_ONLY function 1682
FCLJ READ_UPDATE
file control 1683
file control, FCLJ READ_UPDATE function 1683
FCLJ SYNCHRONIZE_READ_UPDATE
file control 1687
file control, FCLJ SYNCHRONIZE_READ_UPDATE function 1687
FCLJ TAKE_KEYPOINT
file control 1688
file control, FCLJ TAKE_KEYPOINT function 1688
FCLJ WRITE_ADD
file control 1685
file control, FCLJ WRITE_ADD function 1685
FCLJ WRITE_ADD_COMPLETE
file control 1686
file control, FCLJ WRITE_ADD_COMPLETE function 1686
FCLJ WRITE_DELETE
file control 1687
file control, FCLJ WRITE_DELETE function 1687
FCLJ WRITE_UPDATE
file control 1684
file control, FCLJ WRITE_UPDATE function 1684
FCMT gate
ADD_FILE function 236
COMMIT_FILES function 241
DELETE_FILE function 241
END_BROWSE_FILE function 242
GET_NEXT_FILE function 242
INQUIRE_FILE function 248
START_BROWSE_FILE function 254
UPDATE_FILE function 254
FCQI gate
COMPLETE_QUIESCE function 259
INITIATE_QUIESCE function 259
INQUIRE_QUIESCE function 260
FCQR gate
RECEIVE_QUIESCES function 261
FCQR RECEIVE_QUIESCES
file control 1689
file control, FCQR RECEIVE_QUIESCES function 1689
FCQRE (file control quiesce receive element)
file control quiesce receive element (FCQRE) 1601
FCQS gate
SEND_QUIESCES function 262
FCQS SEND_QUIESCES
file control 1689
file control, FCQS SEND_QUIESCES function 1689
FCQSE (file control quiesce send element)
file control quiesce send element (FCQSE) 1601
FCQU gate
PROCESS_QUIESCE function 262
FCQU PROCESS_QUIESCE
file control 1690
file control, FCQU PROCESS_QUIESCE function 1690
FCRF gate
BROWSE function 264
DELETE function 266
END_BROWSE function 267
READ function 268
REPLACE function 269
REPLACE_DELETE function 270
RESET_BROWSE function 271
REWRITE function 272
START_BROWSE function 273
UNLOCK function 274
WRITE function 275

FCRL gate
COMMIT_POOLS function 276
SET_POOL function 276
FCRP gate
RESTART_FILE_CONTROL function 277
FCRR gate
LOST_LOCKS_RECOVERED function 278
RESOURCE_AVAILABLE function 278
RESTART_RLS function 279
FCRR LOST_LOCKS_RECOVERED
file control 1694
file control, FCRR LOST_LOCKS_RECOVERED function 1694
FCRR RESOURCE_AVAILABLE
file control 1693
file control, FCRR RESOURCE_AVAILABLE function 1693
FCRR RESTART_RLS
file control 1692
file control, FCRR RESTART_RLS function 1692
FCSD gate
TERMINATE function 281
FCST gate
COLLECT_FILE_STATISTICS function 281
COLLECT_POOL_STATISTICS function 282
END_FILE_IN_POOL_BROWSE function 283
GET_NEXT_FILE_IN_POOL function 283
START_FILE_IN_POOL_BROWSE function 284
FCT (file control table) 1601
FCVC gate
INQUIRE_CATALOG function 285
FCXCWAIT resource type
READ request processing 1594
REWRITE request processing 1597
FEPI as a CICS transaction 1697
FEPI dumps
interpreted areas 1701
FFLE (fast file locate element) 1601
field edit built-in function 1519
field engineering program 1588
file control 1588
BDAM request processor (DFHFCBD) 1615
catalog manager (DFHFCAT) 1614
CFDT UOW pool block (FCUP) 1607
control blocks, illustrated 1601
coupling facility data table load program (DFHFCDL) 1617
coupling facility data table open/close program (DFHFCDO) 1619
coupling facility data table request processor (DFHFCDR) 1619
coupling facility data table resynchronization program (DFHFCDY) 1620
file control *(continued)*
coupling facility data table RMC program (DFHFCDW) 1620
coupling facility data table UOW calls program (DFHFCDU) 1620
data table request processor (DFHFCDTS) 1619
DELETE request 1598
DSNAME block manager (DFHFCDN) 1617
ENDBR request 1600
ENF servicer (DFHFCES) 1620
EXEC interface module (DFHEIFC) 1613
FBWA (file browse work area) 1605
FCPE (file control CFDT pool element) 1606
FCPW (file control CFDT pool wait element) 1606
FCQRE (file control quiesce receive element) 1605
FCQSE (file control quiesce send element) 1606
FCTE (file control table entry) 1607
FCUP (file control CFDT UOW pool block) 1607
file browse work area (FBWA) 1605
file control CFDT pool element (FCPE) 1606
file control CFDT pool wait element (FCPW) 1606
file control CFDT UOW pool block (FCUP) 1607
file control locks locator blocks (FLLBs) 1609
file control quiesce receive element (FCQRE) 1605
file control quiesce send element (FCQSE) 1606
file control table entry (FCTE) 1607
file lasting access block (FLAB) 1608
file request handler (DFHFCFR) 1620
file state program (DFHFCFS) 1622
FLAB (file lasting access block) 1608
FLLBs (file control locks locator blocks) 1609
FRAB and FLAB processor (DFHFCFL) 1620
function shipping interface module (DFHFCRF) 1641
initialization program 1 (DFHFCIN1) 1625
initialization program 2 (DFHFCIN2) 1626
initialize recovery (DFHFCIR) 1627
locks locator blocks (FLLBs) 1609
log failures handler (DFHFCLF) 1629
logging and journaling program (DFHFCLJ) 1629
modules, organization 1612
non-RLS lock handler (DFHFCNQ) 1636
offsite recovery completion (DFHFCOR) 1637
open/close program (DFHFCN) 1632
pool element (FCPE) 1606
pool wait element (FCPW) 1606
file control *(continued)*
quiesce receive element (FCQRE) 1605
quiesce receive transaction (DFHFCQR) 1637
quiesce send element (FCQSE) 1606
READ request 1594
READNEXT request 1599
READPREV request 1599
recovery control program (DFHFCRC) 1638
request processing 1592
RESETBR request 1599
restart program (DFHFCRP) 1643
REWRITE request 1597
RLS clean up transaction (DFHFCRD) 1640
RLS control ACB manager (DFHFCCA) 1616
RLS open/close program (DFHFCRO) 1643
RLS quiesce common system transaction (DFHFCQT) 1637
RLS quiesce exit (DFHFCQX) 1638
RLS quiesce initiation (DFHFCQI) 1637
RLS quiesce processor (DFHFCQU) 1638
RLS quiesce send transaction (DFHFCQS) 1637
RLS record management processor (DFHFCRS) 1645
RLS restart (DFHFCRR) 1645
RLS VSAM interface processor (DFHFCRV) 1645
share control block manager (DFHFCRL) 1642
shared data table function ship program (DFHFCDTX) 1620
shared resources pool processor (DFHFCL) 1627
shutdown program (DFHFCSD) 1646
STARTBR request 1599
static storage (FC static) 1605
statistics program (DFHFCST) 1646
table entry (FCTE) 1607
table manager (DFHFCMT) 1629
UNLOCK request 1598
user exits 1696
VSAM interface program (DFHFCVR) 1648
VSAM request processor (DFHFCVS) 1649
WRITE request 1595
file input/output area (FIOA) 1601, 1608
file lasting access block (FLAB) 1601
file request anchor block (FRAB) 1601, 1609
file request handler, file control (DFHFCFR) 1620
file request thread element (FRTE) 1601, 1610
file state program, file control (DFHFCFS) 1622
FILE_CLOSE function, FCLJ gate 228
FILE_OPEN function, FCLJ gate 229

files
DELETE request 1598
LSR (local shared resources) 1592
READ request 1593
FIND_HIGHEST_VERSION function, SJTH gate 1054
FIND_PROFILE function, XMFD gate 1359
FIND_RETAINED function, FCFL gate 156
FIND_SIGNATURE function, PICC gate 873
FIND_TRANSACTION_OWNER function, TFAL gate 313
FIND_URIMAP function, WBUR gate 1299
FIOA (file input/output area) 1601, 1608
FREEMAIN after REWRITE request 1597
FREEMAIN after WRITE request 1597
FREEMAIN during UNLOCK request processing 1598
FREEMAIN in ENDBR request processing 1600
READ request 1594
RESETBR request 1599
STARTBR request 1599
WRITE request 1597
FIRE_EVENT function, EMEM gate 529
FLAB (file lasting access block) 1601
FLATTEN process
FLATTEN 1511
FLATTEN_REQUEST function, APRX gate 45
FLATTEN_RESPONSE function, APRX gate 46
FLATTEN_TRANSACTION_USER function, USXM gate 1246
FLATTEN_USER function, USFL gate 1242
FLATTEN_USER_SECURITY function, XSFL gate 1424
FLLBs (file control locks locator blocks)
file control locks locator blocks (FLLBs) 1601
flow for sign-on to consoles
flow for sign-on to consoles 1456
FLUSH_LDAP_CACHE function, DDAP gate 375
FORCE function, LGGL gate 656
FORCE function, RMRE gate 976
FORCE_DATA function, LGWF gate 677
FORCE_INDOUBTS function, FCFL gate 156
FORCE_JNL function, LGGL gate 656
FORCE_LINKS function, ISRE gate 576
FORCE_PURGE_INHIBIT_QUERY function, DSAT gate 491
FORCE_PURGE_INHIBIT_QUERY function, XMPP gate 1402
FORCE_START function, PAGP gate 781
FORCE_TIMER function, EMEM gate 529
FORCE_UOW function, RMUW gate 984
FORGET function, OTCO gate 773
FORGET function, OTSU gate 777
FORGET_TRANSACTION function, OTRS gate 775
FORMAT_EVENT function, EPAS gate 535
FORMAT_TIME function, TIMF gate 1176
formats, domain call
generic 6
ownership of 6
specific 6
formatting data for function shipping 1711
forward recovery logging 1591
FRAB (file request anchor block) 1601, 1609
FRAB and FLAB processor, file control (DFHFCFL) 1620
FREE processing in application-owning region 1853
FREE processing in terminal-owning region 1857
FREE_ALL_TASK_STORAGE function, PGCH gate 790
FREE_CONVERSION_TOKEN function, CCNV gate 65
FREE_EPADAPTER function, EPAD gate 534
FREE_HANDLE_TABLES function, PGHM gate 819
FREE_POLICY function, MPDD gate 743
FREE_SEARCH_RESULTS function, DDAP gate 375
FREE_SUBSPACE_TCBS function, DSAT gate 463
FREE_TCB function, DSIT gate 475
FREE_TCBS function, KEDS gate 596
FREE_TCBS_NOTIFY_IMMED function, KEDS gate 597
FREE_TXN_ENVIRONMENT function, XMXE gate 1396
FREE_UNUSED_BUFFERS function, FCFR gate 165
FREEMAIN function, S2GF gate 1091
FREEMAIN function, SMGF gate 1070
FREEMAIN function, SMMC gate 1075
FREEMAIN_ALL_TERMINAL function, SMMC gate 1076
FRTE (file request thread element) 1601, 1610
function shipping 1706
communication with remote system 1709
data transformations 1711
distributed program link 1547
handling of CICS requests 1711
receiving a reply from a remote system 1716
receiving a request at a remote system 1715
sending a reply at a remote system 1716
sending a request to a remote system 1714
handling of DL/I requests 1717
function shipping *(continued)*
receiving a DL/I reply from a remote system 1719
receiving a DL/I request at a remote system 1719
sending a DL/I reply at a remote system 1719
sending a DL/I request to a remote system 1718
initialization 1708
local and remote resource names 1708
programming functions with 1707
protocols 1709
resynchronization protocol 1710
sender error recovery protocol 1709
shutdown protocol 1709
symmetrical bracket protocol 1709
syncpoint functions
ABORT 1720
COMMIT 1720
PREPARE 1720
SPR (syncpoint request) 1720
terminal control 1719
terminal control functions
ALLOCATE 1719
FREE 1720
POINT 1720
TERM=YES operand 1720
function shipping interface module, file control (DFHFCRF) 1641
function, FCQR gate 261
functional layout of FEPI 1697
functions of CICS, organization 1, 10
functions provided by gates 5
FWA (file work area)
FREEMAIN during REWRITE processing 1597
FREEMAIN during UNLOCK request processing 1598
FREEMAIN during WRITE request 1597
FREEMAIN in BDAM ENDBR request processing 1600
FREEMAIN in VSAM ENDBR request processing 1600
READ request 1594
RESETBR request 1599
STARTBR request 1599

G

gates, domain
functions provided by 5
generic 5
specific 5
general request processing 1593
GENERATE_APPC_BIND function, XSLU gate 1435
GENERATE_APPC_RESPONSE function, XSLU gate 1436
GENERATE_CONTENT_ID function, PIMM gate 883
generic formats 6
generic gates 5

GENERIC option of DELETE request 1598
generic resource z/OS Communications Server 1878
Generic Resource - autoinstall 1455
Generic resource and ATI 1881
GET function, CCCC gate 369
GET function, TSPT gate 1203
GET_ACD function, MNAC gate 716
GET_ASSOCIATED_DATA_LIST function, MNAD gate 720
GET_ASSOCIATED_DATA_LIST function, USAD gate 1242
GET_ATTACHMENT function, PIMM gate 884
GET_ATTRIBUTE_VALUE function, DDAP gate 376
GET_CIB function, CQCQ gate 76
GET_CLIENT_REQUEST function, RXUW gate 1021
GET_CONTAINER_INTO function, BACR gate 350
GET_CONTAINER_INTO function, PGCR gate 797
GET_CONTAINER_LENGTH function, BACR gate 351
GET_CONTAINER_LENGTH function, PGCR gate 800
GET_CONTAINER_SET function, BACR gate 352
GET_CONTAINER_SET function, PGCR gate 801
GET_CONVERSION_TOKEN function, CCNV gate 66
GET_CURRENT function, RZTA gate 1030
GET_DATA_LENGTH function, SOCK gate 1114
GET_DEBUG_DATA function, RZTA gate 1030
GET_DEBUG_PROFILE function, DPFM gate 421
GET_HTTP_RESPONSE function, WBAP gate 1253
GET_IPFACILITY_LIST function, ISIF gate 562
GET_JOIN_DATA function, RZTA gate 1030
GET_MESSAGE function, TFAL gate 313
GET_MESSAGE_BODY function, WBAP gate 1253
GET_NEXT function, AIIQ format 1466
GET_NEXT function, CCCC gate 369
GET_NEXT function, DHTM gate 402
GET_NEXT function, DMIQ gate 413
GET_NEXT function, DSBR gate 465
GET_NEXT function, LGJN gate 662
GET_NEXT function, LGLD gate 670
GET_NEXT function, LGST gate 676
GET_NEXT function, SOTB gate 1156
GET_NEXT function, TSBR gate 1196
GET_NEXT function, TSMB gate 1198
GET_NEXT function, TSSH gate 1216
GET_NEXT_APPLICATION function, LDLB gate 616
GET_NEXT_APPLICATION function, PGIS gate 829
GET_NEXT_ATOMSERVICE function, W2AT gate 1328
GET_NEXT_ATTRIBUTE function, DDAP gate 377
GET_NEXT_BUNDLE function, RLPM gate 931
GET_NEXT_BUNDLERES function, RLRO gate 937
GET_NEXT_CAPTURESPEC function, ECIS gate 80
GET_NEXT_DATA_PRED function, ECIS gate 83
GET_NEXT_DSNB function, FCDN gate 140
GET_NEXT_ENQMODEL function, NQRN gate 768
GET_NEXT_ENQUEUE function, NQIB gate 755
GET_NEXT_ENTRY function, DDAP gate 377
GET_NEXT_ENTRY function, DDBR gate 380
GET_NEXT_ENTRY function, ECIS gate 84
GET_NEXT_ENTRY function, EPIS gate 538
GET_NEXT_EVENT function, EMBR gate 521
GET_NEXT_FILE function, FCMT gate 242
GET_NEXT_FILE_IN_POOL function, FCST gate 283
GET_NEXT_HOST function, WBUR gate 1302
GET_NEXT_IN_DIRECTORY function, DHFS gate 393
GET_NEXT_INFO_SOURCE function, ECIS gate 84
GET_NEXT_INSTANCE function, LDLD gate 630
GET_NEXT_JVMSERVER function, SJJS gate 1049
GET_NEXT_LIBRARY function, LDLB gate 617
GET_NEXT_LINK function, RMLN gate 955
GET_NEXT_MVSTCB function, DSMT gate 479
GET_NEXT_OPTION_PRED function, ECIS gate 85
GET_NEXT_OSGIBUNDLE function, SJBD gate 1042
GET_NEXT_OSGISERVICE function, SJBD gate 1042
GET_NEXT_OSGISERVICE function, SJTH gate 1055
GET_NEXT_PIPELINE function, PIPL gate 891
GET_NEXT_POOL function, PTTW gate 921
GET_NEXT_PROCESSTYPE function, BATT gate 360
GET_NEXT_PROGRAM function, LDLD gate 633
GET_NEXT_PROGRAM function, PGIS gate 829
GET_NEXT_SUBPOOL function, S2AD gate 1090
GET_NEXT_SUBPOOL function, SMAD gate 1066
GET_NEXT_TCLASS function, XMCL gate 1350
GET_NEXT_TDQDEF function, TDTM gate 300
GET_NEXT_TIMER function, EMBR gate 522
GET_NEXT_TRANDEF function, XMBD gate 1344
GET_NEXT_TRANSACTION function, XMIQ gate 1360
GET_NEXT_TSPOOL function, TSSH gate 1216
GET_NEXT_TXN_TOKEN function, XMIQ gate 1365
GET_NEXT_UOW function, RMUW gate 985
GET_NEXT_UOWDSN function, FCFL gate 157
GET_NEXT_URIMAP function, WBUR gate 1303
GET_NEXT_WEBSERVICE function, PIWR gate 909
GET_NEXT_WORK_TOKEN function, RMWT gate 993
GET_NEXT_XMLTRANSFORM function, MLXT gate 714
GET_ODR function, MNOD gate 730
GET_PARAMETERS function, PAGP gate 781
GET_PROCESSED_CIB function, CQCQ gate 77
GET_PUBLIC_ID function, RZTA gate 1031
GET_RELEASE function, TSPT gate 1203
GET_RELEASE_SET function, TSPT gate 1204
GET_RESPONSE function, PITC gate 903
GET_SERVER_DATA function, RZTA gate 1031
GET_SET function, TSPT gate 1204
GET_SOCKET_FROM_POOL function, SOPL gate 1149
GET_SOCKET_OPTS function, SOCK gate 1115
GET_TXN_ENVIRONMENT function, XMXE gate 1397
GET_UPDATE function, CCCC gate 369
GET_USER_DEFAULTS function, DPUM gate 443
GETMAIN function, S2GF gate 1092
GETMAIN function, SMGF gate 1071
GETMAIN function, SMMC gate 1077
GETNEXT_ACTIVITY function, BABR gate 342
GETNEXT_CONTAINER function, BABR gate 343
GETNEXT_CONTAINER function, PGCR gate 803

GETNEXT_IPCONN function, ISIC gate 554
GETNEXT_PROCESS function, BABR gate 343
GETNEXT_SYSTEM_DUMPCODE function, DUDT gate 496
GETNEXT_TRAN_DUMPCODE function, DUDT gate 497
GL_FORCE function, LGLB gate 668
GL_WRITE function, LGLB gate 669
global user exits 1806
  XFCREQ 1593, 1595
good morning message program 1722

H

HANDLE_ATOM_REQUEST function, W2W2 gate 1333
HANDLE_PARSE_EVENT function, PICC gate 874
hash table 1794
high-performance option (HPO) 1823
HIGHEST function, FCCR gate 117
horizontal tabs
  and device independence 1471
HPO (high-performance option) 1823

