System Automation for z/OS
Version 3 Release 4

Product Automation Programmer's Reference and Operator's Guide

IBM

SC34-2643-00

Note!

Before using this information and the product it supports, read the information in Appendix B, "Notices," on page 113.

This edition applies to IBM Tivoli System Automation for z/OS (Program Number 5698-SA3) Version 3 Release 4, an IBM licensed program, and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

This edition replaces SC34-2569-02.

| IBM welcomes your comments. You may forward your comments electronically, or address your comments to:
| IBM Deutschland Research & Development GmbH
| Department 3248
| Schoenaicher Strasse 220
| 71032 Boeblingen
| Germany

If you prefer to send comments electronically, use one of the following methods:
FAX (Germany): 07031 16-3456
FAX (Other Countries): +49 7031 16-3456
Internet: s390id@de.ibm.com

Internet

Visit our home page at http://www.ibm.com/systems/z/os/zos/features/system_automation/

Contents

Figures

Tables

Accessibility

Publications for this product are offered in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when using PDF files, you may view the information through the z/OS® Internet Library website or the z/OS Information Center. If you continue to experience problems, send an email to mhvrcfs@us.ibm.com or write to:

IBM® Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
U.S.A.

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer or Library Server versions of z/OS books in the Internet library at:

`http://www.ibm.com/systems/z/os/zos/bkserv/`

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.
- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

  Notes:

  1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
  2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
  3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.
- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:

1. Send an email to s390id@de.ibm.com
2. Visit the SA z/OS home page at http://www.ibm.com/systems/z/os/zos/features/system_automation/
3. Visit the Contact z/OS web page at http://www.ibm.com/systems/z/os/zos/webqs.html
4. Mail the comments to the following address:
   IBM Deutschland Research & Development GmbH
   Department 3248
   Schoenaicher Str. 220
   D-71032 Boeblingen
   Federal Republic of Germany
5. Fax the comments to us as follows:
   From Germany: 07031-16-3456
   From all other countries: +(49)-7031-16-3456

Include the following information:

- Your name and address
- Your email address
- Your telephone or fax number
- The publication title and order number:
  IBM Tivoli System Automation for z/OS V3R4.0 Product Automation Programmer's Reference and Operator's Guide
  SC34-2643-00
- The topic and page number related to your comment
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you submit.

If you have a technical problem

Do not use the feedback methods listed above. Instead, do one of the following:

- Contact your IBM service representative
- Call IBM technical support
- Visit the IBM zSeries support web page at www.ibm.com/systems/z/support/.

About This Book

This book describes how to customize and operate product automation components (CICS®, DB2®, and IMS™ automation) with IBM Tivoli® System Automation for z/OS (SA z/OS) to provide a simple and consistent way to monitor and control all of the CICS, DB2, and IMS regions, both local and remote, within your organization. SA z/OS automates, simplifies, and standardizes console operations and the management of component, application, and production-related tasks.

Who Should Use This Book

This book is intended for two kinds of users or user groups:

- System programmers, system designers, and application designers who will automate CICS, DB2, and IMS using SA z/OS.

  Installing and customizing CICS, DB2, and IMS automation requires a programmer's understanding of NetView®, CICS, DB2, IMS, and SA z/OS, because most of the definitions take place in these programs. Also, you will modify JCL, command lists, and programs for some of the automation functions
- Operators and administrators who manage and monitor CICS, DB2, and IMS subsystems.

  For operators, a working knowledge of CICS, DB2, and IMS will be assumed.

What's in This Book

This book contains the following:

- Part 1, "CICS Automation," on page 1. This part describes the principal concepts of SA z/OS with regard to CICS Automation and describes the steps that are necessary to customize and set up CICS Automation.
- Part 2, "DB2 Automation," on page 33. This part gives an overview of the functions provided by DB2 Automation and how to set them up.
- Part 3, "IMS Automation," on page 41. This part describes the principal concepts of SA z/OS with regard to IMS Automation and describes the steps that are necessary to customize and set up IMS Automation. It also describes the tasks of the operator who manages IMS subsystems through IMS Automation.

Related Publications

The System Automation for z/OS Library

Table 1 shows the information units in the System Automation for z/OS library:

Table 1. System Automation for z/OS Library

Title	Order Number
IBM Tivoli System Automation for z/OS Planning and Installation	SC34-2645
IBM Tivoli System Automation for z/OS Customizing and Programming	SC34-2644
IBM Tivoli System Automation for z/OS Defining Automation Policy	SC34-2646
IBM Tivoli System Automation for z/OS User's Guide	SC34-2647
IBM Tivoli System Automation for z/OS Messages and Codes	SC34-2648

Table 1. System Automation for z/OS Library (continued)

Title	**Order Number**
IBM Tivoli System Automation for z/OS Operator's Commands	SC34-2649
IBM Tivoli System Automation for z/OS Programmer's Reference	SC34-2650
IBM Tivoli System Automation for z/OS Product Automation Programmer's Reference and Operator's Guide	SC34-2643
IBM Tivoli System Automation for z/OS TWS Automation Programmer's Reference and Operator's Guide	SC34-2651
IBM Tivoli System Automation for z/OS End-to-End Automation Adapter	SC34-2652
IBM Tivoli System Automation for z/OS Monitoring Agent Configuration and User's Guide	SC34-2653

The System Automation for z/OS books are also available on CD-ROM as part of the following collection kit:

IBM Online Library z/OS Software Products Collection (SK3T-4270)

SA z/OS Home Page

For the latest news on SA z/OS, visit the SA z/OS home page at http://www.ibm.com/systems/z/os/zos/features/system_automation

Related Product Information

You can find books in related product libraries that may be useful for support of the SA z/OS base program by visiting the z/OS Internet Library at http://www.ibm.com/systems/z/os/zos/bkserv

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from these locations to find IBM message explanations for z/OS elements and features, z/VM®, z/VSE®, and Clusters for AIX® and Linux:

- The Internet. You can access IBM message explanations directly from the LookAt Website at www.ibm.com/systems/z/os/zos/bkserv/lookat/index.html
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations using LookAt from a TSO/E command line (for example: TSO/E prompt, ISPF, or z/OS UNIX System Services).
- Your Microsoft Windows workstation. You can install LookAt directly from the *z/OS Collection* (SK3T-4269) or the *z/OS and Software Products DVD Collection* (SK3T-4271) and use it from the resulting Windows graphical user interface (GUI). The command prompt (also known as the DOS > command line) version can still be used from the directory in which you install the Windows version of LookAt.
- Your wireless handheld device. You can use the LookAt Mobile Edition from www.ibm.com/systems/z/os/zos/bkserv/lookat/lookatm.html with a handheld

device that has wireless access and an Internet browser (for example: Internet Explorer for Pocket PCs, Blazer or Eudora for Palm OS, or Opera for Linux handheld devices).

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from:

- A CD-ROM in the *z/OS Collection* (SK3T-4269).
- The *z/OS and Software Products DVD Collection* (SK3T-4271).
- The LookAt Website (click **Download** and then select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Summary of Changes for SC34-2643-00

This document contains information that was previously presented in System Automation for z/OS V3.R3.0 Product Automation, SC34-2569-02.

New Information

The following new information has been added:

Part 1. CICS Automation

The following routine is added in Chapter 4, "CICS Automation Routines," on page 25 and moved from *IBM Tivoli System Automation for z/OS Customizing and Programming*:

- "EVEERTRN" on page 26

Part 3. IMS Automation

The following routines are added and moved from *IBM Tivoli System Automation for z/OS Customizing and Programming*:

- EVIECT0X
- EVISTRCT
- EVIEET00
- EVIEI006
- EVISTRMN

They form a new chapter Chapter 14, "IMS Automation Routines," on page 83.

Changed Information

The following information has been updated:

Part 1. CICS Automation

- The messages responded to by the routine EVEES100 are now added in "EVEES100: Recovery Function for Short-On-Storage Conditions" on page 26.

Part 3. IMS Automation

- You can issue type 1 and type 2 IMS commands using IMSINFO and INGIMS, see "IMSINFO: Display Information" on page 64 and "INGIMS: Issue List of Defined Transactions and View the Output" on page 67.
- A data set is used for TCO members and issuing commands for time-driven procedures. See "TCO: Allocate data set with Commands for Time-Driven Procedures" on page 64.

- Use of the INGIMS command parameters (DEP and TCO) is described in "Working with TCO Functions" on page 76 and "Displaying IMS Dependent Region Information" on page 77 for menu options available at the IMS Automation Menu.

Deleted Information

The following information has been deleted:

Part 3. IMS Automation

- The IMS Automation menu is amended and no longer display option 7 - SDF any more.

You may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Part 1. CICS Automation

This part describes:

- The principal concepts of SA z/OS with regard to CICS Automation.
- The steps that are necessary to customize and set up CICS Automation. The description focuses on those steps that are specific to CICS Automation and have to be done in addition to the definitions of the basic functions of SA z/OS that are common to all subsystems.
- CICS-specific message IDs in the MESSAGES/USER DATA policy item and common routines that request information or perform tasks associated with CICS Automation.

It contains the following chapters:

- Chapter 1, “CICS Automation Concepts,” on page 3
- Chapter 2, “How to Set Up the Functions of CICS Automation,” on page 11
- Chapter 3, “MESSAGES/USER DATA Entries for CICS Automation,” on page 19
- Chapter 4, “CICS Automation Routines,” on page 25

Chapter 1. CICS Automation Concepts

This chapter describes the principal concepts of CICS Automation in SA z/OS and gives some hints relating to its customization.

*CICS Best Practices Policy

SA z/OS supplies a sample policy, *CICS, that provides best practice definitions for running CICS Automation.

Diagrams of the sample policies are provided as PDF files that are located in the USS installation path. The default for this path is: /usr/lpp/ing/doc/policies/.

You customize CICS Automation for your specific installation by modifying the policy database in the customization dialog. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* and the online documentation for the *CICS best practices policy for details about how to import the *CICS best practices policy and modify the definitions that this import adds to your policy database.

Customization Hints

Bear in mind the following hints when customizing the *CICS best practices policy:

- The *CICS best practices policy supplies the automation operator definition that is required for CICS Automation. It is located in the CICS_AUTO_OPS policy object for the AOP entry type.

Automated Function	**Operator ID**	**Message Classes**
CICSMSTR	AUTCICS	

  Make sure that this operator ID is defined as a NetView operator.
- All CICS regions must be defined to SA z/OS as APPLICATION objects. The *CICS best practices policy provides you with a number of sample classes and instances of entry type APL that model CICS regions. Copy and modify them to your needs:
  - The names of the APL classes in *CICS have the prefix C_CICS
  - The names of the APL instances in *CICS have the prefix CICS or CPSM
- For the following CICS components, the **Application Type** field must be set to CICS:
  - CICS regions
  - CICSPlex® SM Address Space (CMAS)
  - CICS Transaction Server
  - CICSPlex SM Web User Interface Server
- For applications of type CICS the Subtype field must be set to the appropriate value.
- CICS regions are shut down with the CEMT PERFORM SHUTDOWN command. However, for CMAS regions, the COSD command is used for shutdown.

Operation Hints

In the *CICS best practices policy, relationships have been set up, for example, from monitor resources to the CICS application group to make sure that the monitor resources can rely on all required CICS functions. Be aware that the shutdown of a CICS component may not be performed due to these defined resource dependencies. You should therefore address the shutdown request to the CICS application group instead of a CICS component to avoid hanging shutdown requests.

Using the CICS Automation Message Exit

To enable the CICS Message and Transient Data Queue exits to process messages, the messages must be defined in the policy database. The MESSAGES/USER DATA policy is used to define the messages. Messages IDs that are added to the MESSAGES/USER DATA policy for CICS Message exit processing must not contain special characters. Each message ID must be a single alphanumeric word.

If there are no messages defined to use the exits, the exits are disabled.

Message policy data is processed on a subsystem basis, however, you can put the policy information in a CLASS and link the class to the subsystems to save on data entry.

Each subsystem has its own policy and does not share policy information, except for CLASS information, with other subsystems. This means you can update the policy information for a subsystem or set of subsystems and not affect other subsystems.

Defining CICS Messages

CICS messages that are not already WTO'd can be WTO'd by specifying the message in the customization dialog. In addition you can control the transient data queues and routing codes for any messages beginning with DFH.

There are several special keywords that, when added to the message with the USR command, cause the message information to be loaded into the appropriate exit for processing. The keywords and their descriptions are described in Table 2.

Table 2. Keywords in CICS Message Definitions

Keyword	Description
OFFSET	This keyword is required to load the message into the exit. It specifies the word number in the message that represents the message ID. Its format is a single integer number.

