Publication Updates for OA61850: z/OS OPEN, STOW and CLOSE SVC Installation Exits V2R5

Document Name:	OA61850.pdf
Document Owner:	Cecilia Carranza Lewis (carranc@us.ibm.com)
Version:	V1.0

About this information

This document introduces the DFSMS OPEN, STOW and CLOSE SVC Installation Exits support available with APARs OA61850 and OA61849, in addition to specific updates for certain publications in the z/OS® product library, as required by the APAR OA61850 package. The information consists of pages excerpted from the respective publications or new information added to the respective publications. The information in this document applies to V2R5.

Currency of this information: The complete publication updates will appear in the next editions of the official publications. Thereafter, the information in the official publications supersedes the publication information in this APAR document. Preface

This document describes three new dynamic installation exit points for OPEN, STOW and CLOSE. The following publications will be updated to reflect this support:

- *z/OS DFSMS Installation Exits*
- *z/OS DFSMS Macro Instructions for Data Sets*
- *z/OS DFSMSdfp Advanced Services*
- *z/OS DFSMSdfp Diagnosis*
- *z/OS MVS Installation Exits*
- *z/OS MVS System Messages Volume 7 (IEB-IEE)*

Table of Contents

Table of Figures

1.0 Introduction

This solution provides a documented interface that products from IBM and others can plug into to examine each call to the OPEN, STOW and CLOSE SVCs. They are:

- Near the beginning of the OPEN SVC, which is SVCs 19 and 22. The latter is for TYPE=J, where the caller is providing a JFCB update.
- STOW (SVC 21). Programs that perform operations on PDS or PDSE members normally issue this macro, but the binder does not use STOW for program objects. Program objects reside only in PDSEs. For program objects, the binder uses the DESERV FUNC=PUT macro. DESERV calls a different installation exit that is dynamically installed but it does not use the CSVDYNEX macro. See *z/OS DFSMSdfp Advanced Services*.
- CLOSE (SVC 20). This does not include CLOSE TYPE=T (SVC 23) because it is not really closing the data set; it is only repositioning access to the data set.

Calls to the close and open exit routines also are in EOV to handle certain types of sequential concatenation.

The purposes of the exits typically would be to monitor activity for certain data sets.

This solution uses the existing CSVDYNEX macro to allow more than one exit routine to be called at each exit point.

IBM recommends that products that intercept SVC instructions for OPEN, STOW or CLOSE be converted to use these new dynamic exits functions.

2.0 Publications Updates

2.1 z/OS DFSMS Installation Exits

Chapter 2. Data Management Installation/Dynamic Exits will be enhanced to provide the details regarding the three new exits.

These entries will be added to table 2, “Data Management Dynamic Exits”:

Module Name	Description	When Available
IFG_CLOSE_START	Early CLOSE SVC exit.	Early in the CLOSE SVC
IFG_OPEN_START	Early OPEN SVC exit.	Early in the OPEN SVC
IGG_STOW_START	Early STOW SVC exit.	Early in the STOW SVC

2.1.1 OPEN/CLOSE/EOV and Access Method SVC Exits

The OPEN, STOW and CLOSE SVCs invoke these SVC installation exits:

- IFG_OPEN_START. Near the beginning of the OPEN SVC, which is SVCs 19 and 22. SVC 22 is for OPEN TYPE=J, where the caller is providing a JFCB update. The system also calls this exit when processing a sequential concatenation. This call is when the application program begins reading another data set; the call is not when the application program issues the OPEN macro. For a partitioned concatenation this exit is called only when the application program issues the OPEN macro.
- IGG_STOW_START. Near the beginning of STOW, SVC 21. Programs that perform operations on PDS or PDSE members normally issue this macro, but the binder does not use STOW for program objects. Program objects reside only in PDSEs. For program objects, the binder uses the DESERV FUNC=PUT macro. DESERV calls a different installation exit that is dynamically installed but it does not use the CSVDYNEX macro. See *z/OS DFSMSdfp Advanced Services*.
- IGG_CLOSE_START (SVC 20). Near the beginning of CLOSE SVC 20. This does not include CLOSE TYPE=T (SVC 23) because it is not really closing the data set; it is only repositioning access to the data set. The system also calls this exit when processing a sequential concatenation. In a sequential concatenation, this call is when the application program finishes reading a data set other than the last data set; later this exit is called when the application program issues the CLOSE macro or task termination closes the data set.

These exits use the dynamic exits service, CSVDYNEX, which is described in *z/OS MVS Programming: Authorized Assembler Services Guide* and *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*. The dynamic exits facility allows multiple exit routines to be simultaneously defined to a single exit point. Using this facility, the system programmer can associate new exit routines with these exit points. IBM does not provide default exit routines for these exits.

(z/OS also supports an installation exit called IFG0EX0B. It is not dynamic. See “DCB Open Installation Exit (IFG0EX0B)”.)