I

ICE (interval control element) 1733
ICP (interval control program)
  mapping control program (MCP) 1481
  terminal page retrieval program (TPR) 1492
  undelivered messages clean up program (TPQ) 1490
ICRX_TO_USERID function, USAD gate 1241
ICUS gate
  CREATE_ICE_USER_EXTENSION function 285
  DELETE_ICE_USER_EXTENSION function 286
  DELETE_USER_TOKEN function 286
  SET_USER_TOKEN function 287
ICXM gate
  INQUIRE_FACILITY function 287
IDENTIFY_PROGRAM function, LDLD gate 635
IE (IP ECI) domain 543
IEFJSCVT 1775
IEFJSSVT 1775
IEIE gate
  PROCESS_ECI_FLOW function 543
  RECEIVE function 544
  SEND function 544
  SEND_ERROR function 545
IGNORE_CONDITIONS function, PGHM gate 820
IIP (non-3270 input mapping) 1478
  interfaces, illustrated 1478
  mapping control program (MCP) 1479, 1481
  storage control 1479
  terminal control 1479
IMMCLOSE function, SORD gate 1152
IMPLICIT_OPEN function, LGJN gate 663
IMPORT_ALL function, APCR gate 26
IMPORT_CERTIFICATE_DATA function, SOIS gate 1132
IMPORT_CHANGED function, APCR gate 27
IMPORT_SYMBOL_LIST function, DHSL gate 398
IMPORT_TRAN function, OTTR gate 779
IMS service modules
  DL/I interface 1560
in-doubts, resolution of 1808
INACTIVATE_DEBUG_PROFILE function, DPFM gate 423
INBOUND_FLOW function, RMLN gate 958
INCREMENT_USE_COUNT function, PIWR gate 910
indexes 1828
indirect transient data queues 1867
INIT_ACTIVITY_REQUEST function, BAXM gate 363
INIT_TRANSACTION_USER function, USXM gate 1247
INIT_XM_CLIENT function, APXM gate 55
INIT_XM_CLIENT function, DPXM gate 453
INIT_XM_CLIENT function, MPXM gate 751
INIT_XM_CLIENT function, XMAC gate 1398
INITIAL_LINK function, PGPG gate 863
INITIALISE function, APRD gate 39
INITIALISE function, CCNV gate 67
INITIALISE function, SMMC gate 1078
INITIALISE function, TSAD gate 1195
INITIALISE function, TSSH gate 1217
INITIALISE_DOMAIN function, DMDM gate 416
INITIALISE_FILE_CONTROL function, FCIN gate 226
INITIALISE_TRANSIENT_DATA function, APTD gate 51
INITIALISE_WEBSERVICE function, PIWR gate 911
INITIALIZE function, CQCQ gate 77
INITIALIZE function, LGGL gate 657
INITIALIZE function, LGJN gate 664
INITIALIZE function, LGLD gate 671
INITIALIZE function, LGST gate 676
initialize recovery, file control (DFHFCIR) 1627
INITIALIZE_AID_POINTERS function, TFAL gate 313
INITIALIZE_ATOMSERVICES function, W2AT gate 1329
INITIALIZE_CONNECTION function, ISCO gate 547
INITIALIZE_DOCTEMPLATES function, DHTM gate 403
INITIALIZE_ENVIRONMENT function, SOIS gate 1132
INITIALIZE_EXIT function, PGEX gate 817
INITIALIZE_RECEIVER function, ISIS gate 567
INITIALIZE_TRANSACTION function, PGXM gate 865
INITIALIZE_TRANSACTION function, WBAP gate 1254
INITIALIZE_URIMAPS function, WBUR gate 1306
INITIATE_QUIESCE function, FCQI gate 259
INITIATE_RECOVERY function, RMLN gate 958
initiation of transactions
  automatic 1867
  time ordered 1733
input TIOA
  message switching 1748
INQ_ABEND function, PGHM gate 820
INQ_AID function, PGHM gate 821
INQ_APPLICATION_DATA function, APIQ gate 29
INQ_BASEDSNAME function, FCAT gate 98
INQ_CAPTURESPEC function, ECIS gate 86
INQ_CATALOG_QUIESCESTATE function, FCAT gate 99
INQ_CATALOG_RECOV_REQD function, FCAT gate 99
INQ_CONDITION function, PGHM gate 822
INQ_DATASET_STATE function, FCAT gate 100
INQ_DOMAIN_BY_ID function, DMIQ gate 414
INQ_DOMAIN_BY_NAME function, DMIQ gate 414
INQ_DOMAIN_BY_TOKEN function, DMIQ gate 415
INQ_EPADAPTER function, EPIS gate 540
INQ_EVENTBINDING function, ECIS gate 88
INQ_EVENTPROCESS function, ECIS gate 89
INQ_LOCAL_DATETIME_DECIMAL function, KETI gate 607
INQ_MONITORING function, MNSR gate 731
INQ_SECURITY_DOMAIN_PARMS function, XSIS gate 1427
INQ_SIT_PARM function, APIQ gate 30
INQ_STATISTICS_OPTIONS function, STST gate 1167
INQ_TRANSACTION_ISOLATION function, SMSR gate 1080
INQUIRE function, FCCI gate 112
INQUIRE function, FCCU gate 133
INQUIRE function, FCDU gate 149
INQUIRE function, LGJN gate 664
INQUIRE function, LGLD gate 671
INQUIRE function, LGST gate 676
INQUIRE function, RLCB gate 943
INQUIRE function, RXUW gate 1022
INQUIRE function, SOIS gate 1133

INQUIRE function, WBAP gate 1255
INQUIRE_ABEND_RECORD function, ABAB gate 14
INQUIRE_ACCESS function, SMSR gate 1080, 1094
INQUIRE_ACCESS_TOKEN function, S2SR gate 1095
INQUIRE_ACCESS_TOKEN function, SMSR gate 1081
INQUIRE_ACTIVATION function, BABR gate 344
INQUIRE_ACTIVITY function, BABR gate 344
INQUIRE_ADCB function, MNAD gate 721
INQUIRE_ALL_TCLASSES function, XMCL gate 1351
INQUIRE_ALLOCATE_AID function, TFAL gate 314
INQUIRE_ANCHOR function, KEDD gate 584
INQUIRE_ASSOCIATED_DATA function, MNAD gate 721
INQUIRE_ATOMSERVICE function, W2AT gate 1330
INQUIRE_AUTOINSTALL function, PGAQ gate 784
INQUIRE_AUXILIARY_TRACE function, TRSR gate 1186
INQUIRE_BOUND_CHANNEL function, PGCH gate 790
INQUIRE_BROWSE_CONTEXT function, PGCR gate 804
INQUIRE_BUNDLE function, RLPM gate 933
INQUIRE_BY_NAME function, RLBR gate 945
INQUIRE_CATALOG function, FCVC gate 285
INQUIRE_CERTIFICATE function, XSCT gate 1415
INQUIRE_CERTIFICATE_USERID function, XSPW gate 1438
INQUIRE_CHANNEL function, PGCH gate 790
INQUIRE_CHANNEL_BY_TOKEN function, PGCH gate 791
INQUIRE_CLIENT_DATA function, RMCD gate 948
INQUIRE_CONNECTION function, SOIS gate 1138
INQUIRE_CONTAINER function, BABR gate 346
INQUIRE_CONTAINER function, PGCR gate 804
INQUIRE_CONTAINER_BY_TOKEN function, PGCR gate 806
INQUIRE_CONTAINER_POOL function, PGCP gate 795
INQUIRE_CONTEXT function, BRIQ gate 56
INQUIRE_CONVERSION_SIZE function, CCNV gate 69
INQUIRE_CURRENT_ACD function, MNAC gate 717
INQUIRE_CURRENT_ACD TOKEN function, MNAC gate 717
INQUIRE_CURRENT_CHANNEL function, PGCH gate 792
INQUIRE_CURRENT_DUMPDS function, DUSR gate 510
INQUIRE_CURRENT_PROGRAM function, PGIS gate 836
INQUIRE_CURRENT_SESSION function, WBSV gate 1291
INQUIRE_DATE_FORMAT function, KETI gate 608
INQUIRE_DEBUG_TASK function, DPIQ gate 431
INQUIRE_DEFAULT_USER function, USAD gate 1234
INQUIRE_DEFER_INTERVAL function, LGCC gate 651
INQUIRE_DEFERRED_ABEND function, XMER gate 1355
INQUIRE_DEFERRED_MESSAGE function, XMER gate 1356
INQUIRE_DISPATCHER function, DSIT gate 476
INQUIRE_DISTINGUISHED_NAME function, XSEJ gate 1420
INQUIRE_DOCTEMPLATE function, DHTM gate 403
INQUIRE_DOCUMENT function, DHDH gate 387
INQUIRE_DOMAIN function, USIS gate 1245
INQUIRE_DOMAIN_BY_NAME function, KEDD gate 585
INQUIRE_DOMAIN_BY_TOKEN function, KEDD gate 585
INQUIRE_DOMAIN_TRACE function, KEDD gate 586
INQUIRE_DSA_LIMIT function, SMSR gate 1082
INQUIRE_DSA_SIZE function, SMSR gate 1082
INQUIRE_DSNB function, FCDN gate 141
INQUIRE_DTRTRAN function, XMSR gate 1377
INQUIRE_DUMPDS_AUTOSWITCH function, DUSR gate 511
INQUIRE_DUMPDS_OPEN_STATUS function, DUSR gate 511
INQUIRE_ELEMENT_LENGTH function, S2GF gate 1093
INQUIRE_ELEMENT_LENGTH function, SMGF gate 1072
INQUIRE_ELEMENT_LENGTH function, SMMC gate 1079
INQUIRE_ENQMODEL function, NQRN gate 768
INQUIRE_ENQUEUE function, NQIB gate 757
INQUIRE_EVENT function, EMBR gate 523
INQUIRE_EXIT_PROGRAM function, SHRT gate 1038
INQUIRE_EXPINT function, TSMB gate 1201
INQUIRE_EXPIRATION_TOKEN function, TISR gate 1180
INQUIRE_FACILITY function, ICXM gate 287
INQUIRE_FACILITY function, ISIS gate 567
INQUIRE_FEATURE function, DUFT gate 507
INQUIRE_FILE function, FCMT gate 248
INQUIRE_GARBAGE_INTERVAL function, PTTW gate 922
INQUIRE_GLOBAL_TRACE function, KEDD gate 586
INQUIRE_GTF_TRACE function, TRSR gate 1187
INQUIRE_HASH_CODE function, XSEJ gate 1422
INQUIRE_HFS_FILE function, DHFS gate 393
INQUIRE_HOST function, WBUR gate 1307
INQUIRE_ICRX function, USAD gate 1240
INQUIRE_ICRX function, XSAD gate 1413
INQUIRE_INITIAL_ACD function, MNAC gate 718
INQUIRE_INITIAL_DUMPDS function, DUSR gate 511
INQUIRE_INTERNAL_TRACE function, TRSR gate 1188
INQUIRE_IPCONN function, ISIC gate 556
INQUIRE_IPCONN_BY_APPLID function, ISIC gate 558
INQUIRE_IPFACILITY function, ISIF gate 563
INQUIRE_ISOLATION_TOKEN function, SMSR gate 1083
INQUIRE_JSONTRANSFRM function, MLJS gate 703
INQUIRE_JVMSERVER function, SJJS gate 1050
INQUIRE_KERNEL function, KEGD gate 603
INQUIRE_KEYPOINT_FREQUENCY function, LGCC gate 651
INQUIRE_KEYPOINT_STATS function, LGCC gate 652
INQUIRE_LIBRARY function, LDLB gate 619
INQUIRE_LINK function, RMLN gate 960
INQUIRE_LOCAL_LU_NAME function, RMDM gate 950
INQUIRE_LOGNAME function, RMNM gate 971
INQUIRE_MEMLIMIT function, S2SR gate 1095
INQUIRE_MESSAGE function, MEME gate 689
INQUIRE_MESSAGE_DEFINITION function, MEBM gate 686
INQUIRE_MESSAGE_LENGTH function, MEBM gate 686
INQUIRE_MESSAGE_LENGTH function, MEME gate 689

INQUIRE_MONITOR_DATA function, TFIQ gate 326
INQUIRE_MONITORING_DATA function, MNMN gate 727
INQUIRE_MVSTCB function, DSMT gate 479
INQUIRE_MVSTCB function, KEDS gate 597
INQUIRE_MXT function, XMSR gate 1378
INQUIRE_NQRNAME function, NQRN gate 769
INQUIRE_ODR function, MNOD gate 731
INQUIRE_OPTIONS function, LDLD gate 637
INQUIRE_OSGIBUNDLE function, SJBD gate 1043
INQUIRE_OSGIBUNDLE function, SJTH gate 1055
INQUIRE_OSGIBUNDLE_BY_ID function, SJBD gate 1045
INQUIRE_OSGISERVICE function, SJBD gate 1044
INQUIRE_OSGISERVICE function, SJTH gate 1056
INQUIRE_PARAMETERS function, DPIQ gate 432
INQUIRE_PARAMETERS function, LGPA gate 673
INQUIRE_PARAMETERS function, SOIS gate 1140
INQUIRE_PASSWORD_DATA function, XSPW gate 1439
INQUIRE_PIPELINE function, PIPL gate 892
INQUIRE_POLICYSCOPE function, MPDD gate 743
INQUIRE_POOL_TOKEN function, TSSH gate 1217
INQUIRE_PREVIOUS_HOP_DATA function, MNAD gate 724
INQUIRE_PREVIOUS_TRAN_DATA function, MNAD gate 725
INQUIRE_PRINCIPAL function, XSEJ gate 1422
INQUIRE_PROCESS function, BABR gate 347
INQUIRE_PROCESSTYPE function, BATT gate 361
INQUIRE_PROGRAM function, LDLD gate 638
INQUIRE_PROGRAM function, PGIS gate 841
INQUIRE_QUEUE function, TSBR gate 1197
INQUIRE_QUEUE function, TSRM gate 1214
INQUIRE_QUEUE function, TSSH gate 1218
INQUIRE_QUIESCE function, FCQI gate 260
INQUIRE_REALM_NAME function, XSIS gate 1430
INQUIRE_RECOVERY function, FCCA gate 105
INQUIRE_REENTRANT_PROGRAM function, SMSR gate 1083
INQUIRE_REGION_USERID function, XSIS gate 1431
INQUIRE_REMOTE_TRANDEF function, XMXD gate 1384
INQUIRE_RESOURCE_DATA function, MNMN gate 728
INQUIRE_RETRY_TIME function, DUSR gate 512
INQUIRE_REVOCATION_LIST function, XSCT gate 1417
INQUIRE_RRS function, RXDM gate 1020
INQUIRE_SCOPE function, RLXM gate 940
INQUIRE_SESSION function, WBCL gate 1264
INQUIRE_SESSION function, WBSV gate 1293
INQUIRE_SHORT_ON_STORAGE function, SMSR gate 1083
INQUIRE_SOCKET_POOL function, SOPL gate 1148
INQUIRE_SOCKET_TOKEN function, SOIS gate 1142
INQUIRE_START function, PAGP gate 782, 783
INQUIRE_STARTUP function, RMDM gate 950
INQUIRE_STATISTICS function, SOIS gate 1142
INQUIRE_STATISTICS function, SOPL gate 1151
INQUIRE_STATUS function, EMEM gate 530
INQUIRE_STORAGE_PROTECT function, SMSR gate 1084
INQUIRE_SUBPOOL function, S2AD gate 1091
INQUIRE_SUBPOOL function, SMAD gate 1066
INQUIRE_SYSID_TABLE_TOKEN function, TSSH gate 1219
INQUIRE_SYSTEM function, SAIQ gate 290
INQUIRE_SYSTEM_DUMP function, DUSR gate 512
INQUIRE_SYSTEM_DUMPCODE function, DUDT gate 498
INQUIRE_TARGET_STATUS function, RSSR gate 1014
INQUIRE_TASK function, DSBR gate 467
INQUIRE_TASK_STORAGE function, SMMC gate 1079
INQUIRE_TASK_TRACE function, KEDD gate 587
INQUIRE_TCB function, DSBR gate 469
INQUIRE_TCB function, KEDS gate 597
INQUIRE_TCLASS function, XMCL gate 1351
INQUIRE_TCPIPSERVICE function, SOTB gate 1159
INQUIRE_TDQDEF function, TDTM gate 303
INQUIRE_TEMPLATE_STATUS function, DHTM gate 405
INQUIRE_TERM_MODEL function, AIIQ format 1465
INQUIRE_TERMINAL_FACILITY function, TFIQ gate 328
INQUIRE_TIME function, TIMF gate 1178
INQUIRE_TIMER function, EMBR gate 523
INQUIRE_TRACE_FLAGS function, EJDU gate 520
INQUIRE_TRACKING_DATA function 736
INQUIRE_TRACKING_DATA function, MNTD gate 736
INQUIRE_TRAN_DATA_FACILITY function, TDXM gate 309
INQUIRE_TRAN_DUMPCODE function, DUDT gate 499
INQUIRE_TRANDEF function, XMXD gate 1389
INQUIRE_TRANSACTION function, XMIQ gate 1365
INQUIRE_TRANSACTION_TOKEN function, XMIQ gate 1370
INQUIRE_TRANSACTION_USER function, USXM gate 1248
INQUIRE_TSMODEL function, TSMB gate 1200
INQUIRE_TSPOOL function, TSSH gate 1219
INQUIRE_UOW function, RMUW gate 987
INQUIRE_UOW_ID function, RMUW gate 990
INQUIRE_UOW_TOKEN function, RMUW gate 991
INQUIRE_URIMAP function, WBUR gate 1308
INQUIRE_USER function, USAD gate 1236
INQUIRE_USER_ATTRIBUTES function, XSAD gate 1409
INQUIRE_USER_TOKEN function, PTTW gate 922
INQUIRE_WEBSERVICE function, PIWR gate 911
INQUIRE_WORK_TOKEN function, RMWT gate 994
INQUIRE_XMLTRANSFORM function, MLXT gate 712
INSERT_BOOKMARK function, DHDH gate 388
INSERT_DATA function, DHDH gate 388
INSERT_LINK function, RMLN gate 963
install flow for autoinstall of consoles
  install flow of console autoinstall 1455
INSTALL function, LGLD gate 671
INSTALL_BUNDLE function, RLPM gate 934
INSTALL_JSONTRANSFRM function, MLJS gate 704
INSTALL_OSGIBUNDLE function, SJTH gate 1057

INSTALL_XMLTRANSFORM function, MLXT gate 711
install, CEDA 1517
inter-system 546
intercommunication facilities
  distributed program link 1547
  distributed transaction processing 1549
  transaction routing 1846
INTERPRET_ENQUEUE function, NQNQ gate 762
interregion communication (IRC) 1722, 1750
interval control 1732
interval control element (ICE) 1733
intrapartition transient data queues 1866
  interfaces, illustrated 1868
  recovery of 1866
  referencing using indirect queues 1867
INVALID_TRACKING_TAG 735
INVOKE_JAVA_PROGRAM function, SJTH gate 1058
INVOKE_PROGRAM function, PIPM gate 895
INVOKE_PTHREAD function, LEPT gate 288
INVOKE_STUB function, PIPM gate 896
INVOKE_USER_EXIT function, APEX gate 28
INVOKE_WEBSERVICE function, PIIW gate 876
IOINFO area 1737, 1743
IP ECI domain (IE) 543
IPIC_ROUTE_TRANSACTION function, APRR gate 44
IRC (interregion communication) 1722
  CICS address space modules 1729
    DFHCRC 1731
    DFHCRNP 1729
    DFHCRR 1731
    DFHZCP 1732
    DFHZCX 1731
    DFHZIS2 1731
  control blocks, illustrated 1723
  delay-queue 1730
  input data stream 1730
  interregion program (DFHIRP) 1728
  new conversation 1730
  quiesce 1730
IS 546
IS_ACD_SET function, MNAC gate 718
ISC (intersystem communication) 1732
  secondary half session support 1720
ISCO gate
  ACQUIRE_CONNECTION function 546
  INITIALIZE_CONNECTION function 547
  RELEASE_CONNECTION function 548
  TERMINATE_CONNECTION function 549
ISIC gate
  ADD_IPCONN function 550
  AUTOINSTALL_IPCONN function 552
ISIC gate *(continued)*
  DISCARD_IPCONN function 553
  ENDBROWSE_IPCONN function 553
  GETNEXT_IPCONN function 554
  INQUIRE_IPCONN function 556
  INQUIRE_IPCONN_BY_APPLID function 558
  SET_IPCONN function 560
  STARTBROWSE_IPCONN function 562
ISIF gate
  GET_IPFACILITY_LIST function 562
  INQUIRE_IPFACILITY function 563
ISIS gate
  ALLOCATE_SEND function 563
  BIND_RECEIVER function 564
  CONVERSE function 564
  INITIALIZE_RECEIVER function 567
  INQUIRE_FACILITY function 567
  RECEIVE_BUFFER function 568
  RECEIVE_REQUEST function 569
  RETRIEVE_ICRX function 571
  ROUTING_CONVERSE function 571
  SEND_BUFFER function 572
  SEND_ERROR function 573
  SEND_RESPONSE function 574
  SET_PARAMETERS function 575
ISRE gate
  CICS_RESYNC function 576
  FORCE_LINKS function 576
  KEEP_LINKS function 577
  RESYNC_LINKS function 577
  XA_RESYNC function 578
ISRR gate
  NOTIFY function 578
  NOTIFY_SERVICE function 579
  PROCESS_ERROR_QUEUE function 580
  PROCESS_INPUT_QUEUE function 580
  TERMINATE_INPUT function 581
ISSUE function, PITC gate 901
ISSUE_PREPARE function, RMLN gate 963