Table 2. Keywords in CICS Message Definitions (continued)

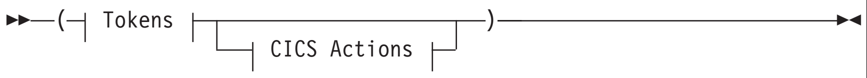
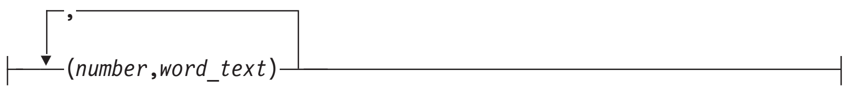
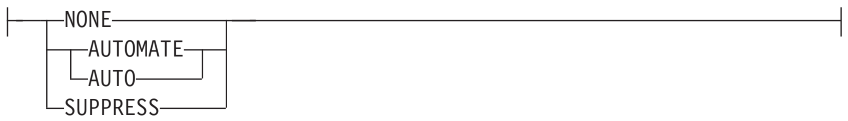
Keyword	Description
INSERT	This keyword specifies the matching Insert values for CICS messages. It is optional and allows you to make decisions to control the message based on its content. Its format is:  **Tokens:**  **CICS Actions:**  Where *number* is the insert number to be checked and *word_text* is the text that is to be checked. Inserts are described in *CICS Messages and Codes* for each of the messages that have them. The *word_text* specification is checked for the length specified. This allows for trailing data to be ignored. The format of the number is a single integer. The format for *word_text* is a single alphanumeric word; no blanks or special characters are allowed. The case of the value is ignored. The actions that are available are: **NONE** No actions are to be taken. **AUTOMATE|AUTO** WTO the message for SA z/OS to trap. **SUPPRESS** The message is suppressed from z/OS consoles and also all TD queues. CICS reserves the right not to suppress messages that it determines are critical. The default action is no action if the MSGDISP keyword is specified. This allows for the removal of an action based on message content. If no MSGDISP keyword is specified the default action is AUTOMATE.
TDQUEUE	This keyword specifies the matching Transient Data Queue. Do not specify DFH messages that are already issued as a WTO with a TDQUEUE value. The format of the data is a single alphanumeric word up to 4 bytes in length. The specification of this keyword in a message policy causes the message information to be matched against all messages written to the Transient Data Queue that is specified. If a match is made, the message may be WTO'd depending upon any other keywords that are specified.

Table 2. Keywords in CICS Message Definitions (continued)

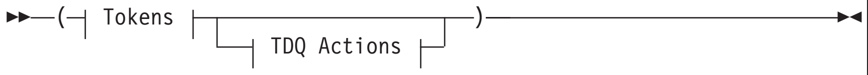
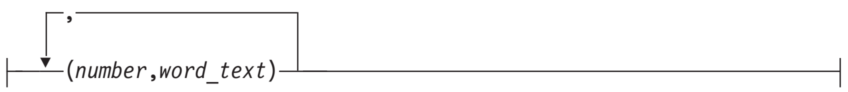
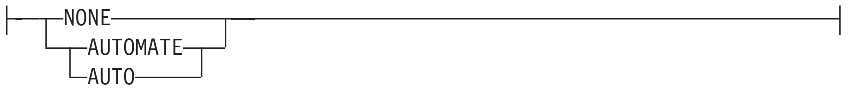
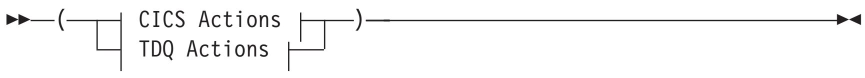
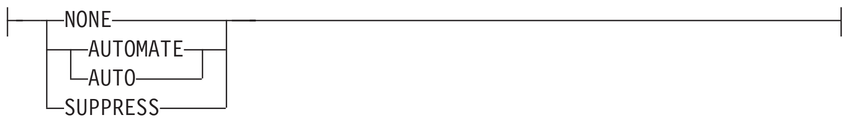
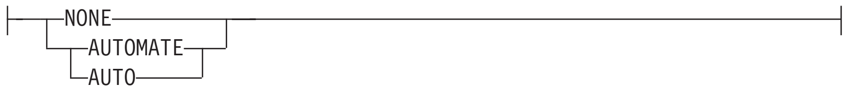
Keyword	Description
TOKEN	This keyword specifies the matching token values for user messages. It is optional, but if present, only messages that match the values specified are WTO'd. Its format is:  **Tokens:**  **TDQ Actions:**  Where *number* is the word number to be checked and *word_text* is the text to be checked. Words are counted from the beginning of the message starting from 1. A word is considered to be a contiguous set of characters that contain alphanumerics or any of the '$#§_Ü?' characters. Any other characters are treated as word delimiters. The *word_text* specification is checked for the length specified. This allows for trailing data to be ignored. The format of the number is a single integer. The format for *word_text* is a single alphanumeric word; no blanks or special characters are allowed. The case of the value is ignored. The actions that are available are: **NONE** Specifies that no action is to occur for the matching message. **AUTOMATE | AUTO** WTO the message for SA z/OS to trap. The default action is AUTOMATE.

Table 2. Keywords in CICS Message Definitions (continued)

Keyword	Description
MSGDISP	This keyword specifies the default actions for a message ID. The format of the data is dependent on the type of message being processed and is as follows:  **CICS Actions:**  **TDQ Actions:**  The actions that are available are: **NONE** The message is not acted on by the exit. Use this option to set a default of no actions when used with the INSERT or TOKEN keywords. **AUTOMATE|AUTO** WTO the message for SA z/OS to trap. **SUPPRESS** The message is suppressed from z/OS consoles and also all TD queues. CICS reserves the right not to suppress messages that it determines are critical. The default action is AUTOMATE if no INSERTs or TOKENs are present. If either INSERTs or TOKENs are present the default action is NONE.

To define a CICS message to be WTO'd, do the following:

1. Specify the message ID in the MESSAGES/USER DATA policy item.
2. Enter the USR command against the message ID. Specify OFFSET in the **Keyword** field and a data value of 1 in the **Data** field.
3. Optionally enter the USR command against the message ID and specify INSERT in the **Keyword** field. Optionally specify the actions for the INSERT contents.

   Note: You can specify several INSERT keywords.
4. Optionally enter the USR command against the message ID and specify MSGDISP in the **Keyword** field to detail the actions desired.
5. Specify either the AUT or the CMD command, or both. Use the AUT command to specify capturing the message or other automation functions. Use the CMD command to specify commands that are to be executed when the message occurs.

To define a user message to be WTO'd from a Transient Data Queue, do the following:

1. Specify the message ID in the MESSAGES/USER DATA policy item.

2. Enter the USR command against the message ID. Specify OFFSET in the **Keyword** field and the word number that the message ID occurs at in the message in the **Data** field.
3. Enter the USR command against the message ID. Specify TDQUEUE in the **Keyword** field and the name of the Transient Data Queue that the message is written on in the **Data** field.
4. Optionally, enter the USR command against the message ID and specify TOKEN in the **Keyword** field, and the words and their values that are to be matched in the **Data** field.
5. Optionally enter the USR command against the message ID and specify MSGDISP in the **Keyword** field to detail the actions desired.

Defining the Reload Command

To load the updated information into CICS address spaces whenever the automation configuration is refreshed, specify the following command for message ID ACORESTART:

```
MVS F &SUBSJOB,reload_transaction
```

The transaction SARL can be specified as *reload_transaction* to reload the CICS message exit policy.

Refreshing Policy Data

To load the information into CICS address spaces issue the INGAMS REFRESH command. As a part of the ACF load, each CICS that had changes to MESSAGES/USER DATA policy will be reloaded with any changes. If you want to disable the exits, delete the messages from the policy.

Partial Message IDs and Performance

A partial message ID is one where only a part of the message ID is specified. The partial message ID is restricted to the leftmost part of the message ID. You cannot specify a partial message ID, such as `XXX**YY`, only `XXX` would be acceptable. For messages beginning with DFH, the exit does not support partial message IDs and the full message must always be specified. For messages in the TD queues, the first three bytes of the message ID are the minimum required to specify a message to trap in the exit. You can specify a partial message ID within this limitation.

It is recommended that the TOKEN keyword not be used with partial message IDs for messages. This causes a performance degradation because the token data must be matched for the partial message and then again for any specific message.

You can specify MSGDISP policy information on the partial message IDs. This sets a default for actions for all specific message IDs that match the partial message ID. You do not have to specify more specific message IDs. Instead a partial message ID and a matching set of TOKEN keywords can be used to specify the actions for a message. Be aware that token scanning is much less efficient than message ID scanning.

You can use the partial message IDs to improve exit performance. An extreme example might be if you are trapping messages ABC000-ABC100. You might choose to create partial message IDs for ABC00-ABC10. The partial message IDs would specify MSGDISP=(NONE) to prevent them supplying default actions. This would cause the exit to first scan the set of partial message IDs ABC00-ABC10 or 11 checks. Then, assuming it found a match on ABC10, it would scan the specific messages ABC110-ABC119 for a total of 21 checks as opposed to 101 for the original set. This would require 112 MESSAGES/USER DATA policy definitions. 101 for messages ABC000-ABC100 and 11 for messages ABC00-ABC10. In normal circumstances you can ignore the usage of partial message IDs for performance reasons and use them for functional reasons.

Chapter 2. How to Set Up the Functions of CICS Automation

This chapter explains how to set up the functions of CICS Automation for your specific needs. For the setup of base functions, for example, starting and stopping subsystems, see the SA z/OS documentation.

| For how to define transactions and programs, please refer to *IBM Tivoli System*
| *Automation for z/OS Planning and Installation*.

Automating Recovery for Application Components

CICS Automation provides automation for transaction recovery and short-on-storage conditions, and also enables monitoring of CICS DB2 connections.

You can control automated recovery through the following policy items of the APPLICATION object:

MINOR RESOURCE FLAGS

With these flags, you can switch automated recovery on and off for transactions or for a certain problem area. To do this, define a minor resource and set its **Recovery** flag as required. For the definition of minor resources, see *IBM Tivoli System Automation for z/OS Defining Automation Policy*. The names of these minor resources must be as follows:

Problem area	Minor resource name
Short-on-storage conditions	RCVRSOS
Transaction recovery	TRAN[.*trans_id*]

For transaction recovery, you can also define second-level minor resources by suffixing TRAN with the transaction name. The recovery flag of the TRAN minor resource applies to all transactions of the respective application; TRAN.*trans_id* only applies to the *trans_id* transaction. The transaction-specific recovery flag overrides the general TRAN flag.

When no minor resources are defined, CICS Automation acts according to the recovery setting of the application (AUTOMATION FLAGS policy item). When no second-level minor resource is defined for a transaction, the TRAN minor resource is applied. If that does not exist either, the application setting is applied. You only need to define minor resources when the recovery setting for a lower level is to be different from the next higher level.

MESSAGES/USER DATA

For every recovery type, there are one or more keywords that are used to specify how recovery is to proceed. These keywords are:

Problem area	Keywords
Short-on-storage conditions	RCVRSOS (see page 22)
Transaction recovery	ABCODETRAN (see page 19), RCVRTRAN (see page 23)

Transaction recovery is the most complex of these recovery types. Therefore this type is used to explain recovery configuration in more detail.

How to Define Transaction Recovery

Customization of transaction recovery consists of:

1. Identifying the transactions that will have recovery automation.
2. Identifying the error threshold level when recovery should be stopped.
3. Identifying specific abend codes when you want recovery procedures to take place (there are probably several that you would want to ignore).
4. Specifying the recovery procedure, which usually consists of invoking a command, a routine, or sending notification to an operator, or all three.

The recovery itself is typically triggered from the AT by calling the EVEERTRN routine when certain messages arrive at NetView. EVEERTRN then consults the ACF in order to learn what it has to do for recovery.

The following sections illustrate the configuration process by an example.

Specifying the Transactions to Be Recovered

If recovery is enabled for the CICS1 application on the application level, and you want to enable it also for transactions PAYR, DBTS, and BLNG, but not for any other transaction, you must define four minor resource flags for CICS1 in the customization dialogs:

- TRAN
- TRAN.BLNG
- TRAN.DBTS
- TRAN.PAYR

Set the recovery automation flag to `NO` for TRAN and to `YES` for the three second level minor resources. For more information, see *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Defining Recovery Thresholds

You can specify that recovery is to be stopped when the number of abends within a certain time interval reaches a certain threshold. To do this, define thresholds in the MINOR RESOURCES policy item of the APPLICATION policy object. The thresholds must have the name TRAN or TRAN.*tranid*, where the values of the TRAN thresholds will be used for all transactions *tranid* for which no TRAN.*tranid* thresholds exist. The **Critical** value of the thresholds will be used.

If you want to stop recovery specifically for PAYR if two or more abends occur within one hour, you must enter the values on the Thresholds Definitions panel as follows:

	Critical		**Frequent**		**Infrequent**	
Resource	Number	Interval (hh:mm:ss)	Number	Interval (hh:mm:ss)	Number	Interval (hh:mm:ss)
`CICS1.TRAN.PAYR`	`2`	`01:00:00`	`2`	`05:00:00`	`2`	`24:00:00`

For more details, see *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Selecting the Abend Codes

The abend codes that recovery is to take place for are specified through the ABCODETRAN keyword in the MESSAGES/USER DATA policy item for CICS1. If you want to initiate recovery for transaction PAYR only when the abend code is

AEI0 or AKC3, you must create the ABCODETRAN.PAYR entry in the Message Processing panel and associate codes with this entry as follows:

Code 1	Code 2	Code 3	Value Returned
*	AEI0	*	INCLUDE
*	AKC3	*	INCLUDE
*	*	*	EXCLUDE

For more details, see "ABCODETRAN: Transaction Abend Recovery" on page 19.