The purposes of the exits might be to monitor activity for certain data sets or to enforce restrictions.

These exits were implemented via a PTF to z/OS 2.5. If the DFAOCEExits bit in the DFAFEAT11 byte in the DFA is on, it means that the dynamic exit support for open, STOW and close is installed and available. See the IHADFA macro.

2.1.1.1 General Programming Considerations

The exit routines run in task mode, protection key 5 and supervisor state. As with all subroutines, do not bind installation exits with APF authorization.

They are entered with AMODE 31, enabled for interrupts. Primary=Home=Secondary. Primary ASC mode.

The parameter list passed to the exit routines resides in non fetch protected, key 5 storage. The only fetch-protected area that it points to is the user’s DCB.

Although the service will not enforce reentrancy, it is recommended that the routine be reentrant.

The exit routines must not dequeue SYSZTIOT (in open or close) or unlock the DEB (in STOW or close). Doing so will have unpredictable results.

The caller of the exit routine will provide ESTAE protection. This is to support the FASTPATH option. In the event of an abend in open or close, the ESTAE recovery routine will issue CSVDYNEX REQUEST=RECOVER to clean up and write an IEC997I message and otherwise ignore this problem. Resources acquired by the exit routine will be lost unless the exit routine uses ESTAE to recover. An abend in an exit routine will not affect calls to other exit routines at that exit point and it will not affect that user program's call to the SVC. See further comments about abends later.

2.1.1.2 Controlling the OPEN/CLOSE/EOV and Access Method SVC Exits through the Dynamic Exits Facility

Installations can associate their own exit routines with these exit points. See "Adding an Exit Routine to an Exit" in *z/OS MVS Programming: Authorized Assembler Services Guide* for details. The dynamic exits facility allows multiple exit routines to be defined to a single exit.

The dynamic exit service also is described under CSVDYNEX in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN.*

Operator commands and system services that apply to managing dynamic exits apply to these dynamic exits. For example, to add an exit routine to a dynamic exit, you can use the following services:

- A program can use CSVDYNEX REQUEST=ADD macro. This is an example:

```
            CSVDYNEX REQUEST=ADD,EXITNAME=ONAME,MODNAME=OModule,                *
                  STATE=ACTIVE,DSNAME=ODSNAME,ADDABENDNUM=10,PARAM=YES,         *
                  SERVICEMASK=OSMask
. . .
ONAME     DC    CL16'IFG_OPEN_START'
OModule   DC    CL8'OPENEX1'
ODSNAME   DC    CL44'DEPT.MAIN.LIBRARY'
OSMask    DC    AD(B'1')              1 padded on left with 63 zeroes
Yes       DC    CL8'YES'
```

- The operator can issue the SETPROG EXIT command. This is an example:

```
SETPROG EXIT,ADD,EXITNAME=IFG_OPEN_START,MODNAME=OPENEX1,STATE=(ACTIVE),
DSNAME=DEPT.MAIN.LIBRARY,ADDABENDNUM=10,PARAM=YES,SERVICEMASK=1
```

  The module OPENEX1 is a member in data set DEPT.MAIN.LIBRARY. The number of abends in the exit routine that the system will tolerate before making it inactive is ten.

- The system programmer can add an EXIT statement in a PROG*xx* PARMLIB member and then issue the SET PROG=xx operator command. This is an example of the EXIT statement:

```
EXIT ADD EXITNAME(IFG_OPEN_START) MODNAME(OPENEX1) STATE(ACTIVE)
     DSNAME(DEPT.MAIN.LIBRARY) ADDABENDNUM(10) PARAM(YES) SERVICEMASK(1)
```

These are among the options that you can specify:

- ADDABENDNUM. Due to the FASTPATH=YES option on CSVDYNEX REQUEST=DEFINE and on CSVDYNEX REQUEST=CALL, if an abend occurs in an exit routine, the ESTAE recovery routine for open, STOW or close will get control. It then will write message IEC997I and issue CSVDYNEX REQUEST=RECOVER so that CSVDYNEX can call any exit routines that have not yet been called for this instance of OPEN, STOW or CLOSE.

An abend in an exit routine will not affect the other exit routines that are defined for that exit or affect the application program. However, you can specify the number of abends in the exit that the system will tolerate since IPL. If this limit reached, the system will make the exit module inactive. The default for these exits is 2. If you code ADDABENDNUM with CSVDYNEX REQUEST=ADD, it overrides the default value of ABENDNUM=2.

The system programmer might want to use message automation to handle this condition.