J

JES 1775
JOIN function, RZSO gate 1027
journaling, automatic 1591
JVM domain (SJ) domain 1041
JVMSERVER_STATISTICS function, SJTH gate 1059

K

KCP (task control program)
  deferred work element (DWE) 1480
  mapping control program (MCP) 1480, 1481
  terminal control 1818, 1819
  terminal page retrieval program (TPR) 1492
KE (Kernel domain) domain 582
KEAR gate
  DEREGISTER function 582
  READY function 583
  REGISTER function 583
  WAITPRED function 583
KEDD gate
  ADD_DOMAIN function 583
  ADD_GATE function 584
  DELETE_GATE function 584
  INQUIRE_ANCHOR function 584
  INQUIRE_DOMAIN_BY_NAME function 585
  INQUIRE_DOMAIN_BY_TOKEN function 585
  INQUIRE_DOMAIN_TRACE function 586
  INQUIRE_GLOBAL_TRACE function 586
  INQUIRE_TASK_TRACE function 587
  PERFORM_SYSTEM_ACTION function 588
  SET_ANCHOR function 588
  SET_DEFAULT_RECOVERY function 589
  SET_DOMAIN_TRACE function 589
  SET_GLOBAL_TRACE function 590
  SET_TASK_TRACE function 590
  SET_TRAP_OFF function 591
  SET_TRAP_ON function 591
KEDS gate
  ABNORMALLY_TERMINATE_TASK function 592
  ADD_CRITICAL_MODULE function 592
  ADD_CRITICAL_WINDOW function 593
  CREATE_TASK function 593
  CREATE_TCB function 594
  DETACH_TERMINATED_OWN_TCBS function 595
  END_TASK function 596
  FREE_TCBS function 596
  FREE_TCBS_NOTIFY_IMMED function 597
  INQUIRE_MVSTCB function 597
  INQUIRE_TCB function 597
  POP_TASK function 598
  PROCESS_KETA_ERROR function 598
  PUSH_TASK function 598
  RESTORE_STIMER function 599
  SEND_DEFERRED_ABEND function 600
  START_FORCE_PURGE_PROTECT function 601
  START_PURGE_PROTECTION function 601
  START_RUNAWAY_TIMER function 601
  STOP_FORCE_PURGE_PROTECT function 602
  STOP_PURGE_PROTECTION function 602
  STOP_RUNAWAY_TIMER function 602
  TASK_REPLY function 609

KEDS gate *(continued)*
TCB_REPLY function 610
KEEP_LINKS function, ISRE gate 577
KEGD gate
INQUIRE_KERNEL function 603
SET_KERNEL function 604
kernel linkage routines 1
KerneldDomain (KE) domain 582
KETI gate
ADJUST_STCK_TO_LOCAL function 606
CONVERT_TO_DECIMAL_TIME function 606
CONVERT_TO_STCK_FORMAT function 607
INQ_LOCAL_DATETIME_DECIMAL function 607
INQUIRE_DATE_FORMAT function 608
NOTIFY_RESET function 611
REQUEST_NOTIFY_OF_A_RESET function 608
RESET_LOCAL_TIME function 608
SET_DATE_FORMAT function 608
KEXM gate
TRANSACTION_INITIALISATION function 609
keypoint list element (KPLE) block 1611
KEYPOINT_DATA function, RMRE gate 977
KILL_JVMSERVER function, SJTH gate 1059
KPLE (keypoint list element) block 1611
KSDS (key-sequenced data set)
GENERIC option of DELETE request 1598
WRITE request for KSDS file 1596

L

LACB (logon address control block) 1726
Language Environment interface 1734
establishing connection 1736
function calls 1735
parameter lists 1739
storage acquisition 1738
storage for transaction 1737
LCB (logon control block) 1726
LCBE (logon control block entry) 1726
LD (Loader domain) domain 612
LDLB gate
ADD_REPLACE_LIBRARY function 612
DISCARD_LIBRARY function 615
END_BROWSE_LIBRARY function 616
GET_NEXT_APPLICATION function 616
GET_NEXT_LIBRARY function 617
INQUIRE_LIBRARY function 619
LOG_LIBRARY_ORDER function 622
SET_LIBRARY function 622
START_BROWSE_LIBRARY function 623
LDLD gate
ACQUIRE_PROGRAM function 624
CATALOG_PROGRAMS function 626
CONVERT_NAME function 626
DEFINE_PROGRAM function 627
DELETE_PROGRAM function 629
END_BROWSE function 630
GET_NEXT_INSTANCE function 630
GET_NEXT_PROGRAM function 633
IDENTIFY_PROGRAM function 635
INQUIRE_OPTIONS function 637
INQUIRE_PROGRAM function 638
REFRESH_PROGRAM function 640
RELEASE_PROGRAM function 641
SET_OPTIONS function 642
START_BROWSE function 643
LEAVE function, RZSO gate 1027
LEPT gate
CREATE_LE_ENCLAVE function 288
CREATE_PTHREAD function 288
INVOKE_PTHREAD function 288
PTHREAD_REPLY function 289
TERMINATE_LE_ENCLAVE function 289
TERMINATE_PTHREAD function 289
LG (Logger Domain) domain 645
LGBA gate
BROWSE_ALL_GET_NEXT function 646
END_BROWSE_ALL function 646
START_BROWSE_ALL function 647
LGCB gate
CHAIN_BROWSE_GET_NEXT function 647
END_CHAIN_BROWSE function 648
START_CHAIN_BROWSE function 648
LGCC gate
BROWSE_CHAINS_GET_NEXT function 648
CREATE_CHAIN_TOKEN function 649
DELETE_ALL function 649
DELETE_HISTORY function 650
END_BROWSE_CHAINS function 650
INQUIRE_DEFER_INTERVAL function 651
INQUIRE_KEYPOINT_FREQUENCY function 651
INQUIRE_KEYPOINT_STATS function 652
RELEASE_CHAIN_TOKEN function 652
RESET_KEYPOINT_STATS function 652
RESTORE_CHAIN_TOKEN function 653
SET_DEFER_INTERVAL function 653
SET_HISTORY function 654
SET_KEYPOINT_FREQUENCY function 654
START_BROWSE_CHAINS function 655
SYSINI function 655
LGGL gate
CLOSE function 656
ERROR function 680
FORCE function 656
FORCE_JNL function 656
INITIALIZE function 657
OPEN function 657
UOW_TIME function 658
WRITE function 658
WRITE_JNL function 659
LGJN gate
DISCARD function 660
END_BROWSE function 660
EXPLICIT_OPEN function 661
GET_NEXT function 662
IMPLICIT_OPEN function 663
INITIALIZE function 664
INQUIRE function 664
PROCESS_STATISTICS function 665
SET function 665
START_BROWSE function 666
STREAM_FAIL function 666
LGLB gate
CONNECT function 666
DISCONNECT function 667
DISCONNECT_ALL function 668
GL_FORCE function 668
GL_WRITE function 669
LGLD gate
DISCARD function 669
END_BROWSE function 670
GET_NEXT function 670
INITIALIZE function 671
INQUIRE function 671
INSTALL function 671
MATCH function 672
START_BROWSE function 672
LGMV gate
MOVE_CHAIN function 672
LGPA gate
INQUIRE_PARAMETERS function 673
SET_PARAMETERS function 673
LGSR gate
LOGSTREAM_STATS function 674
LGST gate
CONNECT function 674
DISCONNECT function 675
END_BROWSE function 675
GET_NEXT function 676
INITIALIZE function 676
INQUIRE function 676
START_BROWSE function 677
LGWF gate
FORCE_DATA function 677
WRITE function 678
limited resources, LU6.2 1888
LINK function, PGLK gate 859
LINK_ACTIVITY function, BAAC gate 338
LINK_EXEC function, PGLE gate 857
LINK_PLT function, PGLK gate 860
LINK_PROCESS function, BAPR gate 356
LINK_URM function, PGLU gate 861
linkage routines, kernel 1
LISTEN function, APTC gate 48

LISTEN function, DMEN gate 412
LISTEN function, SOCK gate 1117
LISTEN function, SOLS gate 1147
LM (Lock manager domain) domain 683
LMLM gate
  ADD_LOCK function 683
  DELETE_LOCK function 683
  LOCK function 683
  TEST_LOCK_OWNER function 684
  UNLOCK function 684
LOAD function, FCCR gate 118
LOAD function, PGLD gate 854
LOAD_EXEC function, PGLD gate 855
Loader domain (LD) domain 612
local resource names in function shipping 1708
local shared resources (LSR) 1592, 1603
LOCATE function, DDLO gate 383
LOCATE_AID function, TFAL gate 314
LOCATE_AND_LOCK_TCLASS function, XMCL gate 1352
LOCATE_AND_LOCK_TRANDEF function, XMLD gate 1376
LOCATE_REMDEL_AID function, TFAL gate 315
LOCATE_SHIPPABLE_AID function, TFAL gate 315
LOCATE_TERM_MODEL function, AIIQ format 1464
LOCATE_URIMAP function, WBUR gate 1312
LOCK function, LMLM gate 683
Lock manager domain (LM) domain 683
locking update model 1590
locks 1829
log failures handler, file control (DFHFCLF) 1629
LOG_LIBRARY_ORDER function, LDLB gate 622
Logger Domain (LG) domain 645
logging and journaling program, file control (DFHFCLJ) 1629
logging, automatic 1591
logical unit type 6.1 (LU6.1) protocols
  distributed transaction processing 1549
  function shipping 1709
logical unit type 6.2 (LU6.2) 1885
logon address control block (LACB) 1726
logon control block (LCB) 1726
logon control block entry (LCBE) 1726
LOGSTREAM_STATS function, LGSR gate 674
LOST_LOCKS_COMPLETE function, FCCA gate 106
LOST_LOCKS_RECOVERED function, FCRR gate 278
lower objects first, rule 1503
LSR (local shared resources) 1592, 1603
LU-initiated autoinstall disconnection flow 1456
LU6.1 1720
LU6.2
  class of service 1885
  command processing in application-owning region 1853
LU6.2 *(continued)*
  command processing in terminal-owning region 1857
  daisy chaining 1854
  exchange log name (XLN) 1887
  limited resources 1888
  modules 1888
  RECEIVE processing 1887
  SEND processing 1887
  session management 1885
  session states 1887
  transaction routing for APPC devices 1865
  z/OS Communications Server 1885

M

M32 (3270 mapping) 1483
  interfaces, illustrated 1483
  mapping control program (MCP) 1481, 1483
  page and text build (PBP) 1483
  storage control 1484
  terminal control 1484
  terminal input/output area (TIOA) 1484
  terminal page processor (TPP) 1484
MAKE_HFS_DIRECTORY function, DHFS gate 394
MAKE_PARTNERSHIP function, PTTW gate 923
Managed platform domain (MP) 740
MARK_THREAD_DELETED function, SJJS gate 1051
Markup language (ML) domain 703
MASSINSERT option of WRITE request 1596
MATCH function, LGLD gate 672
MATCH function, TSMB gate 1201
MATCH_TASK_TO_AID function, TFAL gate 315
MCB (message control block) 1472
MCP (mapping control program) 1479
  3270 mapping (M32) 1481, 1483
  application programs 1480
  BMS fast-path module (MCX) 1481
  interfaces, illustrated 1479
  interval control 1481
  LU1 printer with extended attributes mapping program (ML1) 1482
  non-3270 input mapping (IIP) 1479, 1481
  page and text build (PBP) 1481, 1485
  partition handling program (PHP) 1482, 1487
  program manager 1481
  route list resolution program (RLR) 1481, 1488
  storage control 1481
  task control 1480, 1481
  temporary-storage control 1480
  terminal page retrieval program (TPR) 1492
  transient data control 1481
  undelivered messages clean up program (TPQ) 1490
MCX (BMS fast path module)
  mapping control program (MCP) 1481
ME (Message domain) domain 686
MEBM gate
  INQUIRE_MESSAGE_DEFINITION function 686
  INQUIRE_MESSAGE_LENGTH function 686
  RETRIEVE_MESSAGE function 687
MEM4 gate
  CONVERSE function 696
  RETRIEVE_MESSAGE function 697
  SEND_MESSAGE function 699
MEME gate
  CONVERSE function 688
  INQUIRE_MESSAGE function 689
  INQUIRE_MESSAGE_LENGTH function 689
  RETRIEVE_MESSAGE function 691
  SEND_MESSAGE function 692
  VALIDATE_LANGUAGE_CODE function 694
  VALIDATE_LANGUAGE_SUFFIX function 695
MERGE_CIB_QUEUES function, CQCQ gate 77
MESR gate
  SET_MESSAGE_OPTIONS function 701
message control block (MCB) 1472
Message domain (ME) domain 686
message generation program (MGP) 1746
message routing 1470
message switching 1747
  BMS 1749
  input TIOA 1748
  program control 1748
  ROUTE operand 1748
  storage control 1748
  task control area (TCA) 1749
  temporary-storage control 1748
  terminal list table (TLT) 1748
messages
  DFHIR3762 1730
MGP (message generation program) 1746
mirror transaction 1708
ML (Markup language domain) domain 703
ML domain
  modules 716
ML1 (LU1 printer with extended attributes mapping program) 1482
  interfaces, illustrated 1482
  mapping control program (MCP) 1482
  page and text build (PBP) 1482
  storage control 1483
  terminal input/output area (TIOA) 1483
  terminal page processor (TPP) 1483
MLJS gate
  DISCARD_JSONTRANSFRM function 703

MLJS gate *(continued)*
INQUIRE_JSONTRANSFRM function 703
INSTALL_JSONTRANSFRM function 704
SET_JSONTRANSFRM function 704
TRANSFORM_JSONTRANSFRM function 705
MLPC gate
PARSE_CONTAINER function 705
MLTF gate
PARSE_XSDBIND_FILE function 706
QUERY_XML function 707
RELEASE_XSDBIND function 708
TRANSFORM_STRUCTURE_TO_XML function 708
TRANSFORM_XML_TO_STRUCTURE function 709
MLXT gate
DISCARD_XMLTRANSFORM function 712
END_BROWSE_XMLTRANSFORM function 716
GET_NEXT_XMLTRANSFORM function 714
INQUIRE_XMLTRANSFORM function 712
INSTALL_XMLTRANSFORM function 711
SET_XMLTRANSFORM function 713
START_BROWSE_XMLTRANSFORM function 714
MN (Monitoring domain) domain 716
MNAC gate
GET_ACD function 716
INQUIRE_CURRENT_ACD function 717
INQUIRE_CURRENT_ACD TOKEN function 717
INQUIRE_INITIAL_ACD function 718
IS_ACD_SET function 718
POP_ACD function 719
PUSH_ACD function 719
TRANSACTION_BIND function 720
TRANSACTION_TERMINATION function 720
MNAD gate
GET_ASSOCIATED_DATA_LIST function 720
INQUIRE_ADCB function 721
INQUIRE_ASSOCIATED_DATA function 721
INQUIRE_PREVIOUS_HOP_DATA function 724
INQUIRE_PREVIOUS_TRAN_DATA function 725
SET_APP_CONTEXT function 726
MNMN gate
ACCUMULATE_RMI_TIME function 726
EXCEPTION_DATA_PUT function 727
INQUIRE_MONITORING_DATA function 727
INQUIRE_RESOURCE_DATA function 728
MNMN gate *(continued)*
MONITOR function 728
PERFORMANCE_DATA_PUT function 729
MNOD gate
DECODE_ODR function 729
ENCODE_ODR function 730
GET_ODR function 730
INQUIRE_ODR function 731
SET_UCD function 731
MNPS 1895, 1897
MNSR gate
INQ_MONITORING function 731
SET_MCT_SUFFIX function 733
SET_MONITORING function 733
MNTD gate 735, 736
MNXM gate
TRANSACTION_BIND function 736
TRANSACTION_INITIALISATION function 737
TRANSACTION_TERMINATION function 737
MODIFY_WSBIND_FOR_JSON function, PIJG gate 878
modules
AS domain 10
EP domain 543
ML domain 716
MP domain 752
PI domain 916
RL domain 947
RS domain 1020
modules list 1913
MONITOR function, MNMN gate 728
Monitoring domain (MN) 716
MOVE_CHAIN function, LGMV gate 672
MOVE_CONTAINER function, BACR gate 352
MOVE_CONTAINER function, PGCR gate 807
MP (managed platform) domain 740
MP domain
modules 752
MPAC gate
PERFORM_ACTION function 740
MPDD gate
ADD_POLICY function 741
ADD_POLICYSCOPE function 741
DISCARD_POLICY function 742
DISCARD_POLICYSCOPE function 742
FREE_POLICY function 743
INQUIRE_POLICYSCOPE function 743
MPIB gate
END_BROWSE_POLICY_RULE function 744
GET_NEXT_POLICY_RULE function 744
START_BROWSE_POLICY_RULE function 748
MPIB_END_BROWSE_POLICY_RULE function, MPIB gate 744
MPIB_GET_NEXT_POLICY_RULE function, MPIB gate 744
MPMO gate
DEREGISTER_POLICY function 749
REGISTER_POLICY function 749
SELECT_MODEL function 750
MPXM gate
BIND_XM_CLIENT function 750
INIT_XM_CLIENT function 751
RELEASE_XM_CLIENT function 751
MRO (multiregion operation) 1749
interregion communication 1722, 1750
multinode persistent sessions 1895, 1897
multiregion operation (MRO) 1749
MVS image
MRO links between images, in a sysplex 1723
MVS_STORAGE_NOTIFY function, SMNT gate 1097
MXT_CHANGE_NOTIFY function, XMNT gate 1401
MXT_NOTIFY function, XMNT gate 1401

N

NACP (node abnormal condition program) 1750, 1816
terminal control 1820
z/OS Communications Server 1750
NEP (node error program) 1754, 1816
skeleton sample 1754
terminal control 1820
user-written 1754
z/OS Communications Server 1754
NIB (node initialization block) 1494, 1703, 1828
node initialization block (NIB) 1494, 1703, 1828
non-RLS lock handler, file control (DFHFCNQ) 1636
NOPS 1895
NOTIFY function, ISRR gate 578
NOTIFY function, RLRO gate 938
NOTIFY function, TISR gate 1183
NOTIFY_DELETE_TCB function, DSAT gate 492
NOTIFY_REFRESH function, APLX gate 37
NOTIFY_RESET function, KETI gate 611
NOTIFY_SERVICE function, ISRR gate 579
NOTIFY_USERID function, USAD gate 1238
NQ (Enqueue domain) domain 752
NQED gate
DEQUEUE function 752
ENQUEUE function 753
NQIB gate
END_BROWSE_ENQUEUE function 755
GET_NEXT_ENQUEUE function 755
INQUIRE_ENQUEUE function 757
START_BROWSE_ENQUEUE function 758
NQNQ gate
CREATE_ENQUEUE_POOL function 759

NQNQ gate *(continued)*
DEACTIVATE function 762
DEQUEUE_TASK function 762
INTERPRET_ENQUEUE function 762
REACQUIRE_ENQUEUE function 764
SET_NQRNAME_LIST function 765
NQRN gate
ADD_REPLACE_ENQMODEL function 765
COMMIT_ENQMODEL function 767
DISCARD_ENQMODEL function 767
END_BROWSE_ENQMODEL function 767
GET_NEXT_ENQMODEL function 768
INQUIRE_ENQMODEL function 768
INQUIRE_NQRNAME function 769
REMOVE_ENQMODEL function 770
RESTORE_DIRECTORY function 770
SET_ENQMODEL function 770
START_BROWSE_ENQMODEL function 771