Specifying Recovery Actions

Use the RCVRTRAN keyword in the MESSAGES/USER DATA policy item to specify the actions to be taken when transaction abends occur.

For more details, see "RCVRTRAN: Transaction Recovery" on page 23.

Monitoring of CICS DB2 Connections

CICS Automation allows the monitoring of CICS DB2 connections. The CICS command CEMT INQUIRE DB2CONN is issued regularly after each monitor interval to query the status of the CICS DB2 connection.

The CICS DB2 connection monitoring function is optional, that is, definitions are necessary to enable it.

Implementation of the CICS DB2 connection monitoring function includes the following:

- A monitor resource for the connection to be monitored between a CICS and a DB2 subsystem
- Status messages
- Relationships

Note: You can also establish connection monitoring using CPSM.

CICS DB2 Connection Monitor Resource

The CICS DB2 connection monitor resource has to be defined in the MTR policy object with the following characteristics:

Field Name	Field Value
Monitored Object	DB2, MVS™ subsystem ID of the DB2 subsystem or the DB2 group name
Monitored Jobname	*jobname* of the CICS subsystem
Monitor command	INGRMCDB
Monitoring Interval	*hh:mm*

The specified monitor command INGRMCDB is executed after each monitoring interval. It issues the CEMT INQUIRE DB2CONN command and analyzes the status of the connection between the CICS and a DB2 subsystem in the response to this command.

If a DB2 subsystem ID or a DB2 group name is specified in the Monitored Object field, the monitor command only analyzes the status of a connection to this specified target. If DB2 is entered as the monitored object, any connection to an arbitrary DB2 subsystem is accepted.

Depending on the monitor results, INGRMCDB ends with the following return codes, which are mapped to the related health status for the monitor resource:

Return Code	Health Status	Description
1	BROKEN	No DB2 connection is defined in CICS
2	FAILED	CEMT INQUIRE DB2CONN command failed
3	NORMAL	CICS is connected to DB2
5	MINOR	CICS is not connected to DB2

Status Messages

Apart from active monitoring by issuing the INGRMCDB monitor command after each monitoring interval, the CICS DB2 monitoring function of SA z/OS also performs passive monitoring by updating the health status whenever it receives one of the following status messages:

```
DFHDB2023I date time applid The CICS-DB2 attachment has connected to
 DB2 subsystem db2-id{|group}db2-group

DFHDB2025I date time applid The CICS-DB2 attachment has disconnected
 from DB2 subsystem db2-id{|group}db2-group

DFHDB2037 date time applid DB2 {subsystem |group} db2-id {is not active.
|has no active members.} The CICS-DB2 attachment facility is waiting.
```

These messages indicate whether the attempt to connect the CICS region *applid* to the DB2 subsystem *db2-id* was successful. In response to these messages, the health status is set to NORMAL if it was successful or MINOR if it was not.

Relationships

To propagate the health status for the monitor resource to the health status of the application, you must define a HASMONITOR relationship between the CICS region and the monitor resource.

The monitor resource is kept active only during the UP time of the CICS application and the DB2 application group. This is achieved with an additional HASPASSIVEPARENT and a FORCEDOWN/WhenObservedDown relationship between the monitor resource and both the CICS application and the DB2 application group. This ensures that the monitor resource can rely on all required functions.

Event Monitoring

CICS Automation of SA z/OS allows the monitoring of events that are issued by CICSPlex System Manager (CICSPlex SM). When an event is received, the event severity is mapped to the health status of a related monitor resource and, in addition, defined recovery commands are issued.

The event monitoring function is optional, that is, some definitions are necessary to get it up and running. The implementation of the event monitoring function includes the following:

- The definition of a monitor resource for each event that is to be monitored
- Relationships

The event monitoring function requires the infrastructure for the event-based CICS link and health monitoring using CICSPlex SM objects. For details see *IBM Tivoli System Automation for z/OS Customizing and Programming*.

Setting Up an Event Monitor Resource

You must define a passive monitor resource in the policy database for each event that is to be monitored. You must make these definitions for the monitored object that has the following naming convention:

```
prefix.target.resource.name
```

Where:

prefix The value of the PREFIX parameter of the INGCPSM command. For details, see *IBM Tivoli System Automation for z/OS Programmer's Reference*.

target The name of the CICS system that is the target of the event (as in the CICSPlex SM EVENT view).

resource
The type of resource for which the event was generated (as in the CICSPlex SM EVENT view). For events that are generated automatically by CICSPlex SM, the resource is `!!`.

name The name of the event.

For events that are generated as a result of evaluating an analysis or status definition, the event name is the same as the definition name.

For events that are generated automatically by CICSPlex SM, the name is one of the following (as in the CICSPlex SM EVENT view):
- !!SAMOPS System unavailable
- !!SAMSDM System dump
- !!SAMTDM Transaction dump
- !!SAMSOS Short-on-storage
- !!SAMMAX Maximum tasks
- !!SAMSTL System stall

Make the following definitions for the monitored object:

1. Define codes for message ING150I with the MESSAGES/USER DATA policy item of the monitor resource to map the event severities to health states of the monitor resource.

   Specify the CICSPlex SM severity of the received events in the **Code 1** field. Use the first token of the **Value Returned** field to specify a recovery command that is to be executed. The specified token is used to select the associated command from the list of defined commands for this message. The second token is taken as the health status for the monitor resource. For example:

Code 1	Code 2	Code 3	Value Returned
`VLS`			`* NORMAL`
`LS`			`* WARNING`
`LW`			`WCMD WARNING`

2. Define recovery commands for the different CICSPlex SM event severities. Define the commands for message ING150I with the MESSAGES/USER DATA policy item of the monitor resource.

   | Use the **Ps/Select** field of the Command Processing panel of message ING150I
   | to correlate the command to the first token of the **Value Returned** in the code
   | definition.

Ps/Select	AutoFn/*	Command Text
WCMD		MYRECV

The second token of the **Value Returned** field is the status of the new health status that the monitor has determined, for example:

NORMAL
The monitor has obtained good results from the object, or objects, that it is watching

WARNING
The monitor detected a certain degree of degradation in the operation of the monitored object.

Continuing with the example, the severities VLS and LS do not execute recovery commands because there are no commands defined for the selection *. Severity LW however has the selection WCMD specified, which you probably have defined commands for.

For more details about code processing, see INGMON in *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Example: CICS Link Monitor Resource

CICS communicates with other systems or subsystems using either intersystem communication (ISC) or interregion communication (IRC). In either ISC or IRC, communication between different systems or subsystems takes place across predefined sessions. Sessions are logical links that are allocated whenever there is a need to communicate.

Monitoring such a link is a suitable example for event monitoring.

To monitor events for a connection with the name CON1 to the target CICS1TOR, specify the monitored object as:

```
CPSM.CICS1TOR.CONNECT.CON1
```

Relationships

To propagate the health status of the monitor resource to the health status of the application, you must define a HASMONITOR relationship between the CICS region and the monitor resource.

The monitor resource is kept active only during the UP time of the CICS application with both of the following relationships:

- A HASPASSIVEPARENT and a FORCEDOWN/WhenObservedDown relationship between the monitor resource and the CICS region
- An additional HASPASSIVEPARENT relationship to the connected CICS region

The monitor resource depends on the CICSPlex SM event listener with a HASPARENT and a FORCEDOWN/WhenObservedDown relationship. This ensures that the monitor resource can rely on all required functions.

Health Monitoring

CICSPlex SM employs the concept of *status programs*. You can make definitions so that a program is executed periodically and sets one of the CICSPlex SM severities. This can produce events that can be monitored with the monitoring infrastructure described in "Event Monitoring" on page 14.

The naming convention for the monitored object in this case is:

prefix.*target*.PROGRAM.*name*

Link Monitoring

CICS communicates with other systems or subsystems using either intersystem communication (ISC) or interregion communication (IRC). In either ISC or IRC, communication between different systems or subsystems takes place across predefined sessions. Sessions are logical links that are allocated whenever there is a need to communicate.

CICS Automation of SA z/OS allows the monitoring of those links or connections. This is done by registering for events that are issued by CICSPlex System Manager (CICSPlex SM) whenever there is a problem with the monitored link. When an event is received, the event severity is mapped to the health status of a relating monitor resource and in addition, defined recovery commands are issued.

The link monitoring function is optional, that is, some definitions are necessary to get it up and running.

The implementation of the link monitoring function includes the following:
- The definition of a monitor resource for each link
- Code definitions for message ING150I
- Command definitions for message ING150I
- Relationships

The link monitoring function requires the infrastructure for the event-based CICS link and health monitoring by using CICSPlex System Manager (CICSPlex SM) objects. For details see *IBM Tivoli System Automation for z/OS Customizing and Programming*.

CICS Link Monitor Resource

A link monitor resource has to be defined in the MTR policy object with the following characteristic:

Field Name	Field Value
Monitored Object	Name of the monitored CPSM object according to the naming conventions, for example, CPSM.CICS1TOR.CONNECT.CON1

Code Definitions for Message ING150I

Define codes for message ING150I with the MESSAGES/USER DATA policy item of the monitor resource to map the event severities to health states of the monitor resource. Specify the CPSM severity of the received events in the **Code 1** field. Use the first token of the **Value Returned** field to specify a recovery command to be executed. The specified token is used to select the relating command from the list of defined commands for this message. The second token is taken as the health status for the monitor resource. For example:

Code 1	Code 2	Code 3	Value Returned
VLS	*	*	* NORMAL
LS	*	*	* WARNING

Code 1	Code 2	Code 3	Value Returned
LW	*	*	* WARNING

Command Definitions for Message ING150I

Define recovery commands for the different CPSM event severities. The commands are to be defined for message ING150I using the MESSAGES/USER DATA policy item of the monitor resource. Use the selection field to correlate the command to the first token of the **Value Returned** field in the code definition.

Relationships

To propagate the health status of the monitor resource to the health status of the application, you must define a HASMONITOR relationship between the CICS region and the monitor resource.

The monitor resource is kept active only during the UP time of the CICS application with a HASPASSIVEPARENT and a FORCEDOWN/ WhenObservedDown relationship between the monitor resource and the CICS region and an additional HASPASSIVEPARENT relationship to the connected CICS region. The monitor resource depends on the CPSM event listener with a HASPARENT and a FORCEDOWN/WhenObservedDown relationship. This ensures that the monitor resource can rely on all required functions.

Chapter 3. MESSAGES/USER DATA Entries for CICS Automation

You must specify any information for CICS Automation in the SA z/OS policy database through the customization dialog. In most cases, the customization dialog itself restricts you to the format that this information must be entered in. There are, however, some CICS application-specific automation parameters that must be specified as entries in the MESSAGES/USER DATA policy item of the appropriate application. For further information about the MESSAGES/USER DATA policy item, see *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

This chapter contains detailed descriptions of these automation entries. However, a general understanding of the MESSAGES/USER DATA policy item is assumed.

CICS-Specific MESSAGES/USER DATA Keywords

The following keywords are specific for CICS Automation. They need to be entered in the MESSAGE ID field on the Message Processing panel.

Entry	Description
"ABCODETRAN: Transaction Abend Recovery."	Use this ID to define actions to be taken for transaction abend codes.
"CICSINFO: Display Information" on page 20.	Use this ID to define the effect of `INGCICS REQ=INFO`.
"LISTSHUT: Transaction Purging During Shutdown" on page 21.	Use this ID to define those transactions running under this CICS subsystem that should or should not be purged during a shutdown.
"RCVRSOS: Short-On-Storage Handling" on page 22.	Use this ID if you want CICS Automation to take action for short on storage conditions.
"RCVRTRAN: Transaction Recovery" on page 23.	Use this ID to define actions to be taken when this specific transaction has abended.

ABCODETRAN: Transaction Abend Recovery

Use this keyword to specify whether to initiate recovery actions for certain transactions, abend codes, and programs.

Use Code Processing and enter the following data:

Code 1	Code 2	Code 3	Value Returned
tran	*abend*	*pgm*	`INCLUDE` or `EXCLUDE`

Keyword and Parameter Definitions

ABCODETRAN[.*tran*]
You can add the name of a transaction as a suffix to the keyword. In this case the specifications of the CODE attributes will only apply to this transaction.

CODE
Defines which abends are recoverable, as shown in the following descriptions:

tran The transaction ID.

`abend` The abend code.

`pgm` The program that abended.