- PARAM. You can think of this as being a *latent parameter*. See 2.1.1.4 “Latent Parameters” on page 6.
- SERVICEMASK is a bit string of up to 64 bits that are padded on the left by zeroes. For each invocation of these exits, the system sets a 64-bit service ID (identifier). You can associate a service mask with your exit routine. Only if the ANDed value of the service mask for your exit routine and the service ID for a given call is nonzero will the system call the exit routine. In other words, the system will call the exit routine only if at least one corresponding bit is 1 in both masks. If you do not specify SERVICEMASK, the default is that the system will call this exit routine.

  For IFG_OPEN_START and IFG_CLOSE_START, the rightmost bit means that the system will call this exit routine only for opens of a DCB for a DASD data set. This excludes spooled data sets, subsystem data sets and z/OS UNIX files and directories even though they might be on DASD. The only reason to specify a service mask for this case is to prepare for the future when IBM might define more bits to cover more kinds of data set. If you do not expect that you will want the system to call your exit routine for other than DASD non-VSAM data sets, then you should set the rightmost mask bit to 1.

  For purposes of the open and close exits, a z/OS UNIX directory is not regarded as DASD but there is one exception. If it is part of a partitioned concatenation being opened with a BPAM DCB and the concatenation includes at least one PDS or PDSE, then each z/OS UNIX directory will be represented by a pseudo DSCB. In this case:

  The DS1PDSEX bit will be on. This bit normally means that the data set is an HFS file system but BPAM does not support opening such a data set. BPAM supports opening a z/OS UNIX directory.

  The data set name in the DSCB and JFCB is not real.

  For IGG_STOW_START, the rightmost bit means to invoke the exit routine for PDSs. The next higher bit means to invoke the exit routine for PDSEs. For example, if you want the system to call the exit only for PDSEs, code any of these values: 10, 00000010, 000000000000010, etc.

For example, using the SETPROG operator command, the following will add the exit routine to the dynamic exit IFG_OPEN_START:

```
SETPROG EXIT,ADD,EXITNAME=IFG_OPEN_START,MODNAME=OPENSTRT
```

You can have multiple exit routines associated with each dynamic exit. If you do that without specifying FIRST or LAST, the order in which the system calls them is unpredictable.

You can replace already active dynamic exit routine without an IPL. For example, you can issue the SETPROG EXIT,DELETE operator command and the SETPROG EXIT,ADD operator command to replace a dynamic exit routine. For further information about the SETPROG command, see *z/OS MVS System Commands*.

For further information about the CSVDYNEX macro, see *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN.* For further information about the PROG*xx* PARMLIB member, see *z/OS MVS Initialization and Tuning Reference*.

2.1.1.3 Effects of Concatenation

When the user opens a sequential concatenation, the close and open exits will be called for each data set as the user makes the transition to the next data set. This means that the application program did not issue a CLOSE or OPEN macro for this transition.

Application programs cannot close a DCB under a task that differs from the task that opened the DCB but the application program can read the data set in a task that differs from the task that opened the DCB. When making the transition to the next sequential data set or partitioned member, the system can close and re-open the DCB. In this case, the system will call the close and open exits under the reading task. The application program issued an access method macro (CHECK, GET or FEOV) that caused the transition to the next data set or the application program issued an EOV macro for an EXCP DCB.

The DEB and the SVC exit parameter list always contain the address of the TCB that opened the data set.

When opening a partitioned concatenation, the exit will receive a list of DSCBs for the data sets. The exit routines will not be called separately for each data set.

2.1.1.4 Latent Parameters

Your exit routine can use a *latent parameter*. That is eight bytes that you supply with the PARAM option when defining your exit routine to the exit. On each call to the exit, a copy of the first four bytes will be in access register 0 and the second four bytes will be in access register 1. It might be the address of a system-wide control block for that exit routine. The exit supplier can pass these eight bytes with the PARAM keyword on CSVDYNEX REQUEST=ADD.

The parameter lists have no provision for an exit to pass information, such as the address of a work area, to another exit invocation. Your exit can pass information to another of your exits by using the name/token service. You would call IEANTCR to create a name/token pair. In another exit call IEANTRT to retrieve the information and IEANTDL to delete the information. See *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG.*

2.1.1.5 Registers on entry to the OPEN/CLOSE/EOV and access method dynamic exits

Register	Contents
0	Not applicable.
1	Address of the OPEN/CLOSE/EOV and access method dynamic exits parameter list. See Figure 1 ”Parameter list for OPEN/CLOSE/EOV and access method SVC exit routines”.
2-12	Not applicable.
13	Address of a 144-byte register save area.
14	Return address to the system
15	Address of the entry point

2.1.1.6 Registers on exit from the OPEN/CLOSE/EOV and access method dynamic exits

Register	Contents
0, 1	Unpredictable.
2-14	Same as at entry.
15	Return code. A return code of 0 from the exit routine means to continue the OPEN, STOW or CLOSE. A return code of 8 means to fail the call to OPEN, STOW or CLOSE. In the OPEN case, this will result in a specific abend code 013-C1. In the CLOSE case, this will result in a specific abend code 414-18. In the case of STOW, it will result in a return code 32 from the STOW macro. STOW has return codes that are smaller and larger than 32. The STOW reason code will be the exit routine return code, but the user program cannot know which exit routine gave the return code. The person that submitted the job can tell because of the IEC997I message that the system issued to identify the exit routine. The system might support new return codes in a future release. If an abend occurs in the exit routine, it will have no effect on the application program and the system will call other exit routines that are defined for the exit point. If an exit routine is disabled, a SETPROG command can be used to enable or replace it. The exit routines are not expected to change anything that is visible to the system or the user such as the DCB, OPEN parameter list or JFCB.