O

object code only (OCO) 4
object transaction service domain (OT) 773
OCO (object code only) 4
OCO components 4
offsite recovery completion, file control (DFHFCOR) 1637
OPEN function, APTC gate 48
OPEN function, FCCT gate 127
OPEN function, LGGL gate 657
OPEN_FILE function, FCFS gate 224
OPEN_SESSION function, WBCL gate 1266
OPEN_TRANSIENT_DATA function, TDOC gate 294
open/close program, file control (DFHFCN) 1632
operator error
abnormal condition program (ACP) 1843
OSPWA (output services processor work area) 1472
partition handling program (PHP) 1487
OT (object transaction service) domain 773
OTCO gate
FORGET function 773
RESYNC function 773
SET_COORDINATOR function 774
SET_LAST_AGENT function 774
OTCP gate
RESYNC_COORDINATOR function 774
RESYNC_SUBORDINATE function 775
OTRS gate
FORGET_TRANSACTION function 775
OTRS gate *(continued)*
PERFORM_RESYNC function 775
SET_REMOTE_STATUS function 776
OTSU gate
ADD_SUBORDINATE function 776
FORGET function 777
RESYNC function 777
SET_VOTE function 777
OTTR gate
BEGIN_TRAN function 778
COMMIT function 778
COMMIT_ONE_PHASE function 779
IMPORT_TRAN function 779
PREPARE function 780
ROLLBACK function 780
SET_ROLLBACK_ONLY function 780
ownership of formats 6

P

PA (Parameter manager domain) domain 781
PAGP gate
FORCE_START function 781
GET_PARAMETERS function 781
INQUIRE_START function 782, 783
parallel sessions
allocation 1720
Parameter manager domain (PA) domain 781
PARSE_CONTAINER function, MLPC gate 705
PARSE_CONTENT_TYPE function, PIMM gate 885
PARSE_ICM function, PIII gate 875
PARSE_MIME_HEADERS function, PIMM gate 886
PARSE_MIME_MESSAGE function, PIMM gate 887
PARSE_MULTIPART_FORM function, WBFM gate 1280
PARSE_MULTIPART_RELATED function, PIMM gate 888
PARSE_URL function, WBCL gate 1268
PARSE_URL_ENCODED_FORM function, WBFM gate 1281
PARSE_URL_ENCODED_LIST function, WBFM gate 1282
PARSE_XOP function, PIXI gate 914
PARSE_XSDBIND_FILE function, MLTF gate 706
Partner management domain (PT) domain 917
partner resource manager 1763, 1765
paths
REWRITE processing 1597
pattern structure 1501
PATTERN_MATCH_PROFILE function, DPPM gate 440
PATTERN_MATCH_TASK function, DPPM gate 443
patterns 1501, 1514
patterns and subpatterns 1501
PAUSE_AUXILIARY_TRACE function, TRSR gate 1188
PBP (page and text build) 1484
PBP (page and text build) *(continued)*
3270 mapping (M32) 1483
data stream build (DSB) 1478
interfaces, illustrated 1484
LU1 printer with extended attributes mapping program (ML1) 1482
mapping control program (MCP) 1481, 1485
program manager 1486
storage control 1486
PCP (program control program) 1755
message switching 1748
PEEK function, WBSR gate 1284
PEEK_HEADERS function, WBSV gate 1290
PEP (program error program) 1758
transaction failure program (TFP) 1843
PERFORM_ACTION function, MPAC gate 740
PERFORM_COMMIT function, RMLK gate 1003
PERFORM_COMMIT function, RMRO gate 998
PERFORM_JOIN function, RZRJ gate 1024
PERFORM_JSON_PARSE function, PIJP gate 878
PERFORM_PIPELINE function, PIPL gate 893
PERFORM_PRELOGGING function, RMLK gate 1005
PERFORM_PREPARE function, RMLK gate 1005
PERFORM_PREPARE function, RMRO gate 999
PERFORM_REGULAR_DREDGE function, SHRQ gate 1036
PERFORM_RESTART_DREDGE function, SHRQ gate 1036
PERFORM_RESYNC function, OTRS gate 775
PERFORM_RESYNC function, PIRE gate 898
PERFORM_SHUNT function, RMLK gate 1006
PERFORM_SHUNT function, RMRO gate 999
PERFORM_SHUTDOWN function, SHRQ gate 1036
PERFORM_SYSTEM_ACTION function, KEDD gate 588
PERFORM_UNSHUNT function, RMCD gate 1000
PERFORM_UNSHUNT function, RMLK gate 1007
PERFORM_XML_PARSE function, PICC gate 874
PERFORMANCE_DATA_PUT function, MNMN gate 729
persistent sessions 1895, 1897
diagnosing problems 1905
sessions unbind during restart 1905
persistent sessions delay interval 1897
persistent sessions restart flow 1898
PG (Program manager domain) domain 784

PGAQ gate
INQUIRE_AUTOINSTALL function 784
SET_AUTOINSTALL function 785
SET_SYSTEM function 786
PGCH gate
ADD_CHANNEL function 786
BIND_CHANNEL function 787
COPY_CHANNEL function 787
CREATE_CHANNEL function 787
DELETE_CHANNEL function 788
DELETE_OWNED_CHANNELS function 789
DETACH_CHANNEL function 789
FREE_ALL_TASK_STORAGE function 790
INQUIRE_BOUND_CHANNEL function 790
INQUIRE_CHANNEL function 790
INQUIRE_CHANNEL_BY_TOKEN function 791
INQUIRE_CURRENT_CHANNEL function 792
RENAME_CHANNEL function 793
SET_CURRENT_CHANNEL function 793
PGCP gate
COPY_CONTAINER_POOL function 794
CREATE_CONTAINER_POOL function 794
DELETE_CONTAINER_POOL function 794
INQUIRE_CONTAINER_POOL function 795
PGCR gate
COPY_CONTAINER function 795
DELETE_CONTAINER function 796
ENDBR_CONTAINER function 797
GET_CONTAINER_INTO function 797
GET_CONTAINER_LENGTH function 800
GET_CONTAINER_SET function 801
GETNEXT_CONTAINER function 803
INQUIRE_BROWSE_CONTEXT function 804
INQUIRE_CONTAINER function 804
INQUIRE_CONTAINER_BY_TOKEN function 806
MOVE_CONTAINER function 807
PUT_CONTAINER function 808
SET_CONTAINER function 810
STARTBR_CONTAINER function 811
TRACE_CONTAINERS function 811
PGDD gate
ADOPT_PROGRAM function 812
DEFINE_PROGRAM function 812
DELETE_PROGRAM function 817
PGEX gate
INITIALIZE_EXIT function 817
TERMINATE_EXIT function 818
PGHM gate
CLEAR_LABELS function 819
FREE_HANDLE_TABLES function 819
PGHM gate *(continued)*
IGNORE_CONDITIONS function 820
INQ_ABEND function 820
INQ_AID function 821
INQ_CONDITION function 822
POP_HANDLE function 824
PUSH_HANDLE function 824
SET_ABEND function 824
SET_AIDS function 826
SET_CONDITIONS function 827
PGIS gate
END_BROWSE_APPLICATION function 828
END_BROWSE_PROGRAM function 828
GET_NEXT_APPLICATION function 829
GET_NEXT_PROGRAM function 829
INQUIRE_CURRENT_PROGRAM function 836
INQUIRE_PROGRAM function 841
REFRESH_PROGRAM function 848
SET_PROGRAM function 849
START_BROWSE_APPLICATION function 852
START_BROWSE_PROGRAM function 853
PGLD gate
LOAD function 854
LOAD_EXEC function 855
RELEASE function 856
RELEASE_EXEC function 857
PGLE gate
LINK_EXEC function 857
PGLK gate
LINK function 859
LINK_PLT function 860
PGLU gate
LINK_URM function 861
PGMINFO1 area 1737, 1744
PGMINFO2 area 1744
PGPG gate
INITIAL_LINK function 863
PGRE gate
PREPARE_RETURN_EXEC function 863
PGXE gate
PREPARE_XCTL_EXEC function 864
PGXM gate
INITIALIZE_TRANSACTION function 865
TERMINATE_TRANSACTION function 866
phonetic conversion subroutine 1519
PHP (partition handling program) 1486
interfaces, illustrated 1486
mapping control program (MCP) 1482, 1487
output services processor work area (OSPWA) 1487
program manager 1487
storage control 1487
terminal control table terminal entry (TCTTE) 1487
terminal output macro (TOM) 1487
PHP (partition handling program) *(continued)*
terminal partition extension (TPE) 1487
PHPPIN 1486
PHPPSC 1486
PHPPSI 1486
PHPPXE 1486
PI (Pipeline manager domain) domain 868
PI domain
modules 916
PIAT gate
CREATE_CONTEXT function 869
CREATE_CONTEXT_RESP function 869
CREATE_NON_TERMINAL_MSG function 870
CREATE_REGISTER_REQUEST function 870
CREATE_REGISTER_RESP function 871
CREATE_TERMINAL_MSG function 871
PROCESS_CONTEXT function 872
PROCESS_CONTEXT_RESP function 872
PROCESS_MSG function 873
PICC gate
FIND_SIGNATURE function 873
HANDLE_PARSE_EVENT function 874
PERFORM_XML_PARSE function 874
PIII gate
PARSE_ICM function 875
PIIW gate
INVOKE_WEBSERVICE function 876
PIJG gate
CONVERT_DATA_TO_JSON function 877
MODIFY_WSBIND_FOR_JSON function 878
PIJP gate
PERFORM_JSON_PARSE function 878
PIJS gate
PROCESS_JSON_REQUEST function 879
PIMM gate
BUILD_CONTENT_TYPE function 879
BUILD_MIME_HEADERS function 880
BUILD_MIME_MESSAGE function 881
BUILD_MULTIPART_RELATED function 881
BUILD_XOP function 915
CONVERT_CID_TO_CONTENT_ID function 882
CONVERT_CONTENT_ID_TO_CID function 883
DELETE_ATTACHMENTS function 883
GENERATE_CONTENT_ID function 883

PIMM gate *(continued)*
GET_ATTACHMENT function 884
PARSE_CONTENT_TYPE function 885
PARSE_MIME_HEADERS function 886
PARSE_MIME_MESSAGE function 887
PARSE_MULTIPART_RELATED function 888
PUT_ATTACHMENT function 888
Pipeline manager domain (PI) domain 868
PIPI_CALL_SUB function, APLI gate 35
PIPI_INIT_SUB_DP function, APLI gate 36
PIPI_TERM function, APLI gate 37
PIPL gate
ADD_PIPELINE function 889
COMPLETE_PIPELINE function 890
DISCARD_PIPELINE function 890
END_BROWSE_PIPELINE function 891
ESTABLISH_PIPELINE function 891
GET_NEXT_PIPELINE function 891
INQUIRE_PIPELINE function 892
PERFORM_PIPELINE function 893
RELINQUISH_PIPELINE function 894
RESOLVE_PIPELINE function 894
SET_PIPELINE function 894
START_BROWSE_PIPELINE function 895
PIPM gate
INVOKE_PROGRAM function 895
INVOKE_STUB function 896
START_PIPELINE function 897
PIRE gate
PERFORM_RESYNC function 898
PISC gate
DYN_CREATE_WEBSERVICE function 898
UPDATE_WEBSERVICE function 899
PISF gate
SOAPFAULT_ADD function 899
SOAPFAULT_CREATE function 900
SOAPFAULT_DELETE function 900
PISN gate
SOAP_11 function 901
SOAP_12 function 901
PITC gate
GET_RESPONSE function 903
ISSUE function 901
TRUST_CLIENT function 903
VALIDATE function 902
PITG gate
CONVERSE function 905
RECEIVE_REQUEST function 906
SEND_ERROR_RESPONSE function 906
SEND_REQUEST function 904
SEND_RESPONSE function 904
PITL gate
PROCESS_SOAP_REQUEST function 906
PIWR gate
CREATE_WEBSERVICE function 907
PIWR gate *(continued)*
DECREMENT_USE_COUNT function 908
DISCARD_WEBSERVICE function 908
END_BROWSE_WEBSERVICE function 909
GET_NEXT_WEBSERVICE function 909
INCREMENT_USE_COUNT function 910
INITIALISE_WEBSERVICE function 911
INQUIRE_WEBSERVICE function 911
RESOLVE_ALL_WEBSERVICES function 913
SET_WEBSERVICE function 913
START_BROWSE_WEBSERVICE function 914
PIXI gate
PARSE_XOP function 914
POINT function, FCCR gate 119
POP_ACD function, MNAC gate 719
POP_HANDLE function, PGHM gate 824
POP_SCOPE function, RLXM gate 940
POP_TASK function, KEDS gate 598
PRE_INITIALISE function, APRD gate 40
PRE_INITIALISE function, DMDM gate 417
PREPARE function, FCCU gate 134
PREPARE function, FCDU gate 151
PREPARE function, OTTR gate 780
PREPARE function, RMOT gate 972
PREPARE_FILE_REQUEST function, FCFR gate 165
PREPARE_RETURN_EXEC function, PGRE gate 863
PREPARE_TO_BACKOUT function, FCFR gate 166
PREPARE_XCTL_EXEC function, PGXE gate 864
Problem solving for generic resource generic resource 1883
problem solving 1883
process overview
adapter 1697
EXEC stubs 1697
FEPI as CICS transaction 1697
logic flow
FEPI application programming commands 1697
FEPI system programming commands 1698
within adapter 1699
Resource Manager 1697
PROCESS_CONTEXT function, PIAT gate 872
PROCESS_CONTEXT_RESP function, PIAT gate 872
PROCESS_DEAD_TCBS function, DSIT gate 477
PROCESS_ECI_FLOW function, IEIE gate 543
PROCESS_ENF_EVENT function, DMEN gate 418
PROCESS_ERROR_QUEUE function, ISRR gate 580
PROCESS_INPUT_QUEUE function, ISRR gate 580
PROCESS_JSON_REQUEST function, PIJS gate 879
PROCESS_KETA_ERROR function, KEDS gate 598
PROCESS_MSG function, PIAT gate 873
PROCESS_PAGE function, DPWD gate 448
PROCESS_PAGE function, DPWE gate 449
PROCESS_PAGE function, DPWJ gate 450
PROCESS_PAGE function, DPWL gate 451
PROCESS_QUIESCE function, FCQU gate 262
PROCESS_SOAP_REQUEST function, PITL gate 906
PROCESS_STATISTICS function, LGJN gate 665
PROCESS_SUBMIT function, DPWD gate 449
PROCESS_SUBMIT function, DPWE gate 450
PROCESS_SUBMIT function, DPWJ gate 451
PROCESS_SUBMIT function, DPWL gate 452
processing using data tables 1592
processing using VSAM 1592
processors 1804
PROFILE function, APID gate 28
program check interrupt system recovery program (SRP) 1787
program isolation deadlock 1844
Program manager domain (PG) domain 784
program preparation utilities command-language translator 1759
program termination block (PTB) 1737, 1745
programming functions with function shipping 1707
protocols, function shipping 1709
protocols, LU6.1 function shipping 1709
PSB (program specification block) DL/I interface 1560
PSB scheduling, DBCTL 1544
PSB scheduling, DL/I 1544, 1760
PSB termination, DBCTL 1544
PSB termination, DL/I 1544, 1760
PSDINT 1897
PSTYPE 1895, 1897
PT (Partner management domain) domain 917
PTB (program termination block) 1737, 1745
PTHREAD_REPLY function, LEPT gate 289
PTTW gate
BREAK_PARTNERSHIP function 917

PTTW gate *(continued)*
CREATE_PARTNERSHIP function 919
CREATE_POOL function 919
DESTROY_PARTNERSHIP function 920
DESTROY_POOL function 921
END_POOL_BROWSE function 921
GET_NEXT_POOL function 921
INQUIRE_GARBAGE_INTERVAL function 922
INQUIRE_USER_TOKEN function 922
MAKE_PARTNERSHIP function 923
QUERY_PARTNERSHIP function 924
QUERY_POOL function 925
SET_GARBAGE_INTERVAL function 926
SET_USER_TOKEN function 926
START_POOL_BROWSE function 926
TRIGGER_PARTNER function 927
WAIT_FOR_PARTNER function 928
PURGE_ALLOCATE_AIDS function, TFAL gate 316
PURGE_INHIBIT_QUERY function, DSAT gate 491
PURGE_TRANSACTION function, XMIQ gate 1371
PUSH_ACD function, MNAC gate 719
PUSH_HANDLE function, PGHM gate 824
PUSH_SCOPE function, RLXM gate 940
PUSH_TASK function, KEDS gate 598
PUT function, TSPT gate 1205
PUT_ATTACHMENT function, PIMM gate 888
PUT_CIB function, CQCQ gate 78
PUT_CLIENT_REQUEST function, RXUW gate 1022
PUT_CONTAINER function, BACR gate 353
PUT_CONTAINER function, PGCR gate 808
PUT_EVENT function, EPEV gate 537
PUT_PROCESSED_CIB function, CQCQ gate 78
PUT_REPLACE function, CCCC gate 370
PUT_REPLACE function, TSPT gate 1206

Q

QUERY function 1831
QUERY_NETNAME function, APID gate 29
QUERY_PARTNERSHIP function, PTTW gate 924
QUERY_POOL function, PTTW gate 925
QUERY_URIMAP function, WBAP gate 1256
QUERY_XML function, MLTF gate 707
QUEUE_EVENT function, EPEV gate 537
queues, transient data
extrapartition 1867, 1870
queues, transient data *(continued)*
indirect 1867
intrapartition 1866, 1868
quiesce receive transaction, file control (DFHFCQR) 1637
QUIESCE_COMPLETE function, FCCA gate 107
QUIESCE_DOMAIN function, DMDM gate 417
QUIESCE_REQUEST function, FCCA gate 107
QUIESCE_SYSTEM function, DMDM gate 410
quiesce, table manager 1798

R

RABN (Resource definition Atom Block Name) 1497
RACB (receive-any control block) 1828
RACE (receive-any RPL pool) 1833
range table 1795
RDO (resource definition online) 1761
CEDA transaction 1761
terminal control autoinstallation 1761
RDUB (Resource Definition Update Block) 1498
REACQUIRE_ENQUEUE function, NQNQ gate 764
READ function, FCCR gate 119
READ function, FCRF gate 268
READ function, WBAP gate 1257
read integrity 1590
read locks, table manager 1797
READ_DELETE function, FCCR gate 121
READ_HEADER function, WBCL gate 1270
READ_HFS_FILE function, DHFS gate 394
READ_INTO function, FCFR gate 166
READ_INTO function, TSQR gate 1208
READ_INTO function, TSSH gate 1220
READ_NEXT function, WBAP gate 1258
READ_NEXT_HEADER function, WBCL gate 1271
READ_NEXT_INTO function, FCFR gate 170
READ_NEXT_INTO function, TSQR gate 1209
READ_NEXT_INTO function, TSSH gate 1220
READ_NEXT_SET function, FCFR gate 173
READ_NEXT_SET function, TSQR gate 1210
READ_NEXT_SET function, TSSH gate 1221
READ_NEXT_UPDATE_INTO function, FCFR gate 175
READ_NEXT_UPDATE_SET function, FCFR gate 178
READ_ONLY function, FCLJ gate 229
READ_PREVIOUS_INTO function, FCFR gate 181
READ_PREVIOUS_SET function, FCFR gate 184
READ_PREVIOUS_UPDATE_INTO function, FCFR gate 186
READ_PREVIOUS_UPDATE_SET function, FCFR gate 189
READ_REQUEST function, WBSV gate 1288
READ_RESPONSE function, WBCL gate 1273
READ_SET function, FCFR gate 192
READ_SET function, TSQR gate 1211
READ_SET function, TSSH gate 1222
READ_TEMPLATE function, DHTM gate 405
READ_TRANSIENT_DATA function, APTD gate 52
READ_UPDATE function, FCLJ gate 230
READ_UPDATE_INTO function, FCFR gate 195
READ_UPDATE_SET function, FCFR gate 198
READNEXT_DEBUG_PROFILE function, DPLM gate 434
READNEXT_INPUT function, DPLM gate 436
READNEXT_PM_PROFILE function, DPFM gate 424
READY function, KEAR gate 583
REATTACH function, XMAT gate 1339
REATTACH_REPLY function, RMUW gate 991
REBUILD_RESOURCE_CLASSES function, XSRC gate 1448
RECEIVE function, APTC gate 49
RECEIVE function, IEIE gate 544
RECEIVE function, SOCK gate 1118
RECEIVE function, WBSR gate 1284
RECEIVE processing, LU6.2 1887
RECEIVE_BUFFER function, ISIS gate 568
RECEIVE_REPLY function, RZSO gate 1028
RECEIVE_REQUEST function, ISIS gate 569
RECEIVE_REQUEST function, PITG gate 906
RECEIVE_REQUEST function, RZTA gate 1032
RECEIVE_REQUEST function, SHRR gate 1036
RECEIVE_SSL_DATA function, SOCK gate 1120
receive-any control block (RACB) 1828
receive-any RPL pool (RACE) 1833
RECLAIM_SOCKET function, SOPL gate 1150
record locking
DELETE request processing 1599
READ UPDATE request 1594
WRITE request for BDAM file 1597
WRITE request for ESDS file 1595
RECORD_NONCOMPLIANCE function, XSRN gate 1448
RECORD_STATISTICS function, STST gate 1168
RECORD_VOTE function, RMLN gate 964