`INCLUDE|EXCLUDE`
Indicates whether or not to initiate a recovery for this transaction, abend code, and program. Use INCLUDE to initiate a recovery and EXCLUDE if you do not want a recovery initiated.

Comments and Usage Notes

1. The actions to be taken if recovery is allowed are defined with the RCVRTRAN keyword.
2. The transaction name is specified either in the keyword ABCODETRAN.*tran* or as the value of the first CODE attribute. Use ABCODETRAN.*tran* when you want all of the specifications to apply to one specific transaction. Use the CODE attribute when you want to code several transactions.
3. If a match is not found when checking the code definitions of ABCODETRAN[.*tran*], the value INCLUDE is assumed as the default value.
4. Ensure that the abend transaction message is issued as a WTO. For more details, see "Defining CICS Messages" on page 4

Example of Usage

In the following example, recovery will not take place for transaction CSFE if the abend code is ATNI or AKC3. Recovery will take place for all other transactions and abend codes.

Code 1	Code 2	Code 3	Value Returned
CSFE	ATNI	*	EXCLUDE
CSFE	AKC3	*	EXCLUDE
*	*	*	INCLUDE

CICSINFO: Display Information

These commands are issued when the INGCICS REQ=INFO command is used to display the state of the selected CICS region. The commands are issued via the MODIFY subsystem ID on an MVS EMCS console and the resulting messages are either displayed on the INGCICS panel or written to the users NetView console.

For further information about the INFO request, see the description of the INGCICS command in *IBM Tivoli System Automation for z/OS Operator's Commands*.

Use User-Defined Processing and enter the following data:

Keyword	Data
CICSCMD	(*description*,*CICS_command*)

Keyword and Parameter Definitions

`description`
This is text that is placed before the output of *CICS_command*. It can be used to identify the command output in the output stream. The description can be any string, but must be enclosed in quotation marks.

`CICS_command`
This is the command that is to be executed. It is appended to a MODIFY

subsystem and issued as an MVS command to an EMCS console. The output is collected and displayed. The command can be any valid CICS transaction that writes to an MVS console. The command must be enclosed in quotation marks.

You may code multiple CICS commands, separated by a comma, in order to group results under a common description.

Comments and Usage Notes

This policy is required for correct operation of the INGCICS REQ=INFO command and also for function key PF10 of the DISPINFO panel.

Examples of Usage

Keyword	Data
`CICSCMD`	`('DISPLAY ACTIVE TASKS','CEMT I TA')`
`CICSCMD`	`('DISPLAY SYSTEM INFORMATION', 'CEMT I SYS')`

LISTSHUT: Transaction Purging During Shutdown

Use this keyword to define those transactions running under this CICS application that should or should not be purged during a shutdown.

Use User-Defined Processing and enter the following data:

Keyword	Data
`EXCLUDE`	* | *transsid*
`INCLUDE`	* | *transid* | (*transid*,FORCE)

Keyword and Parameter Definitions

EXCLUDE
Do not purge this transaction during a shutdown.

*	Specifies any transaction name.
transid	Specifies a specific transaction name.

Any number of transactions may be specified by repeating the EXCLUDE keyword and its data.

INCLUDE
Purge this transaction during a shutdown.

*	Specifies any transaction name.
transid	Specifies a specific transaction name.

(*transid*,FORCE)
Specifies that the transaction named is to be FORCE purged.

Any number of transactions may be specified by repeating the INCLUDE keyword and its data.

Comments and Usage Notes

1. If this entry is not used, no transactions are purged.
2. When the LISTSHUT entry is used for this application, the CICSPURG command list must be coded in the shutdown policy for this application. For information on CICSPURG, see “CICSPURG: Purge Transactions” on page 25.

3. For CICS transactions that are not purgeable (SPURGE=N) the FORCE option must be used to purge the transaction. CICS will return PURGE FAILED for the purge of a non-purgeable transaction.

Examples of Usage

The following example specifies that:

1. The PAYR transaction is specifically not purged during CICS shutdown
2. BAT1 is purged
3. BAT2 is force purged

Keyword	Data
`EXCLUDE`	`PAYR`
`INCLUDE`	`BAT1`
`INCLUDE`	`(BAT2,FORCE)`

RCVRSOS: Short-On-Storage Handling

Use this keyword to notify the operator and optionally issue a command when a short-on-storage condition exceeds the specified time limit.

- Use User-Defined Processing and enter the following data:

Keyword	Data
`TIMELIMIT`	*mm*:*ss*

- Optionally, use Command Processing and enter the following data:

Ps/Select	AutoFn/*	Command Text
		cmd

Keyword and Parameter Definitions

TIMELIMIT
Specifies the time limit that a short-on-storage condition can exist before the commands specified by the CMD attribute are executed.

cmd The command or commands to be issued when the short-on-storage condition exceeds the time limit specified with the TIMELIMIT attribute.

Comments and Usage Notes

1. There is a command list, EVEERDMP, that can be specified for the CMD attribute when a dump is to be produced.
2. The critical value of the RCVRSOS thresholds of the subsystem is used to determine how often the specified commands are allowed to be issued within a specific time frame. The RCVRSOS thresholds must be defined in the customization dialog with the application's MINOR RESOURCES policy item.
3. Pass processing is supported. The pass counter is reset as soon as the short-on-storage situation is reported to be relieved.
4. The defined commands for message ID RCVRSOS are only issued if, in addition, command EVEES100 is defined to be issued in response to messages DFHSM0131, DFHSM0132, DFHSM0133 and DFHSM0134.

It is, however, advisable to replace the technique for the automation of short-on-storage situations with event-based CICSPlex monitoring.

Examples of Usage

The following example shows how the TIMELIMIT attribute can be specified:

Keyword	Data
`TIMELIMIT`	`00:30`

The following shows how the CMD attribute can be specified:

Ps/Select	AutoFn/*	Command Text
		`EVEERDMP CICS1`

The example specifies that the operator is notified and a dump produced if CICS is short on storage for more than 30 seconds.

RCVRTRAN: Transaction Recovery

Use this keyword to define actions to be taken when transaction abends occur.

Use Command Processing and enter the following data:

Ps/Select	AutoFn/*	Command Text
		cmd

Note: The RCVRTRAN keyword may be followed with a dot and a transaction ID *tran*.

Keyword and Parameter Definitions

tran
A specific transaction. If this is not used, then the commands apply to all transactions.

cmd
The command or commands to be issued when the transaction abends.

Comments and Usage Notes

1. The critical value of the TRAN or TRAN.*tranid* thresholds for the subsystem is used to indicate how many abends can occur before automation is stopped. The thresholds must be defined in the customization dialog with the application's MINOR RESOURCES policy item.
2. The SA z/OS variable &EHKVAR1 is set with the name of the transaction. This allows you to tailor your commands using the transaction name, such as disabling the transaction. Also, when available, EHKVAR2 will be set to the abend code and EHKVAR3 set to the program name.
3. The RCVRTRAN entry is used by the EVEERTRN routine that is delivered with CICS Automation. EVEERTRN is called from the NetView automation table by the following messages:

DFHAC2231	DFHAC2232	DFHAC2233	DFHAC2236
DFHAC2245	DFHAC2247	DFHAC2248	DFHAC2249
DFHAC2250	DFHAC2251	DFHAC2252	DFHAC2253

Ensure that these abend transaction messages are issued as a WTO. For more details, see "Defining CICS Messages" on page 4.

Examples of Usage

The following command definition for message ID RCVRTRAN indicates that the command shown is to be executed for all transactions that do not have a specific command definition for message ID RCVRTRAN.*tranid*.

Ps/Select	AutoFn/*	Command Text
		`MSG OP1,TRAN &EHKVAR1 FAILED`

The following example shows the user definitions that are needed to issue a WTO for the abend transaction messages listed in usage note 3:

Keyword	Data
`TDQUEUE`	`CSMT`
`OFFSET`	`1`

Chapter 4. CICS Automation Routines

This section is intended to help system administrators to use CICS Automation functions. It describes the following routines that should be executed from the automation policy:

CICSPURG	Purge transactions at CICS shutdown, see "CICSPURG: Purge Transactions."
EVEERDMP	Creates a system DUMP of the CICS address space, see "EVEERDMP: CICS Dump" on page 26.
EVEES100	Execute defined recovery actions for short-on-storage conditions, see "EVEES100: Recovery Function for Short-On-Storage Conditions" on page 26.
EVEERTRN	Handles transaction recovery, see "EVEERTRN" on page 26.

CICSPURG: Purge Transactions

This command purges running transactions at the time it is issued. It consults the LISTSHUT MESSAGES/USER DATA policy item (see "LISTSHUT: Transaction Purging During Shutdown" on page 21) to determine which transactions are eligible for purging.

CICSPURG should be placed in either the pre-shutdown policy or as the first phase of the shutdown policy. Under normal circumstances the default action that CICS takes to purge transactions at shutdown will suffice to get the CICS subsystem to shut down.

Format

```
CICSPURG [subsys]
```

Keyword and Parameter Definitions

subsys
The name that this CICS subsystem is known to SA z/OS by.

Comments and Usage Notes

If a subsystem name is not specified, the task SUBSAPPL global variable, which is set by AOCQRY, is used.

Examples of Usage

In this example, CICSPURG is used on the second attempt to shut down this subsystem.

Ps/Select	AutoFn/*	Command Text
1		`MVS F &SUBSJOB,CEMT PERFORM SHUTDOWN`
2		`CICSPURG`

EVEERDMP: CICS Dump

EVEERDMP creates a dump for specific CICS problems. It dumps the associated MVS region using the MVS DUMP command. It can be used for situations such as short on storage conditions if you want an MVS dump instead of a CICS internal dump.

Format

```
EVEERDMP {jobname|subsys}
```

Keyword and Parameter Definitions

jobname
The job name for this CICS.

subsys
The name that this CICS subsystem is known to SA z/OS by.

Examples of Usage

Ps/Select	AutoFn/*	Command Text
		EVEERDMP &SUBSAPPL

EVEES100: Recovery Function for Short-On-Storage Conditions

This routine performs CICS short-on-storage recovery actions that are defined for message ID RCVRSOS.

Format

```
EVEES100
```

Comments and Usage Notes

This automation routine can be defined in the automation policy to respond to the following messages for CICS short-on-storage events:

Table 3. Message IDs and Severity for EVEES100

Message ID	Severity
DFHSM0131	CRITICAL
DFHSM0132	NORMAL
DFHSM0133	CRITICAL
DFHSM0134	NORMAL

It can also be invoked from the NetView automation table.

EVEERTRN

The EVEERTRN routine handles transaction recovery. It captures each message listed below with severity NORMAL.

Format

```
►►──EVEERTRN────────────────────────────────────────────────────────────►◄
```

Comments and Usage Notes

The EVEERTRN automation routine is intended to respond to the following messages by issuing the transaction recovery commands that are defined for the RCVRTRAN message ID:

```
DFHAC2231 date time applid Transaction tranid running program program name term
          termid has lost contact with its coordinator system during syncpoint and
          has abended with code ASP1. The unit of work is shunted until contact is
          restored{. EXCI job = }exci_id. condmsg
DFHAC2232 date time applid Transaction tranid running program program name term
          termid has lost contact with its coordinator system during syncpoint and
          has abended with code ASPO. All updates will be unilaterally
          committed{. EXCI job = }exci_id. condmsg
DFHAC2233 date time applid Transaction tranid running program program name term
          termid has lost contact with its coordinator system during syncpoint and
          has abended with code ASPP. All updates will be unilaterally backed
          out{. EXCI job = }exci_id. condmsg
DFHAC2236 date time applid Transaction tranid abend secondary abcode in program
          program name term termid. Updates to local recoverable resources will be
          backed out{. EXCI job = }exci_id. condmsg
DFHAC2245 date time applid A CICS-generated syncpoint request could not be completed
          normally because a connected system has requested that the unit of work be
          rolled back. Transaction tranid running program program name term termid
          has been abnormally terminated with code ASPF{. EXCI job = }exci_id. condmsg
DFHAC2247 date time applid Transaction tranid running program program name term
          termid has requested rollback, but was using a type of processing for which
          rollback is not supported. The transaction has been abnormally terminated
          with code ASP8 {. EXCI job = }exci_id. condmsg
DFHAC2248 date time applid Transaction tranid running program program name term
          termid has failed with abend ASP7 following the failure of a local
          resource owner in the prepare phase of syncpoint. Updates will be backed
          out{. EXCI job = }exci_id. condmsg
DFHAC2249 date time applid Transaction tranid running program program name term
          termid has failed with abend ASP7 following the failure of a remote system
          in the prepare phase of syncpoint. Updates will be backed
          out{. EXCI job = }exci_id. condmsg
DFHAC2250 date time applid the coordinator system has indicated that the current unit of work
          is to be backed out. Transaction tranid running program program name term
          termid has been abnormally terminated with abend ASP3{. EXCI job = }exci_id.
          condmsg
DFHAC2251 date time applid Transaction tranid running program program name term
          termid has failed with abend ASPQ. Syncpoint commit processing has failed
          while communicating with a remote system{. EXCI job = }exci_id. condmsg
DFHAC2252 date time applid Transaction tranid in program program name term termid
          has lost contact with its coordinator system during syncpoint processing.
          No updates have been performed by this system; it has abended with code
          ASPR{. EXCI job = }exci_id. condmsg
DFHAC2253 date time applid Transaction tranid running program program name term
          termid has failed with abend ASP2 due to the links to the remote systems
          being in an invalid state. Updates will be backed out{. EXCI job = }exci_id.
          condmsg
```

Note that these messages are not normally issued to the system console, so if transaction recovery is required the messages should be included in the MESSAGES/USER DATA policy so that the CICS message exit forces CICS to WTO them.