2.1.1.7 General Programming Considerations

The system invokes these exits with the FASTPATH=YES option. If your exit routine gets an abend, the system's ESTAE recovery routine will write message IEC997I, retry the abend and invoke CSVDYNEX REQUEST=RECOVER. This is so that the system can call any exit routines that the system has not yet called for this instance of OPEN, STOW or CLOSE.

The default value for ABENDNUM for these exits is 2. The system will tolerate this number of abends in a particular exit routine since IPL before the system changes its state to inactive. That means that the system will stop calling it. The system programmer might want to use message automation to handle this condition. When an exit routine is added to the system, the supplier can override ABENDNUM with ADDABENDNUM.

An abend in an exit routine will not affect the other exit routines that are defined for that exit or affect the application program.

The only fetch-protected area that the parameter list points to is the user's DCB. Use the key in the IFGSVCPLCallerKey field.

The exit routine must be reentrant.

2.1.1.8 Environment During IFG_OPEN_START

The system has compared the data set's device class to the SERVICEMASK specification on the CSVDYNEX macro. This tells whether the exit routine is interested in this type of data set, which is DASD.

Before open calls the exit, it has done these things:

- Verified the OPEN parameter list.

- Enqueued shared on SYSZTIOT. This is to serialize access to the DSAB chain, TIOT and XTIOT. This also prevents invocation of dynamic allocation and unallocation in the address space. If the exit dequeues this resource, it endangers the information that the OPEN SVC has gathered. For example, the data set being opened could get unallocated or even scratched and system control blocks could get overlaid.
- Found the control blocks for the specified DD name.
- Read the DSCBs.

If the system is not able to perform all of the above actions, the system will not call the exit routines for that DCB. If multiple DCBs are being opened with one call, the system will call the exit for other DCBs.

The SAF interface has not been called to check security.

The system has not yet merged data set attributes to or from the data set label (DSCB) and has not yet called the user program's DCB open exit routine. That means that the DCB fields are not complete and there is no DEB (data extent block).

The system will call the exits only for DASD data sets that are being opened with a DCB. If multiple DCBs are being opened or closed, the exit will be called independently for each.

It is possible to open a single-volume VSAM data set with an EXCP DCB.

2.1.1.8.1 Effects of Concatenation

When the user opens a sequential concatenation, the close and open exits will be called for each data set as the user makes the transition to the next data set. This means that the application program did not issue a CLOSE or OPEN macro for this transition.

Application programs cannot close a DCB under a task that differs from the task that opened the DCB but the application program can read the data set in a task that differs from the task that opened the DCB. When making the transition to the next data set, the system can close and re-open the DCB. In this case, the system will call the close and open exits under the reading task. The application program issued an access method macro (CHECK, GET or FEOV) that caused the transition to the next data set or the application program issued an EOV macro for an EXCP DCB.

The DEB always contains the address of the TCB that opened the data set.

When opening a partitioned concatenation, the exit will receive a list of information about the data sets. The exit routines will not be called separately for each data set.

2.1.1.9 Environment During STOW

The data set's type (PDS or PDSE) has been compared to the SERVICEMASK specification on the CSVDYNEX macro. This tells whether the exit routine is interested in this type of data set.

The parameter list has been checked and the DEB has been verified and locked but little action has been taken except to verify that the data set is partitioned.

Since SYSZTIOT is not held, do not examine other DSABs or TIOT or XTIOT entries unless you issue ENQ for SYSZTIOT to prevent deletion of those control blocks.

2.1.1.10 Environment During Close

The data set has been verified to be open and the DEB is locked. Its type has been compared to the SERVICEMASK specification on the CSVDYNEX macro. This tells whether the exit routine is interested in this type of data set.

2.1.1.10.1 Effects of Concatenation

For information about the close exit being called under a task that differs from the task that opened the DCB, see the concatenation notes above for the open exit.

During this transition for a sequential concatenation, the system might have performed the following for the DCB before calling IFG_CLOSE_START:

- If the data set is subject to QSAM HiperBatch, the caching backend processing has been done.
- If the user coded FREE=CLOSE on the DD statement, the system has called dynamic unallocation even though the DEB remains.