RECOVER_EPADAPTERS function, EPAD gate 534
RECOVER_START_DATA function, TFAL gate 316
RECOVER_STORAGE function, SMCK gate 1069
recoverable data set
  DELETE request processing 1599
  READ request 1594
  REWRITE processing 1597
  WRITE request 1595
recovery
  resource manager interface 1807
  task-related user exits 1807
recovery control program, file control (DFHFCRC) 1638
Recovery manager domain (RM) domain 948
recovery of intrapartition transient data queues 1866
  logical 1867
  physical 1866
recovery point 1619
recovery/restart
  transaction restart program, DFHREST 1844
RECREATE function, RLCB gate 945
RECREATE_COMPLETE function, RLCB gate 946
REFRESH_PROGRAM function, LDLD gate 640
REFRESH_PROGRAM function, PGIS gate 848
Region Status (RS) domain 1012
REGISTER function, DUFT gate 508
REGISTER function, KEAR gate 583
REGISTER function, RLRO gate 939
REGISTER function, RMCD gate 948
REGISTER function, SORD gate 1153
REGISTER_CERTIFICATE_USER function, XSPW gate 1440
REGISTER_CONTROL_ACB function, FCCA gate 108
REGISTER_NOTIFICATION function, SORD gate 1155
REGISTER_POLICY function, MPMO gate 749
REGISTER_TCLASS_USAGE function, XMCL gate 1353
reinstall
  example 1504
  process 1505
relay transaction 1846
relay transaction control blocks 1863
RELAY_TERMINAL_REQUEST function, APRA gate 38
RELEASE function, PGLD gate 856
RELEASE function, TSPT gate 1206
RELEASE_CHAIN_TOKEN function, LGCC gate 652
RELEASE_CONNECTION function, ISCO gate 548
RELEASE_EXEC function, PGLD gate 857
RELEASE_FACILITY function, TFRF gate 331
RELEASE_ICRX function, USAD gate 1241
RELEASE_ICRX function, XSAD gate 1414
RELEASE_ICRX_STORAGE function, XSAD gate 1414
RELEASE_LOCKS function, FCCA gate 109
RELEASE_OPEN_TCB function, DSAT gate 463
RELEASE_PROGRAM function, LDLD gate 641
RELEASE_SURROGATE function, APRS gate 44
RELEASE_TRANSACTION_STG function, SMAR gate 1068
RELEASE_XM_CLIENT function, APXM gate 55
RELEASE_XM_CLIENT function, DPXM gate 453
RELEASE_XM_CLIENT function, MPXM gate 751
RELEASE_XM_CLIENT function, RLXM gate 941
RELEASE_XM_CLIENT function, XMAC gate 1399
RELEASE_XSDBIND function, MLTF gate 708
RELINQUISH function, SOCK gate 1121
RELINQUISH_PIPELINE function, PIPL gate 894
remote DL/I 1760
remote resource names in function shipping 1708
remote system entry 1848
REMOTE_ATTACH function, APRA gate 38
REMOTE_DELETE function, TFAL gate 316
REMOTE_DETACH function, APRA gate 38
REMOVE function, RMRE gate 978
REMOVE_ENQMODEL function, NQRN gate 770
REMOVE_EXPIRED_AID function, TFAL gate 317
REMOVE_EXPIRED_REMOTE_AID function, TFAL gate 318
REMOVE_LINK function, RMLN gate 964
REMOVE_MESSAGE function, TFAL gate 318
REMOVE_REMOTE_DELETES function, TFAL gate 318
REMOVE_SUBEVENT function, EMEM gate 530
RENAME_CHANNEL function, PGCH gate 793
REPLACE function, FCFR gate 202
REPLACE function, FCRF gate 269
REPLACE_DATA function, DDDI gate 383
REPLACE_DATA function, DHDH gate 389
REPLACE_DEBUG_PROFILE function, DPFM gate 426
REPLACE_DELETE function, FCFR gate 204
REPLACE_DELETE function, FCRF gate 270
REPLY_DO_COMMIT function, RMLK gate 1008
REPORT_CONDITION function, APAC gate 22
REPORT_MESSAGE function, XMER gate 1357
REPORT_RECOVERY_STATUS function, RMLN gate 965
request parameter list (RPL), VSAM 1612
request processing, general 1593
Request streams domain (RZ) domain 1024
REQUEST_FORGET function, RMRE gate 979
REQUEST_NOTIFY_INTERVAL function, TISR gate 1181
REQUEST_NOTIFY_OF_A_RESET function, KETI gate 608
REQUEST_NOTIFY_TIME_OF_DAY function, TISR gate 1182
REQUEST_STATISTICS function, STST gate 1168
REROUTE_SHIPPABLE_AIDS function, TFAL gate 319
RESCHEDULE_BMS function, TFAL gate 319
RESERVE function, SOCK gate 1122
RESET_ACTIVITY function, BAAC gate 339
RESET_AID_QUEUE function, TFAL gate 320
RESET_ALL_QUIESCE_STATUS function, FCDN gate 143
RESET_BFAILS function, FCFL gate 158
RESET_BROWSE function, FCFR gate 206
RESET_BROWSE function, FCRF gate 271
RESET_KEYPOINT_STATS function, LGCC gate 652
RESET_LOCAL_TIME function, KETI gate 608
RESET_NONRLS_BATCH function, FCCA gate 110
RESET_PROCESS function, BAPR gate 357
RESET_TRIGGER_LEVEL function, APTD gate 53
RESOLVE_ALL_JVMSERVERS function, SJJS gate 1051
RESOLVE_ALL_WEBSERVICES function, PIWR gate 913
RESOLVE_PIPELINE function, PIPL gate 894
Resource Definition Atom 1497
resource definition atom block name (RABN) 1497
resource definition recovery anchor block (RRAB) 1497
resource definition update block (RDUB) 1498
Resource life-cycle (RL) domain 930

resource manager interface (RMI) 1800
resource manager interface (RMI) recovery 1807
Resource Manager, FEPI 1697
Resource Recovery interface, SAA 1764
resource recovery table 1601
RESOURCE_AVAILABLE function, FCRR gate 278
RESPONSE
  domain interface parameter 7
restart
  emergency 1516
RESTART function, FCCU gate 135
RESTART function, FCDU gate 152
restart of transactions
  DFHREST 1844
restart program, file control (DFHFCRP) 1643
RESTART_FILE_CONTROL function, FCFR gate 208
RESTART_FILE_CONTROL function, FCRP gate 277
RESTART_RLS function, FCRR gate 279
RESTARTBR_DEBUG_PROFILES function, DPLM gate 438
RESTORE_CHAIN_TOKEN function, LGCC gate 653
RESTORE_DIRECTORY function, NQRN gate 770
RESTORE_FROM_KEYPOINT function, TFAL gate 320
RESTORE_STIMER function, KEDS gate 599
RESUME function, DSSR gate 481
RESUME_ACTIVITY function, BAAC gate 339
RESUME_PROCESS function, BAPR gate 357
RESYNC function, OTCO gate 773
RESYNC function, OTSU gate 777
RESYNC_CFDT_LINK function, FCDY gate 153
RESYNC_CFDT_POOL function, FCDY gate 154
RESYNC_COORDINATOR function, OTCP gate 774
RESYNC_LINKS function, ISRE gate 577
RESYNC_SUBORDINATE function, OTCP gate 775
resynchronization protocol 1710
RETAIN function, FCCU gate 135
RETAIN function, FCDU gate 153
RETAIN_DATASET_LOCKS function, FCCA gate 110
RETAIN_UOW_LOCKS function, FCCA gate 111
RETRIEVE_ICRX function, ISIS gate 571
RETRIEVE_MESSAGE function, MEBM gate 687
RETRIEVE_MESSAGE function, MEM4 gate 697
RETRIEVE_MESSAGE function, MEME gate 691
RETRIEVE_REATTACH_EVENT function, EMEM gate 531
RETRIEVE_START_DATA function, TFAL gate 321
RETRIEVE_SUBEVENT function, EMEM gate 531
RETRIEVE_WITH_CTLINFO function, DHDH gate 391
RETRIEVE_WITHOUT_CTLINFO function, DHDH gate 391
RETRY function, FCFL gate 159
RETRY_REQUEST function, SHRR gate 1037
RETURN_CFDT_ENTRY_POINTS function, FCDY gate 155
RETURN_END_ACTIVITY function, BAAC gate 340
RETURN_SOCKET_TO_POOL function, SOPL gate 1149
REWRITE function, FCCR gate 121
REWRITE function, FCFR gate 209
REWRITE function, FCRF gate 272
REWRITE function, TSQR gate 1212
REWRITE function, TSSH gate 1222
REWRITE processing 1597
REWRITE_DELETE function, FCFR gate 211
RIDFLD
  READ request 1594
  READNEXT request processing 1600
  recoverable data set 1594
RL (Resource life-cycle) domain 930
RL domain
  modules 947
RLA (route list area) 1472
RLBR gate
  INQUIRE_BY_NAME function 945
RLCB gate
  CREATE function 942, 946
  DISCARD function 943
  INQUIRE function 943
  RECREATE function 945
  RECREATE_COMPLETE function 946
  SET_MODIFY function 944, 947
RLPM gate
  DISCARD_BUNDLE function 930
  END_BROWSE_BUNDLE function 931
  GET_NEXT_BUNDLE function 931
  INQUIRE_BUNDLE function 933
  INSTALL_BUNDLE function 934
  SET_BUNDLE function 935
  START_BROWSE_BUNDLE function 935
RLR (route list resolution program) 1487
  interfaces, illustrated 1488
  mapping control program (MCP) 1481, 1488
  program manager 1488
  storage control 1488
RLRO gate
  CREATED function 936
  DEREGISTER function 936
  DISCARDED function 937
  DRIVE_PENDING function 937
  END_BROWSE_BUNDLERES function 937
  GET_NEXT_BUNDLERES function 937
  NOTIFY function 938
RLRO gate *(continued)*
  REGISTER function 939
  START_BROWSE_BUNDLERES function 939
RLS clean up transaction, file control (DFHFCRD) 1640
RLS control ACB manager, file control (DFHFCCA) 1616
RLS open/close program, file control (DFHFCRO) 1643
RLS quiesce common system transaction, file control (DFHFCQT) 1637
RLS quiesce exit, file control (DFHFCQX) 1638
RLS quiesce initiation, file control (DFHFCQI) 1637
RLS quiesce processor, file control (DFHFCQU) 1638
RLS quiesce send transaction, file control (DFHFCQS) 1637
RLS record management processor, file control (DFHFCRS) 1645
RLS restart, file control (DFHFCRR) 1645
RLS VSAM interface processor, file control (DFHFCRV) 1645
RLXM gate
  INQUIRE_SCOPE function 940
  POP_SCOPE function 940
  PUSH_SCOPE function 940
  RELEASE_XM_CLIENT function 941
RM (Recovery manager domain) domain 948
RMCD gate
  INQUIRE_CLIENT_DATA function 948
  REGISTER function 948
  SET_CLIENT_DATA function 949
  SET_GATE function 949
RMDE gate
  DELIVER_FORGET function 1001
  DELIVER_RECOVERY function 1001
  END_DELIVERY function 1002
  START_DELIVERY function 1002
RMDM gate
  INQUIRE_LOCAL_LU_NAME function 950
  INQUIRE_STARTUP function 950
  SET_LOCAL_LU_NAME function 950
  SET_PARAMETERS function 951
  SET_STARTUP function 951
RMI (resource manager interface) 1800
RMI (resource manager interface) recovery 1807
RMI_START_OF_TASK function, APXM gate 55
RMKP gate
  TAKE_KEYPOINT function 1003
RMLK gate
  PERFORM_COMMIT function 1003
  PERFORM_PRELOGGING function 1005
  PERFORM_PREPARE function 1005
  PERFORM_SHUNT function 1006
  PERFORM_UNSHUNT function 1007

RMLK gate *(continued)*
REPLY_DO_COMMIT function 1008
SEND_DO_COMMIT function 1009
RMLN gate
ADD_LINK function 951
DELETE_LINK function 954
END_LINK_BROWSE function 954
GET_NEXT_LINK function 955
INBOUND_FLOW function 958
INITIATE_RECOVERY function 958
INQUIRE_LINK function 960
INSERT_LINK function 963
ISSUE_PREPARE function 963
RECORD_VOTE function 964
REMOVE_LINK function 964
REPORT_RECOVERY_STATUS function 965
SET_LINK function 966
SET_MARK function 968
SET_RECOVERY_STATUS function 968
START_LINK_BROWSE function 969
TERMINATE_RECOVERY function 970
RMNM gate
CLEAR_PENDING function 970
INQUIRE_LOGNAME function 971
SET_LOGNAME function 972
RMOT gate
COMMIT function 972
PREPARE function 972
ROLLBACK function 973
SET_OTS_UOW function 973
RMRE gate
APPEND function 973
AVAIL function 975
FORCE function 976
KEYPOINT_DATA function 977
REMOVE function 978
REQUEST_FORGET function 979
RMRO gate
DELIVER_BACKOUT_DATA function 997
END_BACKOUT function 998
PERFORM_COMMIT function 998
PERFORM_PREPARE function 999
PERFORM_SHUNT function 999
PERFORM_UNSHUNT function 1000
START_BACKOUT function 1000
RMSL gate
TAKE_ACTIVITY_KEYPOINT function 980
RMUW gate
BACKOUT_UOW function 980
BIND_UOW_TO_TXN function 981
COMMIT_UOW function 982
CREATE_NETWORK_UOWID function 982
CREATE_UOW function 983
END_UOW_BROWSE function 984
FORCE_UOW function 984
GET_NEXT_UOW function 985
INQUIRE_UOW function 987
INQUIRE_UOW_ID function 990
INQUIRE_UOW_TOKEN function 991
RMUW gate *(continued)*
REATTACH_REPLY function 991
SET_UOW function 992
START_UOW_BROWSE function 992
RMWT gate
END_WORK_TOKEN_BROWSE function 993
GET_NEXT_WORK_TOKEN function 993
INQUIRE_WORK_TOKEN function 994
SET_WORK_TOKEN function 995
START_WORK_TOKEN_BROWSE function 995
ROLLBACK 1509
ROLLBACK function, OTTR gate 780
ROLLBACK function, RMOT gate 973
route list area (RLA) 1472
ROUTE_REQUEST function, SHRR gate 1037
ROUTE_TRANSACTION function, APRT gate 44
ROUTING_CONVERSE function, ISIS gate 571
RPL (request parameter list), VSAM 1612
RPL (request parameter list), VTAM receive-any RPL 1833
RPL (request parameter list), z/OS Communications Server 1819
RRAB (Resource definition Recovery Anchor Block) 1497
RRDS (relative record data set) WRITE request for RRDS file 1596
RRMS domain (RX) 1020
RS (Region status) domain 1012
RS domain modules 1020
RSDU gate
END_SYSTEM_DUMP function 1012
END_TRANSACTION_DUMP function 1013
START_SYSTEM_DUMP function 1013
START_TRANSACTION_DUMP function 1013
RSSR gate
DEREGISTER_INTEREST function 1014
INQUIRE_TARGET_STATUS function 1014
SET_THRESHOLD_PERCENTAGE function 1016
START_RECORDING function 1016
STOP_RECORDING function 1017
TEST_CONNECTION function 1018
RSXM gate
END_TRANSACTION function 1019
START_TRANSACTION function 1019
RUN_ACTIVITY function, BAAC gate 340
RUN_PROCESS function, BAPR gate 358
RUN_SERVICE function, SJTH gate 1060
RUN_TRANSACTION function, XMRU gate 1377
runaway task system recovery program (SRP) 1787
RX (RRMS) domain 1020
RXDM gate
INQUIRE_RRS function 1020
SET_PARAMETERS function 1021
RXUW gate
GET_CLIENT_REQUEST function 1021
INQUIRE function 1022
PUT_CLIENT_REQUEST function 1022
RZ (Request streams domain) domain 1024
RZRJ gate
PERFORM_JOIN function 1024
RZRT gate
SET_EXIT_PROGRAM function 1025
RZSO gate
CREATE function 1025
JOIN function 1027
LEAVE function 1027
RECEIVE_REPLY function 1028
SEND_REQUEST function 1028
WEAK_JOIN function 1029
RZTA gate
GET_CURRENT function 1030
GET_DEBUG_DATA function 1030
GET_JOIN_DATA function 1030
GET_PUBLIC_ID function 1031
GET_SERVER_DATA function 1031
RECEIVE_REQUEST function 1032
SEND_REPLY function 1032
TERMINATE function 1033

S

S2AD gate
ADD_SUBPOOL function 1088
DELETE_SUBPOOL function 1090
END_SUBPOOL_BROWSE function 1090
GET_NEXT_SUBPOOL function 1090
INQUIRE_SUBPOOL function 1091
START_SUBPOOL_BROWSE function 1091
S2GF gate
FREEMAIN function 1091
GETMAIN function 1092
INQUIRE_ELEMENT_LENGTH function 1093
S2SR gate
INQUIRE_ACCESS_TOKEN function 1095
INQUIRE_MEMLIMIT function 1095
SAA Communications interface 1764
SAA Resource Recovery interface 1764
SAB (subsystem anchor block) 1775
SAIQ gate
INQUIRE_SYSTEM function 290
SET_SYSTEM function 293
SAVE_DEBUG_PROFILE function, DPFM gate 428
SAVE_USER_DEFAULTS function, DPUM gate 445