For more details, see the sections "How to Define Transaction Recovery" in the chapter "How to Set Up Functions of CICS Automation" in *IBM Tivoli System Automation for z/OS Product Automation Programmer's Reference and Operator's Guide*.

Chapter 5. INGCICS Operator Command

INGCICS is the CICS-related SA z/OS operator command that can be invoked by operators or via PIPEs.

INGCICS: Issue List of Defined Transactions and View the Output

Purpose

The INGCICS command lets you:

- Issue any console-enabled CICS transaction
- Broadcast messages to all or selected CICS users
- Issue a list of defined transactions and view the output
- Display the output of CICS transactions in full-screen or pipeable line mode

For a detailed description of the INGCICS command, see *IBM Tivoli System Automation for z/OS Operator's Commands*.

Chapter 6. Monitor Command

The following routine should be used as a MONITOR command when defining monitor resources for CICS Automation.

INGRMCDB Routine for the Monitoring of CICS DB2 Connections

This routine should be used as the monitor command for a monitor resource. It is issued after each monitoring interval to check whether CICS is connected to DB2. To achieve this, the CICS command CEMT INQUIRE DB2CONN is issued to query the status of the CICS DB2 connection.

The health status is considered to be NORMAL when the status of the monitored CICS DB2 connection is Connected. A degraded health status is returned in all other cases.

Format

```
►►──INGRMCDB──────────────────────────────────────────────────────────►◄
```

Restrictions

The INGRMCDB routine is assumed to be defined as the monitor command for a monitor resource with the following restrictions:

- One of the following is specified in the Monitored Object field:
  - DB2
  - The MVS subsystem ID of the DB2 subsystem
  - The DB2 group name
- A job name is specified for the monitor resource.
- The monitor resource is related to a CICS subsystem.

Return Codes

Depending on the monitor results, INGRMCDB ends with the following return codes, which are mapped to the related health status for the monitor resource.

Return Code	Health status	Description
1	BROKEN	No DB2 connection is defined in CICS
2	FAILED	CEMT INQUIRE DB2CONN command failed
3	NORMAL	CICS is connected to DB2
5	MINOR	CICS is not connected to DB2

Part 2. DB2 Automation

This part gives an overview of the functions provided by DB2 Automation and how to set them up. It also provides a more detailed description of the line command requests that can be performed against DB2 subsystems.

It contains the following chapter:

Chapter 7. DB2 Automation for System Automation for z/OS

This chapter describes the principal concepts of DB2 automation in SA z/OS and gives some hints concerning its customization.

*DB2 Best Practices Policy

SA z/OS supplies a sample policy, *DB2, that provides best practice definitions for running DB2 automation. It shows a solution for a DB2 data sharing group with multiple DB2 subsystems that are restartable on another system in light mode.

Diagrams of the sample policies are provided as PDF files that are located in the USS installation path. The default for this path is: /usr/lpp/ing/doc/policies/.

You customize DB2 automation for your specific installation by modifying the policy database in the customization dialog. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* and the online documentation to the *DB2 sample policy for details about how to import the *DB2 best practices policy and modify the definitions that this import adds to your policy database.

Customization Hints

Bear in mind the following hints when customizing the *DB2 best practices policy:

- All DB2 address spaces must be defined to SA z/OS as APPLICATION objects. The *DB2 best practices policy provides you with a number of sample classes and instances of entry type APL that model the DB2 subsystem and its dependent address spaces. Copy and modify them to tailor them to your needs:
  - The names of the APL classes in *DB2 have the prefix C_DB2
  - The names of the APL instances in *DB2 have the prefix DB2
- For the following address spaces of DB2, the Application Type field must be set to DB2 and the appropriate subtype:
  - DB2 Master
  - DB2 Database Services
  - DB2 Distributed Data Facility
  - DB2 Resource Lock Manager
- Applications of type DB2 and subtype MSTR have an additional DB2-specific Policy Item, DB2 CONTROL, that allows you to specify the DB2 subsystem ID and parameters for controlling DB2 automation.

Access Authority for Automation Work Operators

| DB2 commands are issued by the SA z/OS automation work operator (AUTWRKnn) that the DB2 subsystem has been assigned to. Because these operators are assigned automatically during ACF load each of the AUTWRKnn operators that are defined to automation should be granted SYSOPR authority to DB2.

| When using OPERSEC=SAFCHECK, or SAFDEF (user level security), the following list of operator IDs should also be granted SYSOPR authority to DB2:

- | GSSOPER
- | RPCOPER

- AUTO1
- SYSOPER (backup task for GSSOPER)
- BASEOPER (backup task for SYSOPER and GSSOPER)

Critical Event Monitoring

Purpose

Critical Event Monitoring handles specific critical events that may occur during normal day-to-day running of DB2. These include:

Message	Description
DSNB250E, DSNB311I, DSNB312I, DSNB320I, DSNB321I, DSNB322I, DSNB323I, DSNB350I, DSNB351I	Recover failed dataspace (for data sharing only)
DSNJ002I	Switch active log data sets
DSNR004I	RESTART...UR STATUS COUNTS
DSNP007I	Data set could not be extended
DSNJ110E	Last active log data set is % full
DSNJ111E	All active log data sets full
DSNJ115I	Archive data set could not be allocated
DSNT500I/DSNT501I	Resource unavailable

Recovery commands that are required should be defined for the message in the automation's MESSAGES/USER DATA policy item for any DB2 subsystem that requires DB2 critical event message recovery.

If the DB2 subsystem is known to SA z/OS as an application of type DB2, event message recovery can be controlled by parameters entered via the DB2 CONTROL policy item for the subsystem.

Restrictions

Critical event monitoring is only done if the DB2 subsystem is defined to SA z/OS and if recovery is enabled for the invoking message ID minor resource.

For some recovery actions SYSOPR needs SYSCTRL authority.

Usage

For each of the Event Monitoring processes, the input parameters are validated for accuracy and any errors that are found are logged and the process is terminated. The requested subsystem is then checked to determine if automation is enabled for the major resource, and also that recovery is enabled for the invoking message ID minor resource. Once these preliminary checks have been successfully completed, the requested recovery action is performed.

DSNB250E, DSNB311I, DSNB312I, DSNB320I, DSNB321I, DSNB322I, DSNB323I, DSNB350I, DSNB351I: Recover Failed Dataspace

Description: You may want to start table spaces in response to these messages. To do this, specify the start command in the MESSAGES/USER DATA policy item for these messages in your automation policy as follows: INGDB2 TABLE &SUBSAPPL START

DSNJ002I: Switch Active Log Data Sets

Description: If commands are defined in the automation policy item MESSAGES/USER DATA for this message to an application of type DB2, these commands are only issued when the triggering message is for the log data set that is specified by the "Active log data set name" in the DB2 CONTROL policy item.

If the application is not of type DB2, the defined commands are issued unconditionally.

DSNR004I: RESTART...UR STATUS COUNTS

Description: If the message reports an INDOUBT counter greater than zero at the end of a DB2 restart process, ISSUEACT is called to issue commands that are defined in the automation policy item MESSAGES/USER DATA of the DB2 application for this message.

DSNP007I: Data Set Could Not Be Extended

Description: The created automation table statement calls ISSUEACT with its code specifications as the parameters. The code values are extracted from the message text. The data set name is passed as the CODE1 value, the return code is passed as the CODE2 value, and the connection ID is passed as the CODE3 value. If a code match is found with the ID of the triggering message in policy item MESSAGES/USER DATA, the value that is returned is used to select and issue the related commands as defined in the automation policy.

DSNJ110E: Last Active Log Data Set Is % Full

Description: If commands are defined in the automation policy item MESSAGES/USER DATA for this message to an application of type DB2, these commands are only issued when the message reports a percentage full figure that is equal to or greater than the critical threshold that is defined in the "Log full threshold" field of the DB2 CONTROL policy item.

If the application is not of type DB2, the defined commands are issued unconditionally.

DSNJ111E: All Active Log Data Sets Full

Description: If commands are defined in the automation policy item MESSAGES/USER DATA for this message to an application of type DB2, these commands are only issued when the number of received messages within a time period exceeds a given threshold. The time period and the threshold can be entered via the DB2 CONTROL policy item in the "Active log alerts" field and the related "Threshold" field.

If the application is not of type DB2, the defined commands are issued unconditionally.

DSNJ115I: Archive Data Set Could Not Be Allocated

Description: If commands are defined in the MESSAGES/USER DATA automation policy item for this message for an application of type DB2, these commands are only issued when the elapsed time since they were triggered by this message is greater than a given time interval, as specified in the **Log offload interval** field in the DB2 CONTROL policy item.

If the application is not of type DB2, the defined commands are issued unconditionally.

DSNT500I/501I: Generate DSNT500I/DSNT501I Alert

Description: The created automation table statement calls ISSUEACT with its code specifications as the parameters. The code values are extracted from the message text. The name is passed as CODE1, the reason is passed as CODE2 and the type is passed as CODE3. If a code match is found with the ID of the triggering message in policy item MESSAGES/USER DATA, the value that is returned is used to select and issue the related commands as defined in the automation policy.

INGDB2 Automation Utility

INGDB2 is an automation utility that can be invoked by operators or via the AT.

The INGDB2 utility lets you:
- Start or stop a table space
- Terminate active threads
- Inform TSO users that their thread is about to be terminated
- Check for indoubt threads

Based on the condition used the following messages are captured:

Table 4. Messages captured for INGDB2 utility

Message ID	Severity
ING114	CRITICAL
ING129	IMPORTANT

For a detailed description of the INGDB2 utility, see *IBM Tivoli System Automation for z/OS Programmer's Reference*

Part 3. IMS Automation

This part describes the principal concepts of SA z/OS with regard to IMS Automation and describes the steps that are necessary to customize and set up IMS Automation. The description focuses on those steps that are specific to IMS Automation and have to be done in addition to the definition of the basic functions of SA z/OS that are common to all subsystems. Furthermore, it contains reference sections for IMS-specific message IDs in the MESSAGES/USER DATA policy item and for common routines that request information or perform tasks associated with IMS Automation.

It also describes the tasks of the operator who manages IMS subsystems through IMS Automation.

It consists of the following chapters:

Chapter 8. IMS Automation Concepts

This section describes the principal concepts of IMS Automation in SA z/OS and gives some hints relating to its customization.

*IMS Best Practices Policy

SA z/OS supplies a sample policy, *IMS, that provides best practice definitions for running IMS Automation.

The *IMS sample policy builds in FDR to provide best practices for IMS Automation in an IMSplex. FDR provides superior, sysplex-aware automated recovery within IMS. It is the recommended solution for high availability of IMS applications.

Diagrams of the sample policies are provided as PDF files that are located in the USS installation path. The default for this path is: /usr/lpp/ing/doc/policies/.

You customize IMS Automation for your specific installation by modifying the policy database in the customization dialog. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* and the online documentation for the *IMS sample policy for details about how to import the *IMS best practices policy and modify the definitions that this import adds to your policy database.

Customization Hints

Bear in mind the following when customizing the *IMS best practices policy:

- The *IMS best practices policy supplies the automation operator definition that is required for IMS automation. It is located in the IMS_AUTO_OPS policy object for the AOP entry type.

Automated Function	**Operator ID**	**Message Classes**
IMSMSTR	AUTIMS	EVI*

  Make sure that this operator ID is defined as a NetView operator.
- All IMS regions must be defined to SA z/OS as APPLICATION objects. The *IMS best practices policy provides you with a number of sample classes and instances of entry type APL that model IMS regions. Copy and modify them to your needs:
  - The names of the APL classes in *IMS have the prefix C_IMS
  - The names of the APL instances in *IMS have the prefix IMS
- For the following IMS regions, the Application Type field must be set to IMS:
  - IMS Control regions
  - IMS dependent regions for Fast Path and Message Processing regions
  - IMS DL/I Secondary Address Space
  - IMS Database Recovery Control
- Applications of type IMS have an additional IMS-specific policy item, IMS CONTROL, that allows you to specify the IMS subsystem ID and the IMS control region type and the IMSplex name.

IMS Operation Hint

In the *IMS best practices policy, relationships have been set up, for example, from monitor resources to the IMS application group, to make sure that the monitor resources can rely on all required IMS functions.