2.1.1.11 Parameter list for OPEN/CLOSE/EOV and access method SVC exit routines

Figure 1. Parameter list for OPEN/CLOSE/EOV and access method SVC exit routines

Offset	Length, Bit Pattern or Value	Name	Description
0 (X'0')		IFGSVCParmL	DSECT name
	4	IFGSVCId	Identifier
		IFGSVCID_Const	Constant 'SVCP' for IFGSVCID
4 (X'4')	1	IFGSVCVER	Version (1)
5 (X'5')	1	IFGSVCPL_CallType IFGSVCPL_EarlyOpen IFGSVCPL_EarlyClose IFGSVCPL_EarlySTOW	Type of exit being called 1 Early open 2 Early close 3 Early Stow
6 (X'6')	2	IFGSVCLen	Length of parameter list
8 (X'8')	1	IFGSVCPLDataSetType	Data set type
	.... .1..	IFGSVCPL_TYPE_DASD	(Open and close exits) DASD opened with DCB
	.... ..1.	IFGSVCPL_TYPE_PDSE	(STOW Exit) PDSE data set
	1	IFGSVCPL_TYPE_PDS	(STOWExit) PDS data set

Offset	Length, Bit Pattern or Value	Name	Description
9 (X'9')	1	IFGSVCPLCallerKey	Protection key of the caller of OPEN, STOW or CLOSE in the high order four bits. It might differ from the task protection key that is in TCBPKF but it should match the key of the caller of OPEN. Do not use key 0 to examine the DCB unless that is the value in IFGSVCPLCallerKey. This is to protect system integrity because an attacker can examine the registers at any time with a timer exit to learn what the exit picked up. The user's key is in this parameter list. The key might differ from the task key, which is in TCBPKF so the exit should not get storage in a user subpool such as 0 to 127 except that a request for subpool 0 while running in key 0 will result in key 0 storage and therefore does not violate system integrity rules.
10 (X'A')	1	IFGSVCPL_OPEN	OPEN or CLOSE macro options
	 xxxx	IFGSVCPL_OPENBITS	Type of I/O accessing being done. The low order four bits correspond to four bits in the OPEN parameter list except that EXTEND or OUTINX have been changed here to OUTPUT or OUTIN respectively. In those two cases OPEN also changes the first two bits of JFCBIND2 to 10 (JFCMOD) to signify DISP=MOD (unless those two bits already were set that way). Make sure to test all four bits here. Ignore the high order four bits.
	 0000	IFGSVCPL_INPUT	INPUT
	 1111	IFGSVCPL_OUTPUT	OUTPUT or EXTEND
	 0011	IFGSVCPL_INOUT	INOUT
	 0111	IFGSVCPL_OUTIN	OUTIN or OUTINX
	 0001	IFGSVCPL_RDBACK	RDBACK (read backwards) Valid on tape only
	 0100	IFGSVCPL_UPDATE	UPDAT
	.xxx		(Open and close exits) These bits represent OPEN and CLOSE macro options.
	.000		DISP
	.001	IFGSVCPL_REREAD	REREAD
	.010	IFGSVCPL_FREE	FREE (close only)
	.011	IFGSVCPL_LEAVE	LEAVE
	.100	IFGSVCPL_REWIND	REWIND (meaningful only on close for tape)
	0...		Always zero

Offset	Length, Bit Pattern or Value	Name	Description
11 (X'B')	1		Reserved
12 (X'C')	4	IFGSVCPL_DCB_ADDR	DCB address. Accessible in the protection key in IFGSVCPLCallerKey to protect system integrity. Do not use a different key even to test a bit. Do not use key 0 unless that is the value in IFGSVCPLCallerKey. Do not modify the DCB. It will have no effect.
16 (X'10')	2	IFGSVCPL_DCB_ORIGIN	DCB origin (undefined bytes at start). This is the number of undefined bytes at the start of the DCB. Do not test DCB bytes that precede the origin. Normally this is zero. It is non-zero if the programmer coded a value for the DEVD (device dependence) keyword that is not DA, the default. A non-DA value means that the DCB was not built for certain classes of device. For BSAM, BPAM and QSAM, this always is zero. For BDAM it is 16 unless the DCBH0 and DCBH1 bits are on. For EXCP, see *z/OS DFSMSdfp Advanced Services*. A DCB origin of zero means that the first word in the DCB is valid. That word might point to the DCBE.
18 (X'12')	2	IFGSVCPL_DCB_LENGTH	DCB length (from offset 0, DCBDCBE). For QSAM, this is 96. For BSAM and BPAM it is 88. For EXCP, it depends on whether the DCB has the OPTCD fields and the fields that identify appendages.
20 (X'14')	4	IFGSVCPL_DEB_ADDR	(STOW and close) DEB Address. The DEB is locked with the DEBCHK macro to ensure that it remains valid in the exit.
24 (X'18')	4	IFGSVCPL_UCB_ADDR	Captured or actual (31-bit) UCB address. A captured UCB has a 24-bit address of a 31-bit actual UCB. If the UCBVRDEV bit is on, that data set is a VIO data set so some UCB fields will be missing.