SCACB (subsystem connection address control block) 1727
SCCB (subsystem connection control block) 1727
SCHEDULE_BMS function, TFAL gate 321
SCHEDULE_RECEIVER_TASK function, SOCK gate 1123
SCHEDULE_START function, TFAL gate 322
SCHEDULE_TDP function, TFAL gate 324
Scheduler services domain (SH) domain 1034
SCP (storage control program)
  3270 mapping (M32) 1484
  data interchange program (DIP) 1547
  LU1 printer with extended attributes mapping program (ML1) 1483
  mapping control program (MCP) 1481
  message switching 1748
  non-3270 input mapping (IIP) 1479
  page and text build (PBP) 1486
  partition handling program (PHP) 1487
  route list resolution program (RLR) 1488
  terminal control 1819
  terminal page processor (TPP) 1489
  terminal page retrieval program (TPR) 1492
  transaction failure program (TFP) 1843
  undelivered messages clean up program (TPQ) 1490
SCTE (subsystem control table extension) 1726, 1775
SEARCH_LDAP function, DDAP gate 378
secondary index 1795
Security domain (XS) domain 1404
SELECT_MODEL function, MPMO gate 750
SEND function, APTC gate 49
SEND function, IEIE gate 544
SEND function, SOCK gate 1124
SEND function, WBSR gate 1286
SEND processing, LU6.2 1887
SEND_BUFFER function, ISIS gate 572
SEND_DEFERRED_ABEND function, KEDS gate 600
SEND_DO_COMMIT function, RMLK gate 1009
SEND_ERROR function, IEIE gate 545
SEND_ERROR function, ISIS gate 573
SEND_ERROR_RESPONSE function, PITG gate 906
SEND_MESSAGE function, MEM4 gate 699
SEND_MESSAGE function, MEME gate 692
SEND_QUIESCES function, FCQS gate 262
SEND_REPLY function, RZTA gate 1032
SEND_REQUEST function, PITG gate 904
SEND_REQUEST function, RZSO gate 1028
SEND_RESPONSE function, ISIS gate 574
SEND_RESPONSE function, PITG gate 904
SEND_RESPONSE function, WBAP gate 1259
SEND_SSL_DATA function, SOCK gate 1125
SEND_STATIC_RESPONSE function, WBSR gate 1287
sequential retrieval 1590
service request block (SRB) 1819
SERVREQ attribute of file
  DELETE request 1598
  READ request 1593
session management, LU6.2 1885
session states, LU6.2 1887
sessions
  recovery 1730
SET function, FCCT gate 130
SET function, LGJN gate 665
SET function, SOIS gate 1143
SET VTAM 1897
SET_ABEND function, PGHM gate 824
SET_AIDS function, PGHM gate 826
SET_ANCHOR function, KEDD gate 588
SET_APP_CONTEXT function, MNAD gate 726
SET_ATOMSERVICE function, W2AT gate 1332
SET_AUTOINSTALL function, PGAQ gate 785
SET_AUX_TRACE_AUTOSWITCH function, TRSR gate 1188
SET_BOUND_REQUEST function, SHPR gate 1035
SET_BUFFERS function, TSSR gate 1225
SET_BUNDLE function, RLPM gate 935
SET_BUNDLE_BY_ID_VER function, APRL gate 43
SET_CATALOG_RECOV_POINT function, FCAT gate 101
SET_CATALOG_RECOV_REQD function, FCAT gate 102
SET_CATALOG_RECOVERED function, FCAT gate 103
SET_CATALOG_RECOVERED function, FCDN gate 144
SET_CLIENT_DATA function, RMCD gate 949
SET_CONDITIONS function, PGHM gate 827
SET_CONTAINER function, PGCR gate 810
SET_COORDINATOR function, OTCO gate 774
SET_CURRENT_CHANNEL function, PGCH gate 793
SET_DATE_FORMAT function, KETI gate 608
SET_DEBUG_PROFILE function, DPIQ gate 432
SET_DEBUGGING function, DPIQ gate 433
SET_DEFAULT_RECOVERY function, KEDD gate 589
SET_DEFER_INTERVAL function, LGCC gate 653
SET_DEFERRED_ABEND function, XMER gate 1358
SET_DEFERRED_MESSAGE function, XMER gate 1358
SET_DISPATCHER function, DSIT gate 477
SET_DOMAIN_TRACE function, KEDD gate 589
SET_DSA_LIMIT function, SMSR gate 1084
SET_DSA_SIZE function, SMSR gate 1085
SET_DSNB function, FCDN gate 144
SET_DTRTRAN function, XMSR gate 1378
SET_DUMPDS_AUTOSWITCH function, DUSR gate 512
SET_DUMPTABLE_DEFAULTS function, DUSR gate 513
SET_ENQMODEL function, NQRN gate 770
SET_EPADAPTER function, EPAD gate 534
SET_EPADAPTER function, EPIS gate 542
SET_EVENT_PROCESSING function, EPIS gate 542
SET_EVENTBINDING function, ECIS gate 90
SET_EVENTPROCESS function, ECIS gate 89
SET_EXIT_PROGRAM function, RZRT gate 1025
SET_EXIT_PROGRAM function, SHRT gate 1038
SET_EXIT_STATUS function, APUE gate 334
SET_GARBAGE_INTERVAL function, PTTW gate 926
SET_GATE function, RMCD gate 949
SET_GLOBAL_TRACE function, KEDD gate 590
SET_HISTORY function, LGCC gate 654
SET_HOST function, WBUR gate 1315
SET_INITIAL_DUMPDS function, DUSR gate 513
SET_INTERNAL_TABLE_SIZE function, TRSR gate 1189
SET_IPCONN function, ISIC gate 560
SET_JSONTRANSFRM function, MLJS gate 704
SET_JVMSERVER function, SJJS gate 1051
SET_JVMSERVER_DISABLING function, SJTH gate 1061
SET_KERNEL function, KEGD gate 604
SET_KEYPOINT_FREQUENCY function, LGCC gate 654
SET_LAST_AGENT function, OTCO gate 774
SET_LIBRARY function, LDLB gate 622
SET_LINK function, RMLN gate 966

SET_LOCAL_LU_NAME function, RMDM gate 950
SET_LOGNAME function, RMNM gate 972
SET_MARK function, RMLN gate 968
SET_MCT_SUFFIX function, MNSR gate 733
SET_MESSAGE_OPTIONS function, MESR gate 701
SET_MODIFY function, RLCB gate 944, 947
SET_MONITORING function, MNSR gate 733
SET_MXT function, XMSR gate 1379
SET_NETWORK_IDENTIFIER function, XSIS gate 1431
SET_NQRNAME_LIST function, NQNQ gate 765
SET_OPTIONS function, LDLD gate 642
SET_OTS_UOW function, RMOT gate 973
SET_PARAMETERS function, DHDH gate 392
SET_PARAMETERS function, DPIQ gate 433
SET_PARAMETERS function, ISIS gate 575
SET_PARAMETERS function, LGPA gate 673
SET_PARAMETERS function, RMDM gate 951
SET_PARAMETERS function, RXDM gate 1021
SET_PARAMETERS function, SOIS gate 1145
SET_PARAMETERS function, W2W2 gate 1334
SET_PHASE function, DMDM gate 410
SET_PIPELINE function, PIPL gate 894
SET_POOL function, FCRL gate 276
SET_PRIORITY function, DSAT gate 463
SET_PROCESSTYPE function, BATT gate 362
SET_PROGRAM function, PGIS gate 849
SET_RECOVERY_STATUS function, RMLN gate 968
SET_REENTRANT_PROGRAM function, SMSR gate 1085
SET_REMOTE_STATUS function, OTRS gate 776
SET_RETRY_TIME function, DUSR gate 514
SET_ROLE_FOR_CODED_ROLE function, XSEJ gate 1423
SET_ROLLBACK_ONLY function, OTTR gate 780
SET_SECURITY_DOMAIN_PARMS function, XSIS gate 1432
SET_SESSION function, APTC gate 50
SET_SESSION function, WBSV gate 1292
SET_SOCKET_OPTS function, SOCK gate 1126
SET_SPECIAL_TOKENS function, XSIS gate 1435
SET_START_TYPE function, TSSR gate 1225
SET_STARTUP function, RMDM gate 951
SET_STATISTICS_OPTIONS function, STST gate 1169
SET_STORAGE_PROTECT function, SMSR gate 1086
SET_STORAGE_RECOVERY function, SMSR gate 1086
SET_STRINGS function, TSSR gate 1225
SET_SYMBOL_VALUE_BY_API function, DHSL gate 398
SET_SYMBOL_VALUE_BY_SSI function, DHSL gate 399
SET_SYSTEM function, PGAQ gate 786
SET_SYSTEM function, SAIQ gate 293
SET_SYSTEM_DUMP function, DUSR gate 514
SET_SYSTEM_DUMPCODE function, DUDT gate 500
SET_TASK function, DSBR gate 470
SET_TASK_TRACE function, KEDD gate 590
SET_TCB function, DSBR gate 471
SET_TCLASS function, XMCL gate 1353
SET_TCPIPSERVICE function, SOTB gate 1162
SET_TDQDEF function, TDTM gate 307
SET_TERMINAL_FACILITY function, TFIQ gate 330
SET_THRESHOLD_PERCENTAGE function, RSSR gate 1016
SET_TRACKING_DATA function 735
SET_TRACKING_DATA function, MNTD gate 735
SET_TRAN_DUMPCODE function, DUDT gate 501
SET_TRANDEF function, XMXD gate 1394
SET_TRANSACTION function, XMIQ gate 1372
SET_TRANSACTION_ISOLATION function, SMSR gate 1087
SET_TRANSACTION_TOKEN function, DSAT gate 464
SET_TRANSACTION_TOKEN function, XMIQ gate 1373
SET_TRANTABLESIZE function, DUSR gate 514
SET_TRANTABLETYPE function, DUSR gate 515
SET_TRAP_OFF function, KEDD gate 591
SET_TRAP_ON function, KEDD gate 591
SET_UCD function, MNOD gate 731
SET_UOW function, RMUW gate 992
SET_URIMAP function, WBUR gate 1316
SET_USER_DOMAIN_PARMS function, USIS gate 1244
SET_USER_TOKEN function, PTTW gate 926
SET_USER_TOKEN, ICUS gate 287
SET_VOTE function, OTSU gate 777
SET_WEBSERVICE function, PIWR gate 913
SET_WORK_TOKEN function, RMWT gate 995
SET_XMLTRANSFORM function, MLXT gate 713
SH (Scheduler services domain) domain 1034
share control block manager, file control (DFHFCRL) 1642
shared data table function ship program, file control (DFHFCDTX) 1620
shared data table services 1591
shared resources control (SHRCTL) block 1611
shared resources pool processor, file control (DFHFCL) 1627
shipping TCTTE for transaction routing 1458
SHPR gate
  ADD_PENDING_REQUEST function 1034
  DELETE_PENDING_REQUEST function 1035
  SET_BOUND_REQUEST function 1035
SHRCTL (shared resources control) block 1611
SHRQ gate
  PERFORM_REGULAR_DREDGE function 1036
  PERFORM_RESTART_DREDGE function 1036
  PERFORM_SHUTDOWN function 1036
SHRR gate
  RECEIVE_REQUEST function 1036
  RETRY_REQUEST function 1037
  ROUTE_REQUEST function 1037
SHRT gate
  INQUIRE_EXIT_PROGRAM function 1038
  SET_EXIT_PROGRAM function 1038
shutdown 1517
shutdown program, file control (DFHFCSD) 1646
side information (partner) 1765
sign-on to consoles flow 1456
SIGNAL_EVENT function, ECSE gate 97
single-node persistent sessions 1895, 1897
single-phase commit process 1809
SJ (JVM domain) domain 1041
SJBD gate
  END_BROWSE_OSGIBUNDLE function 1041
  END_BROWSE_OSGISERVICE function 1042
  GET_NEXT_OSGIBUNDLE function 1042
  GET_NEXT_OSGISERVICE function 1042
  INQUIRE_OSGIBUNDLE function 1043
  INQUIRE_OSGIBUNDLE_BY_ID function 1045

SJBD gate *(continued)*
INQUIRE_OSGISERVICE function 1044
START_BROWSE_OSGIBUNDLE function 1045
START_BROWSE_OSGISERVICE function 1045
SJDS gate
DELETE_THREADED_TCB function 1046
SJJS gate
COMPLETE_JVMSERVER function 1047
CREATE_JVMSERVER function 1046
DISCARD_JVMSERVER function 1048
END_BROWSE_JVMSERVER function 1049
GET_NEXT_JVMSERVER function 1049
INQUIRE_JVMSERVER function 1050
MARK_THREAD_DELETED function 1051
RESOLVE_ALL_JVMSERVERS function 1051
SET_JVMSERVER function 1051
START_BROWSE_JVMSERVER function 1052
SJTH gate
DISABLE_OSGIBUNDLE function 1052
ENABLE_OSGIBUNDLE function 1053
END_BROWSE_OSGISERVICE function 1053
FIND_HIGHEST_VERSION function 1054
GET_NEXT_OSGISERVICE function 1055
INQUIRE_OSGIBUNDLE function 1055
INQUIRE_OSGISERVICE function 1056
INSTALL_OSGIBUNDLE function 1057
INVOKE_JAVA_PROGRAM function 1058
JVMSERVER_STATISTICS function 1059
KILL_JVMSERVER function 1059
RUN_SERVICE function 1060
SET_JVMSERVER_DISABLING function 1061
START_BROWSE_OSGISERVICE function 1061
UNINSTALL_OSGIBUNDLE function 1062
SLCB (subsystem logon control block) 1727
SLOWDOWN_PURGE function, TFAL gate 324
SM (Storage manager domain) domain 1063
SMAD gate
ADD_SUBPOOL function 1064
DELETE_SUBPOOL function 1065
SMAD gate *(continued)*
END_SUBPOOL_BROWSE function 1066
GET_NEXT_SUBPOOL function 1066
INQUIRE_SUBPOOL function 1066
START_SUBPOOL_BROWSE function 1067
SMAR gate
ALLOCATE_TRANSACTION_STG function 1067
RELEASE_TRANSACTION_STG function 1068
SMCK gate
CHECK_STORAGE function 1068
RECOVER_STORAGE function 1069
SMGF gate
FREEMAIN function 1070
GETMAIN function 1071
INQUIRE_ELEMENT_LENGTH function 1072
SMK7 gate
ADD_TRUSTED_EXTENSION function 1073
DELETE_TRUSTED_EXTENSION function 1074
SMMC gate
FREEMAIN function 1075
FREEMAIN_ALL_TERMINAL function 1076
GETMAIN function 1077
INITIALISE function 1078
INQUIRE_ELEMENT_LENGTH function 1079
INQUIRE_TASK_STORAGE function 1079
SMNT gate
MVS_STORAGE_NOTIFY function 1097
STORAGE_NOTIFY function 1098
SMSR gate
INQ_TRANSACTION_ISOLATION function 1080
INQUIRE_ACCESS function 1080, 1094
INQUIRE_ACCESS_TOKEN function 1081
INQUIRE_DSA_LIMIT function 1082
INQUIRE_DSA_SIZE function 1082
INQUIRE_ISOLATION_TOKEN function 1083
INQUIRE_REENTRANT_PROGRAM function 1083
INQUIRE_SHORT_ON_STORAGE function 1083
INQUIRE_STORAGE_PROTECT function 1084
SET_DSA_LIMIT function 1084
SET_DSA_SIZE function 1085
SET_REENTRANT_PROGRAM function 1085
SET_STORAGE_PROTECT function 1086
SET_STORAGE_RECOVERY function 1086
SET_TRANSACTION_ISOLATION function 1087
SWITCH_SUBSPACE function 1087
SMSR gate *(continued)*
UPDATE_SUBSPACE_TCB_INFO function 1088
SNAPSHOT_MVSTCBS function, DSMT gate 480
SNPS 1895, 1897
SO (Sockets domain) domain 1103
SOAD gate
ADD_REPLACE_TCPIPSERVICE function 8, 9, 1103
DELETE_TCPIPSERVICE function 1105
SOAP_11 function, PISN gate 901
SOAP_12 function, PISN gate 901
SOAPFAULT_ADD function, PISF gate 899
SOAPFAULT_CREATE function, PISF gate 900
SOAPFAULT_DELETE function, PISF gate 900
SOCK gate
ACCEPT function 1105
BIND function 1107
CANCEL function 1108
CLOSE function 1109
CONNECT function 1110
CREATE function 1112
ESTABLISH function 1113
GET_DATA_LENGTH function 1114
GET_SOCKET_OPTS function 1115
LISTEN function 1117
RECEIVE function 1118
RECEIVE_SSL_DATA function 1120
RELINQUISH function 1121
RESERVE function 1122
SCHEDULE_RECEIVER_TASK function 1123
SEND function 1124
SEND_SSL_DATA function 1125
SET_SOCKET_OPTS function 1126
SURRENDER function 1128
SWITCH_FOR_ASYNC_IO function 1130
SWITCH_FROM_SOCKET_TCB function 1129
SWITCH_TO_SOCKET_TCB function 1129
Sockets domain (SO) domain 1103
SOIS gate
DELETE_CERTIFICATE_DATA function 1130
EXPORT_CERTIFICATE_DATA function 1131
IMPORT_CERTIFICATE_DATA function 1132
INITIALIZE_ENVIRONMENT function 1132
INQUIRE function 1133
INQUIRE_CONNECTION function 1138
INQUIRE_PARAMETERS function 1140
INQUIRE_SOCKET_TOKEN function 1142
INQUIRE_STATISTICS function 1142
SET function 1143
SET_PARAMETERS function 1145

SOIS gate *(continued)*
VALIDATE_CIPHERS function 1146
VERIFY_IP_ADDRESS function 1146
SOLICIT_INQUIRES function, TSRM gate 1227
SOLS gate
LISTEN function 1147
SOPL gate
CLOSE_SOCKET_IN_POOL function 1150
CREATE_SOCKET_POOL function 1148
DELETE_SOCKET_POOL function 1149
GET_SOCKET_FROM_POOL function 1149
INQUIRE_SOCKET_POOL function 1148
INQUIRE_STATISTICS function 1151
RECLAIM_SOCKET function 1150
RETURN_SOCKET_TO_POOL function 1149
SORD gate
DEREGISTER function 1151
IMMCLOSE function 1152
REGISTER function 1153
REGISTER_NOTIFICATION function 1155
SORL gate
UPDATE_REVOCATION_LIST function 1156
SOTB gate
END_BROWSE function 1156
GET_NEXT function 1156
INQUIRE_TCPIPSERVICE function 1159
SET_TCPIPSERVICE function 1162
START_BROWSE function 1163
specific formats 6
specific gates 5
SPIE exit routine 1780
SPP (syncpoint program) 1719, 1781, 1802, 1807, 1887
SRB (service request block) 1819
SRP (system recovery program)
abnormal termination 1787
program check interrupt 1787
runaway task 1787
system abend 1787
system recovery table (SRT) 1788
SRT (system recovery table) 1788
SSCT (subsystem communication table) 1773, 1775
SSVT (subsystem vector table) 1773, 1775
ST (Statistics domain) domain 1165
START_ABEND function, ABAB gate 17
START_AUXILIARY_TRACE function, TRSR gate 1189
START_BACKOUT function, RMRO gate 1000
START_BROWSE function, AIIQ format 1465
START_BROWSE function, CCCC gate 370
START_BROWSE function, DDBR gate 381
START_BROWSE function, DHTM gate 406
START_BROWSE function, DMIQ gate 415
START_BROWSE function, DSBR gate 471
START_BROWSE function, FCFR gate 213
START_BROWSE function, FCRF gate 273
START_BROWSE function, LDLD gate 643
START_BROWSE function, LGJN gate 666
START_BROWSE function, LGLD gate 672
START_BROWSE function, LGST gate 677
START_BROWSE function, SOTB gate 1163
START_BROWSE function, TSBR gate 1198
START_BROWSE function, TSMB gate 1202
START_BROWSE function, TSSH gate 1223
START_BROWSE function, WBAP gate 1260
START_BROWSE_ALL function, LGBA gate 647
START_BROWSE_APPLICATION function, PGIS gate 852
START_BROWSE_ATOMSERVICE function, W2AT gate 1332
START_BROWSE_BUNDLE function, RLPM gate 935
START_BROWSE_BUNDLERES function, RLRO gate 939
START_BROWSE_CAPTURESPEC function, ECIS gate 90
START_BROWSE_CHAINS function, LGCC gate 655
START_BROWSE_DATA_PRED function, ECIS gate 90
START_BROWSE_DIRECTORY function, DHFS gate 395
START_BROWSE_ENQMODEL function, NQRN gate 771
START_BROWSE_ENQUEUE function, NQIB gate 758
START_BROWSE_EPADAPTER function, EPIS gate 542
START_BROWSE_EVENT function, EMBR gate 524
START_BROWSE_EVENTBINDING function, ECIS gate 91
START_BROWSE_FILE function, FCMT gate 254
START_BROWSE_HEADERS function, WBCL gate 1275
START_BROWSE_HOST function, WBUR gate 1319
START_BROWSE_INFO_SOURCE function, ECIS gate 91
START_BROWSE_JVMSERVER function, SJJS gate 1052
START_BROWSE_LIBRARY function, LDLB gate 623
START_BROWSE_MVSTCB function, DSMT gate 480
START_BROWSE_OPTION_PRED function, ECIS gate 92
START_BROWSE_OSGIBUNDLE function, SJBD gate 1045
START_BROWSE_OSGISERVICE function, SJBD gate 1045
START_BROWSE_OSGISERVICE function, SJTH gate 1061
START_BROWSE_PIPELINE function, PIPL gate 895
START_BROWSE_POLICY_RULE function, MPIB gate 748
START_BROWSE_PROCESSTYPE function, BATT gate 362
START_BROWSE_PROGRAM function, PGIS gate 853
START_BROWSE_RESULTS function, DDAP gate 379
START_BROWSE_TCLASS function, XMCL gate 1354
START_BROWSE_TDQDEF function, TDTM gate 308
START_BROWSE_TIMER function, EMBR gate 525
START_BROWSE_TRANDEF function, XMBD gate 1348
START_BROWSE_TRANSACTION function, XMIQ gate 1374
START_BROWSE_TXN_TOKEN function, XMIQ gate 1375
START_BROWSE_URIMAP function, WBUR gate 1320
START_BROWSE_WEBSERVICE function, PIWR gate 914
START_BROWSE_XMLTRANSFORM function, MLXT gate 714
START_CHAIN_BROWSE function, LGCB gate 648
START_DELIVERY function, RMDE gate 1002
START_DSNB_BROWSE function, FCDN gate 145
START_FILE_IN_POOL_BROWSE function, FCST gate 284
START_FORCE_PURGE_PROTECT function, KEDS gate 601
START_GTF_TRACE function, TRSR gate 1190
START_INIT function, AIIN format 1464
START_INIT function, CPIN format 1767
START_INTERNAL_TRACE function, TRSR gate 1190
START_LINK_BROWSE function, RMLN gate 969
START_PIPELINE function, PIPM gate 897
START_PM_BROWSE function, DPFM gate 431
START_POOL_BROWSE function, PTTW gate 926
START_PROGRAM function, APLI gate 33