Be aware that the shutdown of the IMS control region may not be performed because of these defined resource dependencies. You should therefore address the shutdown request to the IMS application group instead of the IMS control region to avoid hanging shutdown requests.

IMS Message Processing

SA z/OS can only automate messages that are issued via WTO. Most IMS system messages that are important are WTO'd, However, there are many IMS messages that are only logged internally. Some of these messages may be useful in automation situations. To enable SA z/OS to process these messages, exits are installed to WTO messages that would not be WTO'd by IMS. In addition, your own user code might produce messages that are written to IMS. Some of these messages might be of interest in automation situations.

SA z/OS installs an exit to WTO these messages. SA z/OS installs an exit for the AOE Type 2 exit of IMS Control regions.

The messages that are WTO'd from this exit are defined in the MESSAGES/USER DATA policy for the subsystem or the subsystem class.

Installing the IMS Message Exits

IBM Tivoli System Automation for z/OS Planning and Installation details the basic installation steps to install the exits into IMS. This section details the various parameters and commands that can be used to control the exits.

z/OS Exit Router Information

The out-of-the-box configuration of the exit uses the z/OS exit router to enable the user to specify exit modules at three exit points. The exit program is EVIPVEX0 and has a pre-built alias of DFSAOE00. The exit points defined are:

1. **DFSAOE00.CMD**

   This is invoked whenever DFSAOE00 is called with AOE0FUNC=1 to initialize the routines and also when AOE0FUNC=2 is called with AOE0FLG2=X'80' (command entered at a terminal), X'20' (ICMD command) or X'10' (internal command).
2. **DFSAOE00.MSG**

   This is invoked whenever DFSAOE00 is called with AOE0FUNC=1 to initialize the routines and also when AOE0FUNC=2 is called with AOE0FLG2=X'08' (message segment).
3. **DFSAOE00.CMDRESP**

   This is invoked whenever DFSAOE00 is called with AOE0FUNC=1 to initialize the routines and also when AOE0FUNC=2 is called with AOE0FLG2=X'40' (command response segment).

Specification of routines to run at each exit point is done via PROG*xx* members of SYS1.PARMLIB or via the SETPROG EXIT command.

You may enable or disable exits dynamically at any time.

A required definition of EVIPVEX1 is needed to process the exit information for SA z/OS. However, any number of exit routines may be added at any of the points. The z/OS exit router will execute them one after the other. If exit routines are not specified for an exit point, no action will be taken.

The definitions of the required exit points are:

```
EXIT ADD EXITNAME(DFSAOE00.MSG) MODNAME(EVIPVEX1) DSNAME(ING.SINGMOD1) STATE(ACTIVE)
EXIT ADD EXITNAME(DFSAOE00.CMD) MODNAME(EVIPVEX1) DSNAME(ING.SINGMOD1) STATE(ACTIVE)
```

See *z/OS MVS Initialization and Tuning Reference* for additional parameters that can be supplied to the EXIT statement.

The SA z/OS exit routines EVIPVEX0 and EVIPVEX1 use the last three words of the storage pointed to by SXPLAWRK. If these values are changed unpredictable results may occur.

The SA z/OS exit router will set AOE0RPLY to 1 if there are no exit routines enabled for an exit and a DFSAOUE0 module is present in the IMS system. This rule applies to each exit point independently. This means that the user can disable an exit point and still have their DFSAOUE0 module invoked. In addition if the exit point invoked is EVIPVEX1 as specified, see “Using the SA z/OS Exit without the z/OS Exit Router” for details on invoking DFSAOUE0. Note that it is possible using the z/OS exit router to invoke DFSAOUE0 for all three exit points. This can be achieved by the default definition that does not define an exit point for DFSAOE00.CMDRESP or by disabling this exit point.

Using the SA z/OS Exit without the z/OS Exit Router

As detailed in the installation manual, it is possible to use the SA z/OS exit in a standalone manner. In this mode, no z/OS exit router is enabled and dynamic management of the exit is not possible. The procedure to enable this function is detailed in the System Automation installation manual. It basically re-defines the DFSAOE00 alias from the EVIPVEX0 module to the EVIPVEX1 module.

The SA z/OS exit will set AOE0RPLY to 1 if there is a DFSAOUE0 module present in the IMS system and any of the following conditions are met:

- AOE0FUNC=2 and AOE0FLG2=AOE0MSGS
- AOE0FUNC=2 and AOE0FLG2=AOE0TCMD or AOE0ICMD or AOE0INTC

In effect this means that every system Message or Command will also invoke DFSAOUE0 if it is present. Command Responses will not invoke DFSAOUE0. It is not possible to invoke DFSAOUE0 for command responses in this mode of exit operation. See “z/OS Exit Router Information” on page 44 or “Calling the SA z/OS Exit from Your DFSAOE00 Module” on page 46 for alternatives.

Due to an IMS restriction, when DFSAOUE0 is present in the environment only the first segment of a multi-segment message will be presented to DFSAOE00. Therefore, tokens that are defined in the policy database must be present in the first segment to be matched. However, all segments will be passed to DFSAOUE0.

Note:
Make sure that the ING.SINGMOD1 library is not concatenated before of the library that you have redefined the alias in. If it is, the z/OS exit router module will be enabled at the DFSAOE00 exit point.

Calling the SA z/OS Exit from Your DFSAOE00 Module

If neither of the above methods suits your installation, it is possible to directly call the EVIPVEX1 module from your DFSAOE00 exit routine.

Input parameters are:

1. Register 1 points to the SXPL (standard Exit Parameter List) as supplied by IMS to the DFSAOE00 exit.

| **Used storage:** The exit uses the last two words of the storage that is pointed to by SXPLAWRK. The last word is reserved for internal SA z/OS control information. If the values are changed, unpredictable results may occur.

You are responsible for the correct setting of the AOE0RPLY field. Note that after calling EVIPVEX1, the AOE0RPLY field will be set as specified in "Using the SA z/OS Exit without the z/OS Exit Router" on page 45.

A sample call is as follows:

```
LA    R1,SXPL           ; Load address of SXPL
L     R15,=V(EVIPVEX1)  ; get address of routine
BALR  R14,R15           ; invoke the exit
```

Using the IMS Automation Message Exit

To enable the IMS message exit to process messages, the messages must be defined in the policy database. The MESSAGES/USER DATA policy item is used to define the messages. If there are no messages defined to use the exits, the SA z/OS exit will not process any messages.

Each subsystem has its own policy and does not share policy information, except for CLASS information, with other subsystems. This means that you can update the policy information for a subsystem or set of subsystems and will not affect other subsystems.

Defining IMS Messages

There are several special keywords that, when added to the message with the USR action, cause the message information to be loaded into the appropriate exit for processing. The keywords and their descriptions are described in Table 5.

Table 5. Keywords in IMS Message Definitions

Keyword	Description
OFFSET	This keyword is required to load the message into the exit. It specifies the word number in the message that represents the message ID. Its format is a single integer number.

Table 5. Keywords in IMS Message Definitions (continued)

Keyword	Description

MSGDISP	This keyword specifies the default actions for a message ID. The format of the data is as follows:
	`►►──(──┤ IMS Actions ├──)──────────────────►◄`
	IMS Actions:
	`├──┬─NONE──────────────────────────────┬──┤`
	`   └─┬─AUTOMATE─┬──,──┬─FORWARD──┬─┘`
	`     └─AUTO─────┘     └─SUPPRESS─┘`
	The actions that are available are:
	NONE The message is not acted on by the exit. Use this option to set a default of no actions when used with the TOKEN keyword.
	AUTOMATE|AUTO WTO the message for SA z/OS to trap.
	FORWARD This forwards the message segments to the TYPE 1 exit DFSAOUE0, if it exists.
	SUPPRESS The message is suppressed from the IMS Master Console. IMS reserves the right not to suppress messages that it determines are critical.
	The default action is AUTOMATE if no TOKENs are present. If TOKENs are present the default action is NONE.

TOKEN	This keyword specifies the matching token values for user messages. It is optional, but if present only messages that match the values specified are WTO'd. Its format is:
	`►►──(──┤ Tokens ├──┬─────────────────────┬──)──────►◄`
	`                   └──┤ IMS Actions ├──┘`
	Tokens:
	`   ┌─,──────────────────────┐`
	`   ▼                        │`
	`├────(number,word_text)─┴──────────────────┤`
	IMS Actions:
	`├──┬─NONE──────────────────────────────┬──┤`
	`   └─┬─AUTOMATE─┬──,──┬─FORWARD──┬─┘`
	`     └─AUTO─────┘     └─SUPPRESS─┘`
	Where *number* is the word number to be checked and *word_text* is the text to be checked. Words are counted from the beginning of the message starting from 1. A word is considered to be a contiguous set of characters that contain alphanumerics or any of the '$#§_Ü?' characters. Any other characters are treated as word delimiters.
	The *word_text* specification is checked for the length specified. This allows for trailing data to be ignored. The format of the number is a single integer. The format for the value is a single alphanumeric word; no blanks or special characters are allowed. The case of the value is ignored.

To define a message to be WTO'd, do the following:

1. Specify the message ID in the MESSAGES/USER DATA policy item.
2. Enter the USR command against the message ID. Specify OFFSET in the **Keyword** field and the word number that the message ID occurs at in the message in the **Data** field.
3. Optionally, enter the USR command against the message ID and specify TOKEN in the **Keyword** field, and the words and their values that are to be matched in the **Data** field.

Refreshing Policy Data

To load the information into IMS address spaces issue the INGAMS REFRESH command. As a part of the ACF load, each IMS that had changes to MESSAGES/USER DATA policy will be reloaded with any changes.

If you want to disable the exits, delete the messages from the policy.

Partial Message IDs and Performance

A partial message ID is one where only a part of the message ID is specified. The partial message ID is restricted to the leftmost part of the message ID. You cannot specify a partial message ID, such as `XXX**YY`, only `XXX` would be acceptable. The first three bytes of the message ID are the minimum required to specify a message to trap in the exit. You can specify a partial message ID within this limitation.

It is recommended that you do not use the TOKEN keyword with partial message IDs for messages. This causes a performance degradation because the token data must be matched for the partial message and then again for any specific message.

You can specify MSGDISP policy information on actions for all specific message IDs that match the partial message ID. You do not have to specify more specific message IDs. Instead a partial message ID and a matching set of TOKEN keywords can be used to specify the actions for a message. Be aware that token scanning is much less efficient than message ID scanning.

You can use the partial message IDs to improve exit performance. An extreme example might be if you are trapping messages ABC000-ABC100. You might choose to create partial message IDs for ABC00-ABC10. The partial message IDs would specify MSGDISP=(NONE) to prevent them supplying default actions. This would cause the exit to first scan the set of partial message IDs ABC00-ABC10 or 11 checks. Then, assuming it found a match on ABC10, it would scan the specific messages ABC110-ABC119 for a total of 21 checks as opposed to 101 for the original set. This would require 112 MESSAGES/USER DATA policy definitions. 101 for messages ABC000-ABC100 and 11 for messages ABC00-ABC10. In normal circumstances you can ignore the usage of partial message IDs for performance reasons and use them for functional reasons.

Defining IMS Actions

To define automated actions in the SA z/OS automation policy that have to be executed by IMS, it is recommended that you specify an IMS command, prefixed with the IMS subsystem ID, and issue it with the NetView MVS command. The subsystem ID of the IMS control region can be retrieved from the &SUBSSUBID task global variable.

When specifying the command in the application policy object of a dependent IMS region that has a HasParent relationship to the IMS control region with a defined sequence number, the command can be prefixed with the &SUBPSUBID task global variable to address it to the associated IMS control region. You must make sure that the IMS control region is first in the list of defined parents.

If a Command Prefix has been defined in the APPLICATION INFO policy item for the IMS control region, you can also use the &SUBSCMDPFX task global variable to prefix the IMS command instead of the subsystem ID.

Example

The POSTSTART commands for the IMS control region can be specified as follows:

Type	AutoFn/*	Command Text
		`MVS &SUBSSUBIDCQSET SHUTDOWN SHAREDQ ON STRUCTURE ALL`
		`MVS &SUBSSUBIDSTA DC`
		`MVS &SUBSSUBIDCHE`

IMS actions can also be defined to be issued as replies to the outstanding WTOR of the IMS control region. This method is only applicable to the DC control region because the DB control region does not allow communication via an outstanding WTOR. Furthermore, issuing a sequence of actions leads to a processing bottleneck. This is because each time it issues a reply, SA z/OS first has to wait for the next outstanding WTOR before it can continue with issuing the next reply. This method is therefore not recommended.