Offset	Length, Bit Pattern or Value	Name	Description
28 (X'1C')	4	IFGSVCPL_DSAB_ADDR	DSAB address. The DSABTIOX bit tells whether the DSAB and this parameter list point to an XTIOT entry and not to a TIOT entry. The DSABCATM bit tells whether this data set is part of a concatenation. It might be a sequential or a partitioned concatenation. If the DD name in the TIOT entry or XTIOT is not blank, then this is the first data set in the concatenation. If the DSABLCAT bit is on, this is the last data set in the concatenation.
32 (X'20')	4	IFGSVCPL_TIOT_ADDR	TIOT or XTIOT entry address as mapped by IEFTIOT1. If it is a TIOT entry, it contains UCB addresses. They might be captured (24-bit versions of actual 31-bit addresses). If it is an XTIOT entry, it does not contain UCB addresses. To find UCB addresses after the first one, the exit can use the IEFDDSRV macro. The last UCB addresses in the TIOT or XTIOT list for a multivolume data set might be for the dummy SMS UCB that represents slots that SMS will fill in if this program extends the data set.
36 (X'24')	4	IFGSVCPL_JFCB_ADDR IFGSVCPL_DSN_ADDR	(Open and close) Address of a copy of the JFCB. *See notes after this table.* (STOW) Data set name address
40 (X'28')	4	IFGSVCPL_DSCB_ADDR	(Open and close) DSCB Address - It points to an area mapped by the DPL DSECT to contain DSCBs. *See notes after this table.*
44 (X'2C')	4	IFGSVCPL_WORKAREA_ADDR	Address of 256 bytes of work area for use by the exit routine. Contents are unpredictable.
48 (X'30')	8		Reserved.
56 (X'38')	8	IFGSVCPL_JOBNAME	Job name
64 (X'40')	8	IFGSVCPL_STEPNAME	Job step name
72 (X'48')	8	IFGSVCPL_PGMNM	Job step program name as obtained from the SCT, step control table.
80 (X'50')	8	IFGSVCPL_JOBID	Unique job identifier as obtained by the IAZXJSAB macro. This is for consistency with SMFJOBID in the SMF type 14/15 record.
88 (X'58')	1	IFGSVCPL_STOWREQ	STOW request
	1	IFGSVCPL_STOWRA	STOW A (Add)
	2	IFGSVCPL_STOWRC	STOW C (Change)

Offset	Length, Bit Pattern or Value	Name	Description
	3	IFGSVCPL_STOWRD	STOW D (Delete)
	4	IFGSVCPL_STOWRR	STOW R (Replace)
	5	IFGSVCPL_STOWRI	STOW I (Initialize)
	6	IFGSVCPL_STOWRDISC	STOW DISC (Disconnect)
	7	IFGSVCPL_STOWRIFF	STOW IFF (IF and only if)
	8	IFGSVCPL_STOWRDG	STOW DG (Delete Generation)
	9	IFGSVCPL_STOWRRG	STOW RG (Replace Generation)
	10	IFGSVCPL_STOWRRECoverG	STOW RECOVERG (Recover Generation)
89 (X'59')	15		Reserved
104 (X'68')	8	IFGSVCPL_STOWMName	STOW Member Name
112 (X'70')	8	IFGSVCPL_STOWNMName	STOW New Member Name
	120	IFGSVCPLLen	Length of parameter list

Additional comments about certain fields in the parameter list:

- The JFCB that IFGSVCPL_JFCB_ADDR points to begins with the data set name and includes up to five volume serial numbers for the data set. The JFCBNVOL byte contains the volume serial count, which might exceed the number of volumes identified in the JFCBVOLS field. This allows for SMS to extend this data set while it is allocated and being written by this program. It does not include volumes that might be added by a concurrently running program job. If the data set is open for output and is SMS-managed and the user program writes a large amount of data, the system might be able to extend JFCBNVOL and JFCBVOLS to add volumes.

  If the JFCBVLSQ field contains a value greater than 1, it is the volume sequence number that identifies the volume (such a 2, 3 or 4) that the user wants the access method to begin with. It might be reading or writing.

  If the user read the JFCB and turned on the JFCNWRIT bit and issued the OPEN macro with TYPE=J, it means that the user does not want the system to update the system's copy of the JFCB. If the user also changed the data set name in the JFCB, this can be a system integrity problem because the system did not serialize on the new data set name. (It also must already exist.) Therefore, if the application program is not authorized, open calls dynamic allocation for the new data set name and close later unallocates it. The point is that the data set being opened might be different from the data set identified in the original JFCB.