START_PURGE_PROTECTION function, KEDS gate 601
START_RECORDING function, RSSR gate 1016
START_RUNAWAY_TIMER function, KEDS gate 601
START_SUBPOOL_BROWSE function, S2AD gate 1091
START_SUBPOOL_BROWSE function, SMAD gate 1067
START_SYSTEM_DUMP function, RSDU gate 1013
START_TRANSACTION function, RSXM gate 1019
START_TRANSACTION_DUMP function, RSDU gate 1013
START_TSPOOL_BROWSE function, TSSH gate 1224
START_UOW_BROWSE function, RMUW gate 992
START_UOWDSN_BROWSE function, FCFL gate 159
START_WORK_TOKEN_BROWSE function, RMWT gate 995
START_WRITE function, CCCC gate 370
start, cold 1516
start, warm 1516
STARTBR_ACTIVITY function, BABR gate 347
STARTBR_CONTAINER function, BABR gate 348
STARTBR_CONTAINER function, PGCR gate 811
STARTBR_DEBUG_PROFILES function, DPLM gate 439
STARTBR_PROCESS function, BABR gate 348
STARTBR_SYSTEM_DUMPCODE function, DUDT gate 503
STARTBR_TRAN_DUMPCODE function, DUDT gate 503
STARTBROWSE_IPCONN function, ISIC gate 562
STARTUP_CLOSE function, CCCC gate 371
STARTUP_OPEN function, CCCC gate 371
static storage, file control (FC static) 1605
statistics
  CICS-DB2 1528
statistics collection 1768
Statistics domain (ST) domain 1165
statistics program, file control (DFHFCST) 1646
statistics utility program (DFHSTUP) 1768
STATISTICS_COLLECTION function, STST gate 1170
STOP_AUXILIARY_TRACE function, TRSR gate 1190
STOP_FORCE_PURGE_PROTECT function, KEDS gate 602
STOP_GTF_TRACE function, TRSR gate 1191
STOP_INTERNAL_TRACE function, TRSR gate 1191
STOP_PURGE_PROTECTION function, KEDS gate 602
STOP_RECORDING function, RSSR gate 1017
STOP_RUNAWAY_TIMER function, KEDS gate 602
storage control macro-compatibility interface 1772
Storage manager domain (SM) domain 1063
STORAGE_NOTIFY function, SMNT gate 1098
STREAM_FAIL function, LGJN gate 666
STST gate
  COLLECT_RESOURCE_STATS function 1165, 1172
  COLLECT_STATISTICS function 1166, 1173
  DISABLE_STATISTICS function 1167
  INQ_STATISTICS_OPTIONS function 1167
  RECORD_STATISTICS function 1168
  REQUEST_STATISTICS function 1168
  SET_STATISTICS_OPTIONS function 1169
  STATISTICS_COLLECTION function 1170
subsystem anchor block (SAB) 1775
subsystem communication table (SSCT) 1773, 1775
subsystem connection address control block (SCACB) 1727
subsystem connection control block (SCCB) 1727
subsystem control table extension (SCTE) 1726, 1775
subsystem definition 1773
subsystem interface 1773
  console message handling 1774
  control blocks, illustrated 1775
subsystem logon control block (SLCB) 1727
subsystem user definition block (SUDB) 1726
subsystem vector table (SSVT) 1773, 1775
subtask control 1777
SUDB (subsystem user definition block) 1726
SURRENDER function, SOCK gate 1128
surrogate session entry 1848
surrogate TCTTE 1864
SUSPEND function, DSSR gate 482
SUSPEND_ACTIVITY function, BAAC gate 341
SUSPEND_PROCESS function, BAPR gate 358
SVC 53
  REWRITE request processing 1597
  UNLOCK request processing 1598
  WRITE request processing 1597
SWITCH_AUXILIARY_EXTENTS function, TRSR gate 1191
SWITCH_FOR_ASYNC_IOB function, SOCK gate 1130
SWITCH_FROM_SOCKET_TCB function, SOCK gate 1129
SWITCH_SUBSPACE function, SMSR gate 1087
SWITCH_TO_SOCKET_TCB function, SOCK gate 1129
SYNCHRONISE_READ_UPDATE function, FCLJ gate 231
synchronization of tasks
  time 1733
synchronization processing, initiating 1560
syncpoint 1498
  function shipping 1720
SYNCPOINT_REQUEST, CPSP format 1767
SYSINI function, LGCC gate 655
sysplex, MVS
  cross-system coupling facility (XCF)
    for MRO links across MVS images 1723
system abend
  system recovery program (SRP) 1787
system control
  autoinstall terminal model manager 1463
  CICS-DB2 Attachment facility 1520
  DL/I database support 1535, 1559, 1760
  EXEC interface program (EIP) 1570
  file control 1588
  interval control 1732
  program control 1755
  storage control macro-compatibility interface 1772
  subtask control 1777
  syncpoint program (SPP) 1781
  table manager 1794
  task-related user exit control 1800
  terminal control 1814
  trace control macro-compatibility interface 1836
  transient data control 1866
  user exit control 1871
system definition, DBCTL 1544
system definition, DL/I 1760
system dump formatting program 1784
system EIB 1571
system entries, TCT (terminal control table) 1710
system programming commands, FEPI logic flow 1698
system reliability
  node abnormal condition program (z/OS Communications Server) 1750
  node error program (z/OS Communications Server) 1754
  program error program (PEP) 1758
  system recovery program (SRP) 1787
  task-related user exit recovery 1807
  terminal abnormal condition program (BSAM) 1811
  terminal error program 1836
  transaction failure program (TFP/ACP) 1842

system reliability *(continued)*
transaction restart program, DFHREST 1844
system services
dynamic allocation sample program (DYNALLOC) 1565
field engineering program 1588
master terminal 1746
message switching 1747
operator terminal 1746
resource definition online (RDO) 1761
subsystem interface 1773
supervisory terminal 1746
system spooling interface 1792
system spooling interface 1792
system utilities
command-language translator 1759
CSD utility program 1533
dump utility program 1563
statistics utility program 1768
system dump formatting program 1784
trace utility program 1839
SYSTEM_DUMP function, DUDU gate 503
SYSTEM_EVENT_STATUS function, ECSC gate 95
SYSTEM_EVENT_THRESHOLD function, ECSC gate 92, 93, 96

T

table management program (TMP) 1829
table manager, file control (DFHFCMT) 1629
tabs, horizontal
and device independence 1471
tabs, vertical
and device independence 1471
TACLE (terminal abnormal condition line entry)
terminal control 1820
TACP (terminal abnormal condition program) 1811, 1816
BSAM 1811
default error handling 1813
message construction matrix 1812
message routines 1811
terminal control 1820
TAKE_ACTIVITY_KEYPOINT function, RMSL gate 980
TAKE_KEYPOINT function, FCLJ gate 232
TAKE_KEYPOINT function, RMKP gate 1003
TAKE_KEYPOINT function, TFAL gate 324
TAKE_TRANSACTION_DUMP function, ABAB gate 19
TAKEOVER function, USFL gate 1243
task abnormal condition
abnormal condition program (DFHACP) 1842
transaction failure program (DFHTFP) 1842
task control area facility control area associated address (TCAFCAAA) 1818
task interface element (TIE) 1782, 1800, 1806
task synchronization, time 1733
TASK_REPLY function, DSAT gate 490
TASK_REPLY function, KEDS gate 609
task-related user exit
CICS-DB2 1522
task-related user exits 1800, 1808
control 1800
control blocks, illustrated 1805
entry to 1803
implementation 1802
recovery 1807
recovery token 1807
resynchronization 1782
state of 1802
TCA (task control area)
message switching 1749
terminal control 1820
user 1820
TCAFCAAA (task control area facility control area associated address) 1818
TCB_POOL_MANAGEMENT function, DSAT gate 465
TCB_REPLY function, KEDS gate 610
TCT (terminal control table) 1814, 1828
terminal control table prefix 1833
terminal control table wait list 1833
TCTLE (terminal control table line entry) 1819, 1833
TCTSE (terminal control table system entry) 1710
TCTTE (terminal control table terminal entry) 1833
allocation in function shipping 1719
partition handling program (PHP) 1487
shipping for transaction routing 1458
surrogate 1864
TCTTE creation and deletion 1494
TCTTE generation
DFHZCQ 1496
TCTTE layout 1512
TCTTE structure 1501
TDOC gate
CLOSE_ALL_EXTRA_TD_QUEUES function 293
CLOSE_TRANSIENT_DATA function 293
OPEN_TRANSIENT_DATA function 294
TDTM gate
ADD_REPLACE_TDQDEF function 295
COMMIT_TDQDEFS function 299
DISCARD_TDQDEF function 299
END_BROWSE_TDQDEF function 299
GET_NEXT_TDQDEF function 300
INQUIRE_TDQDEF function 303
SET_TDQDEF function 307
START_BROWSE_TDQDEF function 308
TDXM gate
BIND_SECONDARY_FACILITY function 309
INQUIRE_TRAN_DATA_FACILITY function 309
Temporary Storage Domain (TS) domain 1193
temporary-storage browse transaction, CEBR 1583
temporary-storage control
data interchange program (DIP) 1547
mapping control program (MCP) 1480
message switching 1748
terminal page processor (TPP) 1489
terminal page retrieval program (TPR) 1492
undelivered messages clean up program (TPQ) 1490
TEP (terminal error program) 1816, 1836
TERM_AVAILABLE_FOR_QUEUE function, TFAL gate 325
TERM_TRANSACTION_USER function, USXM gate 1250
terminal control 1814
3270 mapping (M32) 1484
access method dependent interface 1819
access methods 1819
autoinstallation 1761
BSAM routines, illustrated 1821
builder parameter set 1494
common interface 1818
control blocks, illustrated 1831
data interchange program (DIP) 1547
flow through device-dependent modules, illustrated 1822
for function shipping 1719
indexes 1828
interfaces, illustrated 1817
locks 1829
node abnormal condition program (NACP) 1820
node error program (NEP) 1820
non-3270 input mapping (IIP) 1479
service request facilities 1815
storage control 1819
system console support 1823
system control services 1815
task control 1818, 1819
task control area (TCA)
facility control area associated address 1818
task control area, user 1820
TCA (task control area), terminal control 1820
terminal abnormal condition line entry (TACLE) 1820
terminal abnormal condition program (TACP) 1820
terminal page retrieval program (TPR) 1492
transmission facilities
z/OS Communications Server 1816

terminal control *(continued)*
transmission facilitiesz/Os Communications Server z/OS Communications Server/non-z/OS Communications Server 1816
WAIT request 1818
ZCP and TCP common control routines, illustrated 1821
terminal control table line entry (TCTLE) 1819, 1833
terminal control table system entry (TCTSE) 1710
terminal definition 1512
installing 1826
terminal error program (TEP) 1816, 1836
terminal error recovery 1816
terminal location (DFHZGTI) 1828
terminal location (DFHZLOC) 1828
terminal page scheduling program (TPS) 1492
terminal paging 1470
terminal query transaction 1831
terminal storage, builders 1511
TERMINAL_NOW_UNAVAILABLE function, TFAL gate 325
terminal-owning region (TOR) 1458
terminals
build/delete 1516
Terminals
autoinstall 1453
LU-initiated disconnection flow 1456
TERMINATE function, FCSD gate 281
TERMINATE function, RZTA gate 1033
TERMINATE_CONNECTION function, ISCO gate 549
TERMINATE_DOMAIN function, DMDM gate 417
TERMINATE_EXIT function, PGEX gate 818
TERMINATE_INPUT function, ISRR gate 581
TERMINATE_LE_ENCLAVE function, LEPT gate 289
TERMINATE_PTHREAD function, LEPT gate 289
TERMINATE_RECOVERY function, RMLN gate 970
TERMINATE_TRANSACTION function, PGXM gate 866
TEST_CONNECTION function, RSSR gate 1018
TEST_EVENT function, EMEM gate 532
TEST_FILE_USER function, FCFR gate 216
TEST_LOCK_OWNER function, LMLM gate 684
TEST_USER function, FCFL gate 160
testing facility, and sequential access method (BSAM) 1817
TFAL gate
ALLOCATE function 309
CANCEL_AID function 310
CANCEL_AIDS_FOR_CONNECTION function 310
CANCEL_AIDS_FOR_TERMINAL function 311
TFAL gate *(continued)*
CANCEL_SPECIFIC_AID function 312
CHECK_TRANID_IN_USE function 312
DISCARD_AIDS function 312
FIND_TRANSACTION_OWNER function 313
GET_MESSAGE function 313
INITIALIZE_AID_POINTERS function 313
INQUIRE_ALLOCATE_AID function 314
LOCATE_AID function 314
LOCATE_REMDEL_AID function 315
LOCATE_SHIPPABLE_AID function 315
MATCH_TASK_TO_AID function 315
PURGE_ALLOCATE_AIDS function 316
RECOVER_START_DATA function 316
REMOTE_DELETE function 316
REMOVE_EXPIRED_AID function 317
REMOVE_EXPIRED_REMOTE_AID function 318
REMOVE_MESSAGE function 318
REMOVE_REMOTE_DELETES function 318
REROUTE_SHIPPABLE_AIDS function 319
RESCHEDULE_BMS function 319
RESET_AID_QUEUE function 320
RESTORE_FROM_KEYPOINT function 320
RETRIEVE_START_DATA function 321
SCHEDULE_BMS function 321
SCHEDULE_START function 322
SCHEDULE_TDP function 324
SLOWDOWN_PURGE function 324
TAKE_KEYPOINT function 324
TERM_AVAILABLE_FOR_QUEUE function 325
TERMINAL_NOW_UNAVAILABLE function 325
UNCHAIN_AID function 325
UPDATE_TRANNUM_FOR_RESTART function 325
TFBF gate
BIND_FACILITY function 326
TFIQ gate
INQUIRE_MONITOR_DATA function 326
INQUIRE_TERMINAL_FACILITY function 328
SET_TERMINAL_FACILITY function 330
TFP (transaction failure program) 1842
TFP/ACP (transaction failure program) functions 1842
TFRF gate
RELEASE_FACILITY function 331
TFUS gate
CREATE_USER_EXTENSION function 331
DELETE_USER_EXTENSION function 332
TI (Timer domain) domain 1175
TIDY_UP_FILE function, FCBU gate 103
TIE (task interface element) 1782, 1800, 1806
time-dependent task synchronization 1733
time-of-day
retrieval 1733
time-of-day control 1733
Timer domain (TI) domain 1175
TIMF gate
CONVERT_TIME function 1175
FORMAT_TIME function 1176
INQUIRE_TIME function 1178
TIOA (terminal input/output area) 1833
3270 mapping (M32) 1484
LU1 printer with extended attributes mapping program (ML1) 1483
TISR gate
CANCEL function 1180
INQUIRE_EXPIRATION_TOKEN function 1180
NOTIFY function 1183
REQUEST_NOTIFY_INTERVAL function 1181
REQUEST_NOTIFY_TIME_OF_DAY function 1182
TLT (terminal list table)
message switching 1748
TMP (table management program) 1794, 1829
browse token 1797
control blocks, illustrated 1799
hash table 1794
quiesce 1798
range table 1795
read locks 1797
secondary index 1795
token, browse 1797
token browse, table manager 1797
token, browse 7
tokens, domain call 6
TOM (terminal output macro)
partition handling program (PHP) 1487
terminal page processor (TPP) 1489
terminal page retrieval program (TPR) 1492
TOR (terminal-owning region) 1458, 1855
ALLOCATE processing in 1856
APPC command processing in 1857
ATTACH processing in 1855
FREE processing in 1857
LU6.2 command processing in 1857
TPE (terminal partition extension)
partition handling program (PHP) 1487
TPEND 1897
TPP (terminal page processor) 1488
3270 mapping (M32) 1484
data stream build (DSB) 1478

TPP (terminal page processor) *(continued)*
interfaces, illustrated 1489
LU1 printer with extended attributes mapping program (ML1) 1483
storage control 1489
temporary-storage control 1489
terminal output macro (TOM) 1489
terminal type parameter 1489
TPQ (undelivered messages clean up program) 1490
allocation program 1490
interfaces, illustrated 1490
interval control 1490
mapping control program (MCP) 1490
storage control 1490
temporary-storage control 1490
transient data control 1490
TPR (terminal page retrieval program) 1491
interfaces, illustrated 1491
interval control 1492
mapping control program (MCP) 1492
program manager 1491
storage control 1492
task control 1492
temporary-storage control 1492
terminal control 1492
terminal output macro (TOM) 1492
transient data control 1492
TPS (terminal page scheduling program) 1492
TR (Trace domain) domain 1184
trace
CICS-DB2 1528
trace control
data interchange program (DIP) 1547
trace control macro-compatibility interface 1836
Trace domain (TR) domain 1184
trace formatting 1838
trace formatting control area (TRFCA) 1841
trace point IDs
AP 00CB 1784
AP 00CD 1493
AP 00CF 1493
AP 00D7 1547
AP 00DC 1844
AP 00DE 1781
AP 00DF 1721
AP 00E0 1747
AP 00E1 1583
AP 00E2 1764
AP 00E3 1794
AP 00E6 1814, 1835
AP 00E7 1807, 1810
AP 00EA 1800
AP 00EB 1764
AP 00EC 1764
AP 00EF 1764
AP 00F2 1758
AP 00F3 1734
AP 00F6 1871
AP 00FA 1493
AP 00FB 1520
trace point IDs *(continued)*
AP 00FC 1517, 1835
AP 00FE 1837
AP 00FF 1837
AP 03xx 1563
AP 04xx 1696
AP 0701 1792
AP 0702 1792
AP 0780 1792
AP 0781 1792
AP 0782 1792
AP 0783 1792
AP 0790 1792
AP 0791 1792
AP 0792 1792
AP 0793 1792
AP 0794 1792
AP 0795 1792
AP 0796 1792
AP 0797 1792
AP 0798 1792
AP 0799 1792
AP 079A 1792
AP 08xx 1866
AP 0Bxx 1696
AP 0Cxx 1768
AP 0F0x 1468
AP 0F1x 1468
AP 23xx 1696
AP 24xx 1696
AP A500 1583
AP A501 1583
AP D5xx 1877
AP D8xx 1784
AP D9xx 1721
AP DBxx 1866
AP DDxx 1732
AP E110 1583
AP E111 1583
AP F1xx 1773
AP F2xx 1758
AP F6xx 1871
AP FBxx 1912
AP FC71 1755
AP FC72 1755
AP FCxx 1462, 1753, 1835
AP FD7E 1753
AP FDxx 1559, 1835
AP FExx 1559, 1835
AP FF0x 1878
RE trace points 1462
UE trace points 1462
WB xxxx 1532
trace utility program 1839
TRACE_CONTAINERS function, PGCR gate 811
TRACE_PUT function, TRFT gate 1184
TRACE_PUT function, TRPT gate 1185
TRACE_PUT_CQ function, CQCQ gate 78
TRACKING_TAG 736
TRANDEF_DELETE_QUERY function, XMDN gate 1400
TRANDEF_NOTIFY function, XMDN gate 1401
transaction initiation, automatic (ATI) 1867
transaction manager domain (XM) 1335
transaction restart program, DFHREST 1844
transaction routing 1517, 1846
data streams for 1860
DFHAPRT 1846
for APPC devices, LU6.2 1865
relay transaction 1846
shipping TCTTE for 1458
surrogate TCTTE 1864
TRANSACTION_BIND function, MNAC gate 720
TRANSACTION_BIND function, MNXM gate 736
TRANSACTION_DUMP function, DUDU gate 504
TRANSACTION_HANG function, XMAC gate 1399
TRANSACTION_INITIALISATION function, KEXM gate 609
TRANSACTION_INITIALISATION function, MNXM gate 737
TRANSACTION_TERMINATION function, MNAC gate 720
TRANSACTION_TERMINATION function, MNXM gate 737
transaction-routed data format 1861
transactions
CATA 1495, 1499, 2046
CATD 2046
CATR 2048
CEBR 1583
CECI 1532
CECS 1532
CEDA 1761
CEDB 1761
CEDC 1761
CEMT 1746, 1768
CEOT 1746
CEST 1746
CITS 2048
CMPX 1715
CMSG 1747, 1749
CPMI 1708
CSFE 1588
CSGM 1722
CSM1 1708
CSM2 1708
CSM3 1708
CSM5 1708
CSMI 1708
CSNC 1729
CSNE 1750
CSPG 1491
CSPQ 1481, 1490
CSPS 1481
CSZI 1697
CVMI 1708
CXRT 1856
mirror 1708
TRANSFER_SIT function, APAP gate 23
TRANSFORM_JSON function, MLJS gate 705
TRANSFORM_STRUCTURE_TO_XML function, MLTF gate 708
TRANSFORM_XML_TO_STRUCTURE function, MLTF gate 709