Chapter 9. How to Set Up the Special Functions of IMS Automation

This chapter explains how to set up the special functions of IMS Automation for your specific needs. For the setting up of base functions, like starting and stopping subsystems, see *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Automating Recovery for Application Components

IMS Automation provides automated recovery for transactions and programs and also enables monitoring of:

- Online log data sets (OLDS)
- Recovery control data sets (RECON)
- VTAM® Access Method Control Block (ACB)
- IMS DB2 connections

How to Define Transaction Recovery

Customization of transaction recovery consists of:

- Determining which application program (TP) transactions will have recovery automation
- Identifying the batch message region (BMP) transactions that will have recovery automation
- Specifying the error threshold level that a recovery should stop at
- Identifying specific abend codes that you want recovery procedures to occur for
- Specifying the recovery procedure, which usually consists of invoking a command, a routine, and/or sending notifications to an operator

The recovery itself is typically triggered from the NetView automation table by calling the EVIECT0X routine when certain messages arrive at NetView. EVIECT0X then consults the automation policy to find out whether recovery is to be attempted and to determine what has to be done.

The following sections illustrate the configuration process.

Specifying the Transactions or Programs to Be Recovered

In this example, suppose that recovery is enabled for the IMS10AA application on the application level, and that you also want it enabled for transactions PAYR, DBTS, and BLNG, but not for any other transaction. You must then define recovery automation flags using the MINOR RESOURCES policy item for IMS10AA in the customization dialog for the following four minor resources:

- TRAN
- TRAN.BLNG
- TRAN.DBTS
- TRAN.PAYR

To do this, enter `f` in the **Cmd** field and the minor resource name in the **Minor Resource** field on the Minor Resource Definitions panel and press Enter. This displays the Automation Flag Specification panel for the minor resource. Here you specify which flags are set and which are not. Set the recovery automation flag to

NO for TRAN and YES for the three second-level minor resources. For more information, see *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

You can fine-tune recovery automation for programs in the same way by replacing the `TRAN` keyword with `PROG`.

Defining Recovery Thresholds

You can specify that recovery is to be stopped when the number of abends within a certain time interval reaches a certain threshold. To do this, define thresholds in the MINOR RESOURCES policy item of the APPLICATION policy object.

The thresholds must have the name TRAN or TRAN.*tranid*, where the values of the TRAN thresholds will be used for all transactions *tranid* for which no TRAN.*tranid* thresholds exist. The **Critical** value of the thresholds will be used.

If you want to stop recovery specifically for PAYR if two or more abends occur within one hour, you must enter the values on the Thresholds Definitions panel, as follows:

	Critical		**Frequent**		**Infrequent**	
Resource	Number	Interval (hh:mm:ss)	Number	Interval (hh:mm:ss)	Number	Interval (hh:mm:ss)
`IMS10AA.TRAN.PAYR`	`2`	`01:00:00`	`2`	`05:00:00`	`2`	`24:00:00`

For more details, see *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

For recovery thresholds of programs, you must use `PROG` instead of `TRAN` as the first part of the thresholds' name.

Selecting the Abend Codes

The abend codes that recovery is to take place for are specified in the MESSAGES/USER DATA policy item for IMS10AA through the following keywords:

- The ABCODETRAN keyword for transactions
- The ABCCODEPROG keyword for programs

If you want to initiate recovery for transaction PAYR only when the abend code is U3033 or U907, you need to create an ABCODETRAN.PAYR entry in the MESSAGES/USER DATA policy item and use action Cod on it. This takes you to the Code Processing panel where you enter the following definitions:

Code 1	**Code 2**	**Code 3**	**Value Returned**
`*`	`U3033`	`*`	`INCLUDE`
`*`	`U907`	`*`	`INCLUDE`
`*`	`*`	`*`	`EXCLUDE`

For more details, see "ABCODETRAN: Transaction Abend Recovery" on page 62.

In a similar way, you can use the ABCODEPROG keyword to include certain abend codes for certain programs in recovery automation.

Specifying Recovery Actions

You specify the commands to be issued for recovery on the Command Processing panel for the DFS554A message ID of IMS10AA. For example:

Ps/Select	AutoFn/*	Command Text
`PROG`		`MVS &SUBSSUBIDSTA PGM &EHKVAR2`
`TRAN`		`MVS &SUBSSUBIDSTA TRAN &EHKVAR1`

For more details, see "DFS554A: Respond to Program Abend" on page 63.

Monitoring of Online Log Data Sets (OLDS)

IMS automation allows the monitoring of the online log data sets of IMS control regions and execution of recovery actions if needed. The IMS display command is issued to analyze the status of the OLDS data sets. If needed, OLDS data sets are started or stopped. Any OLDS data sets that have been defined as spare OLDS data sets are started last and stopped first.

The IMS display command can be triggered by incoming messages that report OLDS-related activities, or it can be issued at regular time intervals.

The implementation of the OLDS monitoring function includes the following:

- A monitor resource to monitor the number of available OLDS
- A monitor resource to monitor for excessive switching of the OLDS
- Input parameters for the OLDS monitor function, defined as user data for the message ID OLDS
- Status messages for passive OLDS monitoring
- Command definitions for the health status update that are dependent on the OLDS switch frequency
- Threshold definitions
- Automation flag settings
- Relationships

The OLDS monitoring function is optional, that is, some definitions are necessary to enable it. For activation the following implementation steps are needed:

IMS OLDS Monitor Resource

The IMS OLDS monitor resource has to be defined with the MTR policy object with the following characteristics:

Field Name	Field Value
Monitored Object	OLDS
Monitored Jobname	*jobname* of the IMS control region
Monitor command	INGRMIOL
Monitoring Interval	*hh:mm* (optional)

Without a specified monitoring interval, the monitor command is issued initially when the monitor resource is started.

Depending on the monitor results, INGRMIOL ends with the following return codes, which are mapped to the related health status for the monitor resource.

Return Code	Health status	Description
1	BROKEN	Monitor encountered a severe error
2	FAILED	DISPLAY OLDS failed
3	NORMAL	No problem found by OLDS monitoring
4	WARNING	One of the following occurred: • Needed to start spare OLDS to have the minimum in AVAILABLE status • AUTOMATIC ARCHIVE is off
5	MINOR	Could not start enough spare OLDS to have the minimum in AVAILABLE status
6	CRITICAL	Number of OLDS in BACKOUT status exceeds maximum limit

Message ID OLDS

The input parameters for OLDS monitoring have to be defined as user data for the special message ID OLDS with the MESSAGES/USER DATA policy item of the IMS control region that the monitor resource is associated with.

Keyword	Data	Description
MINIMUM	*nn*	The minimum number of OLDS that must be available at all times. The default minimum number is 50% of the normal number of OLDS, where the normal number of OLDS is given in the response of the DISPLAY OLDS command.
SPARES	(*nn*,*nn*...)	The spares are OLDS that IMS automation activates when the number of available OLDS drops below the minimum and it is not sufficient to start the stopped OLDS to have the minimum number of OLDS available. The names of the spares are the two-digit numbers taken from the end of the ddname. For example, DFSOLP99 is the spare named 99. Be sure that the names of the spares match the names of existing OLDS.
BACKOUT	*nn*	The maximum number of OLDS that can have an OTHER-STS of BACKOUT. Set this number to the total number of acceptable OLDS data sets with an OTHER-STS of BACKOUT.

Status Messages

Apart from optional active monitoring with the INGRMIOL monitor command after each monitoring interval, the OLDS monitoring function can be implemented with passive monitoring. The following IMS messages can be used to trigger health status updates and recovery actions:

```
DFS3256I OPEN/ALLOCATION FAILED ON ddname
DFS3257I ONLINE LOG NOW SWITCHED - FROM DFSOLPxxx TO ddname2
DFS3258A LAST ONLINE LOG DATA SET IS BEING USED - NEED ARCHIVE
DFS3260I ONLINE LOG DATA SET SHORTAGE - NEED ANOTHER DATA SET
```

Messages DFS3256I, DFS3257I and DFS3260I only indicate OLDS-related activities that may or may not change the health status of the monitor resource. Therefore the defined monitor command first has to be issued to analyze the actual situation and to evaluate the health status before it can be updated. To achieve this, these messages have to be defined as status messages in the MESSAGES/USER DATA policy item of the OLDS monitor resource with the status `Check`.

By contrast, message DFS3258A informs about a concrete problem situation. It can be directly mapped to a degraded health status by selecting CRITICAL as the new health status with the AUT action in the MESSAGES/USER DATA policy item of the OLDS monitor resource.

Automation Flag Settings

Before starting or stopping OLDS data sets, the OLDS monitoring function first checks the Recovery automation flag of the OLDS minor resource for the IMS control region. The recovery actions are only issued if they are allowed by the automation flag.

IMS OLDS Switch Frequency Monitor

The IMS OLDS switch frequency monitor resource has to be defined in the MTR policy object with the following characteristics:

Field Name	Field Value
Monitored Object	OLDS_SWITCH
Monitored Jobname	*jobname* of the IMS control region
Monitor command	(none)
Monitoring Interval	*hh:mm* (optional)

The switch frequency is determined by passive monitoring of message DFS3257I.

Command Definitions for the Health Status Update Related to the Switch Frequency

To set the health status of the monitor resource, depending on the frequency of the received message DFS3257I, define the following commands for message DFS3257I in the MESSAGES/USER DATA policy item of an IMS control region:

Ps/Select	AutoFn/*	Command Text[1]
`INFR`		`INGMON OLDS_SWITCH,JOBNAME= &SUBSJOB,STATUS=WARNING,INFO=(MSG,INFREQUENT THRESHOLDS LIMIT REACHED FOR OLDS SWITCHING)`
`FREQ`		`INGMON OLDS_SWITCH,JOBNAME= &SUBSJOB,STATUS=MINOR,INFO=(MSG,FREQUENT OLDS SWITCHING DETECTED)`
`CRIT`		`INGMON OLDS_SWITCH,JOBNAME= &SUBSJOB,STATUS=CRITICAL,INFO=(MSG,CRITICAL OLDS SWITCHING FREQUENCY REACHED)`
`ALWAYS`		`INGMON OLDS_SWITCH,JOBNAME= &SUBSJOB,STATUS=NORMAL,INFO=(MSG,OLDS SWITCHING FREQUENCY IS NORMAL)`

1. The **Command Text** field is spread across two screens.

Threshold Definitions

Define the threshold levels for the degradation of the health status for the minor resource DFS3257I in the MINOR RESOURCES policy item of the IMS control region.

Relationships

To propagate the health status of the monitor resources to the health status of the application, HASMONITOR relationships have to be defined between the IMS control region and both monitor resources.

The monitor resources are kept active only during the UP time of the IMS application group. This is achieved with an additional HASPASSIVEPARENT and a FORCEDOWN/WhenObservedDown relationship between each monitor resource and the IMS application group. This ensures that the monitor resources can rely on all required functions.

Monitoring of Recovery Control Data Sets (RECON)

IMS Automation allows the monitoring of recovery control data sets of IMS control regions. The IMS command `RMLIST DBRC='RECON STATUS'` is issued regularly to analyze the status of the listed RECON data sets in the response to this command.

The RECON monitoring function is optional, that is, some definitions are necessary to enable it.

The implementation of the RECON monitoring function includes the following:

- A monitor resource to monitor the number of available RECON data sets
- A status message for passive RECON monitoring

IMS RECON Monitor Resource

The IMS RECON monitor resource has to be defined in the MTR policy object with the following characteristics:

Field Name	Field Value
Monitored Object	RECON
Monitored Jobname	*jobname* of the IMS control region
Monitor command	INGRMIRE
Monitoring Interval	*hh:mm*

The monitor command is executed after each monitoring interval. It issues the following command and analyzes the status of the listed RECON data sets in the response to this command:

```
RMLIST DBRC='RECON STATUS'
```

Depending on the monitor results, INGRMIRE ends with the following return codes, which are mapped to the related health status for the monitor resource:

Return Code	Health status	Description
1	BROKEN	Monitor encountered a severe error
2	FAILED	RMLIST DBRC='RECON STATUS' failed
3	NORMAL	No problem found by RECON monitoring
4	WARNING	No COPY2 found for RECON
5	MINOR	No SPARE found for RECON
6	CRITICAL	Neither COPY2 nor SPARE, and possibly no COPY1 found for RECON

Status Message

Apart from active monitoring with the INGRMIRE monitor command after each monitoring interval, RECON monitoring function can be implemented with passive monitoring. The following IMS message can be used to trigger health status updates:

```
DSP0038I RECON INCONSISTENCY RECON HEADER RECORD NOT FOUND
```

When message DSP0038I is received, the defined monitor command first has to be issued to analyze the actual situation and to evaluate the health status before it can be updated. To achieve this, this message has to be defined as a status message in the MESSAGES/USER DATA policy item of the monitor resource with the status `Check`.

Relationships

To propagate the health status of the monitor resource to the health status of the application, a HASMONITOR relationship has to be defined between the IMS control region and the monitor resource.

The monitor resource is kept active only during the UP time of the IMS application group with an additional HASPASSIVEPARENT and a FORCEDOWN/ WhenObservedDown relationship between the monitor resource and the IMS application group. This ensures that the monitor resource can rely on all required functions.

Monitoring of VTAM ACB

IMS automation allows status checking of the VTAM Access Method Control Block (ACB) and the enablement of logons. The IMS command DISPLAY ACTIVE DC is issued regularly after each monitor interval and the response to this command is analyzed.