  In these cases where the application program updated the data set name in the JFCB, it will be the real name being opened but in the case of an authorized program, it might not be the real name for the close call. The warning about this is documented in the RDJFCB macro documentation in *z/OS DFSMSdfp Advanced Services*.

  If the data set being opened is the VTOC (volume table of contents) for the volume, the data set name will be 44 bytes of X'04'. This is not a real data set name.

- In the STOW exit, do not examine other DSABs or TIOT or XTIOT entries unless you issue ENQ for SYSZTIOT to prevent deletion of those control blocks.
- IFGSVCPL_DSCB. In the open exit, this has the address of a chain of one or more DSCBs as mapped by the DPL DSECT defined by the IFGSVCPL macro. If it is a format 4 DSCB, then it is for a VTOC, volume table of contents, and it will be the only DSCB. Otherwise, it will be a format 1 or 8 DSCB, as mapped by the IECSDSL1 macro.

Figure 2. DPL, DSCB parameter list

Offset	Length or Bit Pattern	Name	Description
DPL			DSECT name
0 (0)	44	DPLDSName	Data set name – do not test this field in the first DSCB; the field is not allocated; the next field still is at offset 44.
44 (X'2C')	96	DPLData	DSCB beginning with DS1FMTID (see IECSDSL1)
140 (X'8C')	4	DPLNext	Address of next DPL or zero
144 (X'90')	4	DPLUCB	Address of UCB

In the close exit, this has the address of only one DSCB unless the data set is striped. If it is striped, the chain contains only format 1 or 8 DSCBs. The DSCB might be a format 4 DSCB, for the VTOC.

The DSCB address might be zero.

In the first entry in the chain mapped by the DPL DSECT, the content of the data set name is unpredictable and possibly unaddressable. Only the last 96 bytes of the first DSCB are present. The name of the first or only data set being opened or closed is in the JFCB. In the subsequent DPL entries, the full 140 bytes of the DSCB are present. In each DSCB, you can test the DS1FMTID byte to learn the type of DSCB.

For the OPEN call and not for CLOSE, there might be more DSCBs:

- If the first DSCB is a format 8, then it will be followed by its format 9. An exception is that if the first DSCB is for extended format, the chain will have only format 1 and 8 DSCBs. It will not have format 9 or format 3 DSCBs.
- If the data set is not a PDSE and not extended format and it has more than three extents or more than two extents with a user label extent, then there will be a format 3 DSCB.
- If the access method in the DCB is BDAM and the data set has multiple volumes, then all of the DSCBs for the data set on all of its volumes will be on the chain.
- If the DSORG field in the DCB has PO (for partitioned organization), then the data set might be a concatenation. In that case all DSCBs for all data sets in the concatenation will be read with these exceptions:
  - For each PDSE, any format 3 DSCBs will not be read.
  - For any z/OS UNIX directory, the system will build a pseudo DSCB. If all of the DDs in the partitioned concatenation are for z/OS UNIX directories, then the exit will not be called.

If the DS1DSGPS bit in the DS1DSORG field is on, it is a sequential data set. If the DS1LARGE bit also is on, it is a large format sequential data set. If the DS1STRP bit is on (in addition to DS1DSGPS), it is an extended format sequential data set. If neither of those two bits (DS1LARGE and DS1STRP) is on, it is a basic format sequential data set.

If the DS1DSGPO bit in DS1DSORG is on, it is a partitioned data set (PDS or PDSE). If the DS1PDSE bit is off, it is a PDS. If the DS1PDSE bit is on but the DS1PDSEXbit is off, it is a PDSE. If both DS1PDSE and DS1PDSEX are on, it is an HFS data set (not an HFS or zFS file). These exits will not be called in that HFS case.

If the DS1ENCRP bit is on, it is an encrypted data set. To learn the encryption information, you can call CSI, catalog search interface, to retrieve the ENCRYPTA field.

2.1.1.12 Passing Information from an OPEN Exit Routine to a Later Routine

An exit routine cannot return something that the system will pass to another exit point for that DCB because there is no correlation between types of exits. In other words, if you provide exit routines for open and close and another provider provides exit routines for open and close, the system has no way to make a connection between the two exits or exit routines.

The exit routine could use the name/token service (IEANTCR) to save information that is associated with the current task, address space or system for later use.

2.2 z/OS DFSMS Macro Instructions for Data Sets

The STOW macro has a new return code in register 15. It is decimal 32. Previously it was reserved. Now it means this for any type of STOW invocation:

Reason code 8 means that an installation exit, IGG_STOW_EARLY, gave the return code that means to fail the STOW invocation. Message IEC997I identifies the exit routine.