transformations of data for function shipping 1711
transformer program (DFHXTP) 1859
transient data control
  abnormal condition program (ACP) 1843
  mapping control program (MCP) 1481
  terminal page retrieval program (TPR) 1492
  undelivered messages clean up program (TPQ) 1490
transient data program (TDP)
  deferred work element (DWE) 1866
transient data services 1868
translator, command-language 1759
TRFCA (trace formatting control area) 1841
TRFT gate
  TRACE_PUT function 1184
TRIGGER_PARTNER function, PTTW gate 927
TRPT gate
  TRACE_PUT function 1185
TRSR gate
  ACTIVATE_TRAP function 1186
  DEACTIVATE_TRAP function 1186
  INQUIRE_AUXILIARY_TRACE function 1186
  INQUIRE_GTF_TRACE function 1187
  INQUIRE_INTERNAL_TRACE function 1188
  PAUSE_AUXILIARY_TRACE function 1188
  SET_AUX_TRACE_AUTOSWITCH function 1188
  SET_INTERNAL_TABLE_SIZE function 1189
  START_AUXILIARY_TRACE function 1189
  START_GTF_TRACE function 1190
  START_INTERNAL_TRACE function 1190
  STOP_AUXILIARY_TRACE function 1190
  STOP_GTF_TRACE function 1191
  STOP_INTERNAL_TRACE function 1191
  SWITCH_AUXILIARY_EXTENTS function 1191
TRUST_CLIENT function, PITC gate 903
TS (Temporary Storage Domain) domain 1193
TSAD gate
  ADD_REPLACE_TSMODEL function 1193
  DELETE_TSMODEL function 1194
  INITIALISE function 1195
TSBR gate
  CHECK_PREFIX function 1195
  END_BROWSE function 1195
  GET_NEXT function 1196
  INQUIRE_QUEUE function 1197
  START_BROWSE function 1198
TSMB gate
  END_BROWSE function 1198
TSMB gate *(continued)*
  GET_NEXT function 1198
  INQUIRE_EXPINT function 1201
  INQUIRE_TSMODEL function 1200
  MATCH function 1201
  START_BROWSE function 1202
TSPT gate
  GET function 1203
  GET_RELEASE function 1203
  GET_RELEASE_SET function 1204
  GET_SET function 1204
  PUT function 1205
  PUT_REPLACE function 1206
  RELEASE function 1206
TSQR gate
  ALLOCATE_SET_STORAGE function 1207
  DELETE function 1207
  READ_INTO function 1208
  READ_NEXT_INTO function 1209
  READ_NEXT_SET function 1210
  READ_SET function 1211
  REWRITE function 1212
  WRITE function 1213
TSRM gate
  DELIVER_IC_RECOVERY_DATA function 1226
  INQUIRE_QUEUE function 1214
  SOLICIT_INQUIRES function 1227
TSSH gate
  ADD_POOL function 1214
  DELETE function 1214
  END_BROWSE function 1215
  END_TSPOOL_BROWSE function 1215
  GET_NEXT function 1216
  GET_NEXT_TSPOOL function 1216
  INITIALISE function 1217
  INQUIRE_POOL_TOKEN function 1217
  INQUIRE_QUEUE function 1218
  INQUIRE_SYSID_TABLE_TOKEN function 1219
  INQUIRE_TSPOOL function 1219
  READ_INTO function 1220
  READ_NEXT_INTO function 1220
  READ_NEXT_SET function 1221
  READ_SET function 1222
  REWRITE function 1222
  START_BROWSE function 1223
  START_TSPOOL_BROWSE function 1224
  WRITE function 1224
TSSR gate
  SET_BUFFERS function 1225
  SET_START_TYPE function 1225
  SET_STRINGS function 1225
TTP (terminal type parameter)
  illustration of 1472
  terminal page processor (TPP) 1489
two-phase commit process 1808
type 3 SVC routine
  used for interregion communication 1722
TYPE_PURGE function, CCCC gate 371

U

UEH (user exit handler) 1874
UEM (user exit manager) 1873
UET (user exit table) 1805, 1872
UIB (user interface block) 1561
UNBIND_LDAP function, DDAP gate 379
UNCHAIN_AID function, TFAL gate 325
UNFLATTEN_ESM_UTOKEN function, XSFL gate 1425
UNFLATTEN_REQUEST function, APRX gate 46
UNFLATTEN_RESPONSE function, APRX gate 46
UNFLATTEN_TRANSACTION_USER function, USXM gate 1250
UNFLATTEN_USER function, USFL gate 1243
UNFLATTEN_USER_SECURITY function, XSFL gate 1426
UNINSTALL_OSGIBUNDLE function, SJTH gate 1062
unit of recovery descriptor (URD) 1782, 1807
UNLOCK function, FCCR gate 122
UNLOCK function, FCFR gate 216
UNLOCK function, FCRF gate 274
UNLOCK function, LMLM gate 684
UNLOCK_TCLASS function, XMCL gate 1354
UNLOCK_TERM_MODEL function, AIIQ format 1464
UNLOCK_TRANDEF function, XMLD gate 1376
UNREGISTER_CONTROL_ACB function, FCCA gate 112
UOW_TIME function, LGGL gate 658
UPDATE option of READ request
  backout processing 1594
  exclusive control of control interval 1594
  processing 1594
UPDATE_ABEND_RECORD function, ABAB gate 19
UPDATE_FEATURE function, DUFT gate 509
UPDATE_FILE function, FCMT gate 254
UPDATE_PASSWORD function, XSPW gate 1441
UPDATE_PROFILE_IN_LIST function, DPLM gate 440
UPDATE_RECOVERY_POINTS function, FCDN gate 145
UPDATE_REVOCATION_LIST function, SORL gate 1156
UPDATE_SUBSPACE_TCB_INFO function, SMSR gate 1088
UPDATE_TRANNUM_FOR_RESTART function, TFAL gate 325
UPDATE_WEBSERVICE function, PISC gate 899
URD (unit of recovery descriptor) 1782, 1807
URL_DECODE function, WBFM gate 1283
US (User domain) domain 1229

USAD gate
ADD_USER_VIA_ICRX function 1239
ADD_USER_WITH_PASSWORD function 1229
ADD_USER_WITHOUT_PASSWORD function 1231
DELETE_USER function 1233
GET_ASSOCIATED_DATA_LIST function 1242
ICRX_TO_USERID function 1241
INQUIRE_DEFAULT_USER function 1234
INQUIRE_ICRX function 1240
INQUIRE_USER function 1236
NOTIFY_USERID function 1238
RELEASE_ICRX function 1241
VALIDATE_USERID function 1238
User domain (US) domain 1229
user exit control 1871
user exit handler (UEH) 1874
user exit interface
control blocks, illustrated 1875
user exit manager (UEM) 1873
user exit service module (DFHAPEX) 1875
user exit subroutine (DFHSUEX) 1875
user exit table (UET) 1805, 1872
user exits
"good morning" message program 1722
CICS-DB2 1528
data tables 1696
database control (DBCTL) 1546
DFHCSDUP 1534
DL/I database support 1563
exec interface 1583
file control 1696
function shipping 1721
interval control 1734
program control 1757
system recovery program 1792
terminal control 1835
transient data control 1871
user interface block (UIB) 1561
USFL gate
FLATTEN_USER function 1242
TAKEOVER function 1243
UNFLATTEN_USER function 1243
USIS gate
INQUIRE_DOMAIN function 1245
SET_USER_DOMAIN_PARMS function 1244
USXM gate
ADD_TRANSACTION_USER function 1245
DELETE_TRANSACTION_USER function 1246
END_TRANSACTION function 1246
FLATTEN_TRANSACTION_USER function 1246
INIT_TRANSACTION_USER function 1247
INQUIRE_TRANSACTION_USER function 1248
TERM_TRANSACTION_USER function 1250
USXM gate *(continued)*
UNFLATTEN_TRANSACTION_USER function 1250

V

VALIDATE function, PITC gate 902
VALIDATE_APPC_RESPONSE function, XSLU gate 1437
VALIDATE_CIPHERS function, SOIS gate 1146
VALIDATE_LANGUAGE_CODE function, MEME gate 694
VALIDATE_LANGUAGE_SUFFIX function, MEME gate 695
VALIDATE_USERID function, USAD gate 1238
VALIDATE_USERID function, XSAD gate 1412
VERIFY_CGCSGID function, CCNV gate 70
VERIFY_CICS_CCSID function, CCNV gate 72
VERIFY_IANA_CCSID function, CCNV gate 73
VERIFY_IBM_CCSID function, CCNV gate 74
VERIFY_IP_ADDRESS function, SOIS gate 1146
vertical tabs
and device independence 1471
virtual storage access method (VSAM) 1588
Virtual Telecommunications Method (VTAM) 1814
Virtual Telecommunications Method (z/OS Communications Server)
and node abnormal condition program (NACP) 1750
and node error program (NEP) 1754
VSAM
ENDBR request processing 1600
READ request processing 1594
READNEXT request processing 1599
READPREV request processing 1599
RESETBR request processing 1599
REWRITE request processing 1597
STARTBR request processing 1599
UNLOCK request processing 1598
WRITE request processing 1595
VSAM (virtual storage access method) 1588
VSAM interface program, file control (DFHFCVR) 1648
VSAM request processor, file control (DFHFCVS) 1649
VSAM work area (VSWA) 1601, 1612
VSAM, processing using 1592
VSWA (VSAM work area) 1601, 1612
FREEMAIN during DELETE processing 1599
FREEMAIN during UNLOCK request processing 1598
FREEMAIN in ENDBR request processing 1600
READ request 1594
WRITE request 1595

W

W2 (Web 2.0) domain 1322
W2AT gate
ADD_ATOMSERVICE function 1323
ADD_REPLACE_ATOMSERVICE function 1325
DELETE_ATOMSERVICE function 1326
END_BROWSE_ATOMSERVICE function 1327
GET_NEXT_ATOMSERVICE function 1328
INITIALIZE_ATOMSERVICES function 1329
INQUIRE_ATOMSERVICE function 1330
SET_ATOMSERVICE function 1332
START_BROWSE_ATOMSERVICE function 1332
W2W2 gate
HANDLE_ATOM_REQUEST function 1333
SET_PARAMETERS function 1334
WAIT request
terminal control 1818
WAIT_FOR_FILE_CONTROL function, FCIN gate 226
WAIT_FOR_PARTNER function, PTTW gate 928
WAIT_MVS function, DSSR gate 484
WAIT_OLDC function, DSSR gate 486
WAIT_OLDW function, DSSR gate 487
WAIT_PHASE function, DMDM gate 411
WAITPRED function, KEAR gate 583
warm start 1516
WB (Web) domain 1252
WBAP gate
END_BROWSE function 1252
GET_HTTP_RESPONSE function 1253
GET_MESSAGE_BODY function 1253
INITIALIZE_TRANSACTION function 1254
INQUIRE function 1255
QUERY_URIMAP function 1256
READ function 1257
READ_NEXT function 1258
SEND_RESPONSE function 1259
START_BROWSE function 1260
WRITE_HEADER function 1261
WBCL gate
CLOSE_SESSION function 1262
END_BROWSE_HEADERS function 1263
INQUIRE_SESSION function 1264
OPEN_SESSION function 1266
PARSE_URL function 1268
READ_HEADER function 1270
READ_NEXT_HEADER function 1271
READ_RESPONSE function 1273
START_BROWSE_HEADERS function 1275
WRITE_HEADER function 1276
WRITE_REQUEST function 1278

WBFM gate
PARSE_MULTIPART_FORM function 1280
PARSE_URL_ENCODED_FORM function 1281
PARSE_URL_ENCODED_LIST function 1282
URL_DECODE function 1283
WBSR gate
PEEK function 1284
RECEIVE function 1284
SEND function 1286
SEND_STATIC_RESPONSE function 1287
WBSV gate
CLOSE_SESSION function 1293
INQUIRE_CURRENT_SESSION function 1291
INQUIRE_SESSION function 1293
PEEK_HEADERS function 1290
READ_REQUEST function 1288
SET_SESSION function 1292
WRITE_RESPONSE function 1289
WBUR gate
ADD_REPLACE_URIMAP function 1294
DELETE_URIMAP function 1297
END_BROWSE_HOST function 1298
END_BROWSE_URIMAP function 1298
FIND_URIMAP function 1299
GET_NEXT_HOST function 1302
GET_NEXT_URIMAP function 1303
INITIALIZE_URIMAPS function 1306
INQUIRE_HOST function 1307
INQUIRE_URIMAP function 1308
LOCATE_URIMAP function 1312
SET_HOST function 1315
SET_URIMAP function 1316
START_BROWSE_HOST function 1319
START_BROWSE_URIMAP function 1320
WEAK_JOIN function, RZSO gate 1029
Web (WB) domain 1252
Web 2.0 (W2) domain 1322
WRITE function, CCCC gate 371
WRITE function, FCCR gate 123
WRITE function, FCFR gate 218
WRITE function, FCRF gate 275
WRITE function, LGGL gate 658
WRITE function, LGWF gate 678
WRITE function, TSQR gate 1213
WRITE function, TSSH gate 1224
WRITE_ADD_COMPLETE function, FCLJ gate 232, 233
WRITE_DELETE function, FCLJ gate 235
WRITE_HEADER function, WBAP gate 1261
WRITE_HEADER function, WBCL gate 1276
WRITE_HFS_FILE function, DHFS gate 396
WRITE_JNL function, LGGL gate 659
WRITE_JOURNAL_DATA function, APJC gate 30
WRITE_NEXT function, CCCC gate 372
WRITE_REQUEST function, WBCL gate 1278
WRITE_RESPONSE function, WBSV gate 1289
WRITE_TRANSIENT_DATA function, APTD gate 53
WRITE_UPDATE function, FCLJ gate 235
WTO macro 1877
WTOR macro 1877

X

XA_RESYNC function, ISRE gate 578
XFCREQ, global user exit
READ request 1593
WRITE request 1595
XLN (exchange log name) 1887
XM (transaction manager) domain 1335
XMAC gate
ABEND_TERMINATE function 1398
INIT_XM_CLIENT function 1398
RELEASE_XM_CLIENT function 1399
TRANSACTION_HANG function 1399
XMAT gate
ATTACH function 1336, 1398
BUILD_TRANSACTION function 1340
REATTACH function 1339
TERMINATE_TRANSACTION function 1343
XMBD gate
END_BROWSE_TRANDEF function 1343
GET_NEXT_TRANDEF function 1344
START_BROWSE_TRANDEF function 1348
XMCL gate
ADD_REPLACE_TCLASS function 1348
ADD_TCLASS function 1349
DELETE_TCLASS function 1349
DEREGISTER_TCLASS_USAGE function 1350
END_BROWSE_TCLASS function 1350
GET_NEXT_TCLASS function 1350
INQUIRE_ALL_TCLASSES function 1351
INQUIRE_TCLASS function 1351
LOCATE_AND_LOCK_TCLASS function 1352
REGISTER_TCLASS_USAGE function 1353
SET_TCLASS function 1353
START_BROWSE_TCLASS function 1354
UNLOCK_TCLASS function 1354
XMDD gate
DELETE_TRANDEF function 1355
XMDN gate
TRANDEF_DELETE_QUERY function 1400
TRANDEF_NOTIFY function 1401
XMER gate
ABEND_TRANSACTION function 1355
INQUIRE_DEFERRED_ABEND function 1355
INQUIRE_DEFERRED_MESSAGE function 1356
REPORT_MESSAGE function 1357
SET_DEFERRED_ABEND function 1358
SET_DEFERRED_MESSAGE function 1358
XMFD gate
FIND_PROFILE function 1359
XMIQ gate
END_BROWSE_TRANSACTION function 1360
END_BROWSE_TXN_TOKEN function 1360
GET_NEXT_TRANSACTION function 1360
GET_NEXT_TXN_TOKEN function 1365
INQUIRE_TRANSACTION function 1365
INQUIRE_TRANSACTION_TOKEN function 1370
PURGE_TRANSACTION function 1371
SET_TRANSACTION function 1372
SET_TRANSACTION_TOKEN function 1373
START_BROWSE_TRANSACTION function 1374
START_BROWSE_TXN_TOKEN function 1375
XMLD gate
LOCATE_AND_LOCK_TRANDEF function 1376
UNLOCK_TRANDEF function 1376
XMNT gate
MXT_CHANGE_NOTIFY function 1401
MXT_NOTIFY function 1401
XMPP gate
FORCE_PURGE_INHIBIT_QUERY function 1402
XMRU gate
RUN_TRANSACTION function 1377
XMSR gate
INQUIRE_DTRTRAN function 1377
INQUIRE_MXT function 1378
SET_DTRTRAN function 1378
SET_MXT function 1379
XMXD gate
ADD_REPLACE_TRANDEF function 1379
INQUIRE_REMOTE_TRANDEF function 1384
INQUIRE_TRANDEF function 1389
SET_TRANDEF function 1394

XMXE gate
FREE_TXN_ENVIRONMENT function 1396
GET_TXN_ENVIRONMENT function 1397
XRF 1897
XRF_GET function, XSWM gate 332
XRF_PUT function, XSWM gate 333
XS (Security domain) domain 1404
XSAD gate
ADD_USER_VIA_ICRX function 1412
ADD_USER_WITH_PASSWORD function 1404
ADD_USER_WITHOUT_PASSWORD function 1406
DELETE_USER_SECURITY function 1408
INQUIRE_ICRX function 1413
INQUIRE_USER_ATTRIBUTES function 1409
RELEASE_ICRX function 1414
RELEASE_ICRX_STORAGE function 1414
VALIDATE_USERID function 1412
XSCT gate
INQUIRE_CERTIFICATE function 1415
INQUIRE_REVOCATION_LIST function 1417
XSEJ gate
ADD_REPL_ROLE_FOR_METHOD function 1418
CHECK_CALLER_IN_ROLE function 1418
CHECK_EJB_METHOD function 1419
DELETE_BEAN_SECURITY function 1420
INQUIRE_DISTINGUISHED_NAME function 1420
INQUIRE_HASH_CODE function 1422
INQUIRE_PRINCIPAL function 1422
SET_ROLE_FOR_CODED_ROLE function 1423
XSFL gate
FLATTEN_USER_SECURITY function 1424
UNFLATTEN_ESM_UTOKEN function 1425
UNFLATTEN_USER_SECURITY function 1426
XSIS gate
INQ_SECURITY_DOMAIN_PARMS function 1427
INQUIRE_REALM_NAME function 1430
INQUIRE_REGION_USERID function 1431
SET_NETWORK_IDENTIFIER function 1431
SET_SECURITY_DOMAIN_PARMS function 1432
SET_SPECIAL_TOKENS function 1435
XSLU gate
GENERATE_APPC_BIND function 1435
GENERATE_APPC_RESPONSE function 1436
VALIDATE_APPC_RESPONSE function 1437
XSPW gate
CREATE_PASSTICKET function 1438
INQUIRE_CERTIFICATE_USERID function 1438
INQUIRE_PASSWORD_DATA function 1439
REGISTER_CERTIFICATE_USER function 1440
UPDATE_PASSWORD function 1441
XSRC gate
CHECK_CICS_COMMAND function 1442
CHECK_CICS_RESOURCE function 1445
CHECK_NON_CICS_RESOURCE function 1446
CHECK_SURROGATE_USER function 1447
REBUILD_RESOURCE_CLASSES function 1448
XSRN gate
RECORD_NONCOMPLIANCE function 1448
XSWM gate
XRF_GET function 332
XRF_PUT function 333
XSXM gate
ADD_TRANSACTION_SECURITY function 1449
DEL_TRANSACTION_SECURITY function 1449
END_TRANSACTION function 1450
XWBAUTH 1322, 1531
XWBOPEN 1531
XWBSNDO 1532

Z

z/OS Communications Server
control blocks
ACBs 1703
NIBs 1703
RPLs 1703
persistent sessions support 1895, 1897
transmission facilities 1816
z/OS Communications Server asynchronous receive exit (DFHZASX) 1456
z/OS Communications Server asynchronous send exit (DFHZSAX) 1456
z/OS Communications Server exits 1462, 1911
z/OS Communications Server generic resource 1878
z/OS Communications Server LU6.2 1885
z/OS Communications Server persistent sessions
diagnosing problems 1905
z/OS Communications Serverthe Communications Server
persistent sessions support 1897