2.3 z/OS MVS Installation Exits

In Chapter 50, “DFSMS Exits”, is a table of exits provided in DFSMS. These will be added:

Exit	Description
IFG_CLOSE_START	Early part of data set close
IFG_OPEN_START	Early part of data set open
IGG_STOW_START	Early part of STOW SVC

2.4 z/OS DFSMSdfp Diagnosis

The following will be added in Chapter 15. “OPEN/CLOSE/EOV (common) diagnostic aids” in section “Using error records for debugging”:

Abend error recording due to a failure in an open or close SVC dynamic exit

Early in the processing of an open or close, it calls the IFG_OPEN_START or IFG_CLOSE_START dynamic exit. If an abend occurs in it, O/C/E issues CSVDYNEX REQUEST=RECOVER. If this is successful, it retries the abend with no message. If the RECOVER operation is not successful, it implies a system logic error. In that case, the ESTAE recovery routine issues SETRP to cause a system dump and to record the failure in LOGREC. The SETRP macro has enough VRA keywords so that DAE should be able to elimination

duplicate system dumps if an installation exit causes too many dumps. DAE stands for “dump analysis and elimination”.

2.5 z/OS MVS System Messages Volume 7 (IEB-IEE)

2.5.1 IEC141I 013-*rc,mod,jjj,sss, ddname[-#] [,dev,volser, dsname(member)*] [,*phrase*]

The existing message IEC141I is for system abend 013. It will have a new return code C1 and the *phrase* is a new optional field that can be appended.

Here are changes to the existing lengthy description:

Explanation

phrase

Optional phrase with further information. See the explanation below.

Return Code Explanation

C1

An IFG_OPEN_START installation exit routine gave a failing return code. This means to fail the open of this data set. At the end of the message will be a phrase like this:

Exit *xxxxxxxx* return code *nnn*

This identifies the exit module and its return code in decimal. It is a dynamic exit, meaning that multiple installation exit routines might be registered to be called at this exit point. If multiple exit routines gave a failing return code for the same open, this message reflects only the last one to give a failing return code.

System Programmer Response

For return code C1, investigate the identified exit routine to learn why it gave the failing return code. It should write a message.

Programmer Response

If the return code is C1, contact your system programmer. That person should recognize the exit name.

Module

Add IFG0194E

2.5.2 IEC212I 414-*rc,mod,jjj,sss, ddname*[-#],*dev,ser,dsname*,[*phrase*]

The existing message IEC212I is for system abend 414. It will have a new return code 18 and the *phrase* is a new optional field that can be appended.

Here are changes to the existing description:

Explanation

phrase

Optional phrase with further information. See the explanation below.

Return Code Explanation

18

An IFG_CLOSE_START installation exit routine gave a failing return code. This means to fail the close of this data set. At the end of the message will be a phrase like this:

Exit *xxxxxxxx* return code *nnn*

This identifies the exit module and its return code in decimal. It is a dynamic exit, meaning that multiple installation exit routines might be registered to be called at this exit point. If multiple exit routines gave a failing return code for the same open, this message reflects only the last one to give a failing return code.

System Programmer Response

For return code 18, investigate the identified exit routine to learn why it gave the failing return code. It should write a message.

Programmer Response

If the return code is 18, contact your system programmer. That person should recognize the exit name.

Module

Add IFG0200V.

2.5.3 IEC997I INSTALLATION EXIT *exit-point exit-name* GOT ABEND {Ssss-xxxxxxxx | Uuuuu-xxxxxxxx} JOB *jjjjjjj* STEP *ssssssss*

Explanation

An installation exit in the OPEN/CLOSE/EOV or basic access methods component failed with the specified abend code.

In the message text:

Exit-point

This identifies the dynamic exit point. It is IFG_OPEN_START, IFG_CLOSE_START or IGG_STOW_START.

Exit-name

Name of the module that is defined for the system to call at this exit point.

Sss-xxxxxxxx

The system abend code and return code in hexadecimal. If the abend code was issued by the OPEN/CLOSE/EOV or access method (BAM) component, then the return code is only two hex digits.

Uuuu-xxxxxxxx

The user abend code in decimal and return code in hexadecimal.

Jjjjjjj

Job name.

ssssssss

Job step name.

System action

In the cases of open and close, the system creates a system dump. The abend has no effect on the application program because this appears to be a problem with the exit routine.

Programmer response

Report this message to a system programmer. This does not appear to be an application program problem.

Source

DFSMSdfp

Module

IFG019RR, IGCT0021

Routing code

10, 11

Descriptor code

4

2.6 z/OS DFSMSdfp Advanced Services

A new bit is added to a table in the Data Facilities Area section in Appendix A.

The bit is added to the byte DFAFEAT11 at offset 83 (X'53'):

```
...1 ....  DFAOCEExits   OPEN/CLOSE/EOV and STOW early exits are enabled
```

End of Document