This monitor function is only useful for an IMS control region, not for a DB control region.

The DC monitoring function is optional, that is, some definitions are necessary to get it up and running.

The implementation of the DC monitoring function includes the following:

- A monitor resource to monitor the status of the VTAM ACB and the enablement of logons
- A status message for passive DC monitoring

IMS DC Monitor Resource

The IMS DC monitor resource has to be defined in the MTR policy object with the following characteristics:

Field Name	**Field Value**
Monitored Object	DC
Monitored Jobname	*jobname* of the IMS control region
Monitor command	INGRMIDC
Monitoring Interval	*hh:mm*

The monitor command is executed after each monitoring interval. It issues the following IMS command and analyzes the status of the VTAM ACB and the LOGONS enablement:

```
DISPLAY ACTIVE DC
```

If no monitoring interval is specified, the defined monitor command is only issued once initially after the monitor resource has been started.

Depending on the monitor results, INGRMIDC ends with the following return codes, which are mapped to the relating health status for the monitor resource:

Return Code	Health status	Description
1	BROKEN	Monitor encountered a severe error
2	FAILED	DISPLAY ACTIVE DC failed
3	NORMAL	VTAM ACB is OPEN and LOGONS enabled
4	WARNING	LOGONS are not enabled

Status Message

Apart from active monitoring with the INGRMIDC monitor command after each monitoring interval, passive monitoring is used to update the health status of the monitor resource. The following IMS message is used to trigger health status update:

```
DFS2111I VTAM ACB CLOSED.
```

When message DFS2111I is received, the health status should be set to WARNING. To achieve this, this message has to be defined as a status message in the MESSAGES/USER DATA policy item of the monitor resource with the status `WARNING`.

Relationships

To propagate the health status of the monitor resource to the health status of the application, you have to define a HASMONITOR relationship between the IMS control region and the monitor resource.

The monitor resource is kept active only during the UP time of the IMS application with an additional HASPASSIVEPARENT and a FORCEDOWN/ WhenObservedDown relationship between the monitor resource and the IMS control region. This ensures that the monitor resource can rely on all required functions.

Monitoring of IMS DB2 Connections

IMS automation allows the monitoring of IMS DB2 connections. The IMS command DISPLAY SUBSYS is issued regularly after each monitor interval and the response to this command is analyzed concerning the status of the connection to a DB2 subsystem.

The IMS DB2 connection monitoring function is optional, that is, definitions are necessary to enable it.

The implementation of the IMS DB2 connection monitoring function includes the following:

- A monitor resource for each connection to be monitored between an IMS control region and a DB2 subsystem.
- Status messages
- Relationships

IMS DB2 Connection Monitor Resource

The IMS DB2 connection monitor resource has to be defined in the MTR policy object with the following characteristics:

Field Name	Field Value
Monitored Object	MVS subsystem ID, *db2_id*, of the DB2 subsystem
Monitored Jobname	*jobname* of the IMS control region
Monitor command	INGRMIDB
Monitoring Interval	*hh:mm*

The specified monitor command INGRMIDB is executed after each monitoring interval. It issues the DISPLAY SUBSYS command for the specified DB2 subsystem and analyzes the status of the connection between both subsystems in the response to this command.

Depending on the monitor results, INGRMIDB ends with the following return codes, which are mapped to the related health status for the monitor resource:

Return Code	Health status	Description
1	BROKEN	No DB2 connection is defined in IMS
2	FAILED	DISPLAY SUBSYS *db2_id* command failed
3	NORMAL	IMS is connected to DB2
4	WARNING	The IMS connection to DB2 is pending
5	MINOR	IMS is not connected to DB2
6	CRITICAL	No DB2 connection is defined in IMS, but recovery is outstanding

Status Messages

Apart from active monitoring by issuing the INGRMIDB monitor command after each monitoring interval, the IMS DB2 monitoring function of SA z/OS also performs passive monitoring by updating the health status whenever it receives one of the following status messages:

```
DSNM001I IMS/TM xxxx CONNECTED TO SUBSYSTEM yyyy
DSNM002I IMS/TM xxxx DISCONNECTED FROM SUBSYSTEM yyyy RC=rc
DSNM003I IMS/TM xxxx FAILED TO CONNECT TO SUBSYSTEM yyyy RC=rc
```

These messages indicate whether the attempt to connect the IMS control region *xxxx* to the DB2 subsystem *yyyy* was successful. In response to these messages, the health status is set to NORMAL if it was successful and MINOR if it was not.

Because these messages are issued to the IMS/TM master terminal operator, you have to define them to the IMS message exit in order that they be issued as WTOs to the console. For details, see “Using the IMS Automation Message Exit” on page 46. SA z/OS provides statements in the NetView automation table that is built that ensure an update of the monitor's health status.

Relationships

To propagate the health status for the monitor resource to the health status of the application, you must define a HASMONITOR relationship between the IMS control region and the monitor resource.

The monitor resource is kept active only during the UP time of the IMS application group and the DB2 application group. This is achieved with an additional HASPASSIVEPARENT and a FORCEDOWN/WhenObservedDown relationship between the monitor resource and both application groups. This ensures that the monitor resource can rely on all required functions.

Chapter 10. MESSAGES/USER DATA Entries for IMS Automation

Because IMS automation is integrated into SA z/OS, you must enter any information for IMS automation in the policy database via the customization dialog. In most cases, the customization dialog itself restricts you to the format that this information must be entered in. There are, however, a number of IMS-specific automation parameters that must or can only be specified as entries in the MESSAGES/USER DATA policy item of the appropriate application policy object. In these cases, the customization panels provide no information about the keywords and the format of their parameters. For further details about the MESSAGES/USER DATA policy item, see *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

This chapter contains detailed descriptions of the automation entries that you have to define for IMS-specific keywords. However, a general understanding of the MESSAGES/USER DATA policy item is assumed.

IMS-Specific MESSAGES/USER DATA Keywords

Most of the following keywords must be specified in the MESSAGES/USER DATA item of the respective *control* region and apply to DB control regions as well as to IMS control regions. They need to be entered in the **Message id** field on the Message Processing panel.

ABCODEPROG: Respond to BMP Region Abends

Use this keyword to define actions to be taken for program abends of program-driven batch message processing (BMP) regions. Only abends for program-driven BMPs use this keyword. Transaction driven BMPs use the ABCODETRAN keyword to determine recovery actions.

Use code processing on the Message Processing panel:

Code 1	Code 2	Code 3	Value Returned
*	*acode*	*progid*	`INCLUDE` or `EXCLUDE`

Note: The ABCODEPROG keyword may be followed with a dot and a program ID *progid*.

Keyword and Parameter Definitions

ABCODEPROG[.*progid*]
You can add the name of a program as a suffix to the keyword. In this case the specifications of the CODE attribute(s) will only apply to this program.

CODE
Use this keyword to define which abends should be included or excluded from recovery.

***** An asterisk as the first positional parameter is required for compatibility with the ABCODETRAN entry. Code an asterisk as shown.

acode The abend code. An asterisk (*) can be used for generic specifications. System abend codes should be prefixed with an S, such as S0C1.

progid
The program name.

INCLUDE|EXCLUDE
Indicates whether to initiate a recovery sequence for this program and abend code combination. Use INCLUDE to initiate a recovery and EXCLUDE if you do not want a recovery initiated. INCLUDE is the default.

Comments and Usage Notes

If the ABCODEPROG entry is omitted, no recovery takes place and a warning message is issued.

The program name can either be specified as **ABCODEPROG.***progid* or as the first value of the CODE attribute. Use **ABCODEPROG.***progid* when you want all of the specifications to apply to one specific program. Use the CODE attribute when you want to code several transactions.

Example of Usage

In the following example, recovery is not attempted for program SAMPLE4 when the abend code is U0778. In all other cases, recovery takes place.

Code 1	Code 2	Code 3	Value Returned
*	U0778	SAMPLE4	EXCLUDE
*	*	*	INCLUDE

ABCODETRAN: Transaction Abend Recovery

Use this keyword to include transaction abend codes in recovery or exclude them from recovery. This includes both transaction-driven message processing regions (MP) and transaction-driven batch message processing (BMP) regions.

Use code processing on the Message Processing panel:

Code 1	Code 2	Code 3	Value Returned
tran	*abend*	*pgm*	INCLUDE or EXCLUDE
*	U3033	*	INCLUDE

Note: The ABCODETRAN keyword may be followed with a dot and a transaction ID *tran*.

Keyword and Parameter Definitions

ABCODETRAN[.*tran*]
You can add the name of a transaction as a suffix to the keyword. In this case the specifications of the CODE attribute(s) will only apply to this transaction.

CODE
Defines which abends are recoverable, as shown in the following descriptions:

tran The transaction ID.

abend The abend code.

pgm The program that abended.

INCLUDE|EXCLUDE
Indicates whether to initiate a recovery for this transaction, abend code, and program. Use INCLUDE to initiate a recovery and EXCLUDE if you do not want a recovery initiated.

Comments and Usage Notes

1. The transaction name is either specified as ABCODETRAN.*tran* or as the first value of the CODE attribute. Use ABCODETRAN.*tran* when you want all of the specifications to apply to one specific transaction. Use the CODE attribute when you want to code several transactions.

Example of Usage

In the following example, recovery takes place for transaction PAYR on IMS10AA if the abend code is U903 or U3033. No recovery takes place for any other transaction or abend code.

Code 1	Code 2	Code 3	Value Returned
*	U3033	*	INCLUDE
*	U907	*	INCLUDE
*	*	*	EXCLUDE

DFS554A: Respond to Program Abend

This keyword is used to define commands or replies to be issued in response to the application program abend message DFS554A.

If the DFS554A message indicates that a transaction has stopped, the commands or replies that have been defined for the selection name TRAN are issued to restart the transaction.

Similarly, if the DFS554A message indicates that a program has stopped, the commands or replies that have been defined for the selection name PROG are issued to restart the program.

Variables

Apart from the task global variables that are provided by AOCQRY, you can use the following variables when specifying the command or reply. The variables are replaced by their corresponding value during run time.

Variable	Value
&EHKVAR1	Transaction name
&EHKVAR2	Program name
&EHKVAR3	Job identifier
&EHKVAR4	Region identifier
&EHKVAR5	User abend code
&EHKVAR6	System abend code

Example of Usage

Ps/Select	AutoFn/*	Command Text
PROG		MVS &SUBSSUBIDSTA PGM &EHKVAR2
TRAN		MVS &SUBSSUBIDSTA TRAN &EHKVAR1

IMSINFO: Display Information

These commands are issued when the INGIMS REQ=INFO or INGIMS REQ=XINFO command is used to display the state of the selected IMS Control or DBCTL region. The commands are issued via the IMS subsystem ID on an MVS EMCS console and the resulting messages are either displayed on the INGIMS panel or written to the user's NetView console.

For further information about the INFO request, see the description of the INGIMS command in *IBM Tivoli System Automation for z/OS Operator's Commands*.

Use user processing on the Message Processing panel:

Keyword	Data
IMSCMD	('*description*','*IMS_command*')

Keyword and Parameter Definitions

description
This is text that is placed before the output of *IMS_command*. It can be used to identify the command output in the output stream. The description can be any string, but must be enclosed in quotation marks.

IMS_command
This is the command that is to be executed. It is appended to the IMS subsystem ID and issued as an MVS command. The output is collected and displayed. The command can be any valid IMS Type 1 or Type 2 command. If you user a Type 2 command, be sure to also use INGIMS REQ=XINFO. You must enclose the command in quotation marks.

You can code multiple IMS commands, separated by a comma, in order to group results under a common description.

Comments and Usage Notes

This policy is required for the correct operation of the INFO request of the INGIMS command, and also for the PF10 function key of the DISPINFO panel.

Example of Usage

Keyword	Data
`IMSCMD`	`('CQS DATA','DIS CQS')`
`IMSCMD`	`('Comms','DIS A DC')`

TCO: Allocate data set with Commands for Time-Driven Procedures

This entry allows you to specify the name of the data set holding the TCO members and commands to be issued to initiate, change, start, or stop time-driven procedures for any IMS operation.

Use reply processing on the Message Processing panel:

Pass/Selection	Retry Count	Reply Text
`INIT`		`DFSTCF LOAD DFSTCF  .`

Pass/Selection	Retry Count	Reply Text
`SPEC`		`DFSTCF LOAD &EHKVAR1 .`
`START`		`/START LTERM DFSTCF1 .`
`STOP`		`/PSTOP LTERM DFSTCFI .`

Keyword and Parameter Definitions

INIT
This entry specifies the IMS command required to start the initial time driven procedure DFSTCF.

SPEC
This entry specifies the IMS command required to change from the current TCO script to a script name entered from the operator interface.

START
This entry specifies the IMS command required to start TCO.

STOP
This entry specifies the IMS command required to stop TCO.

Comments and Usage Notes

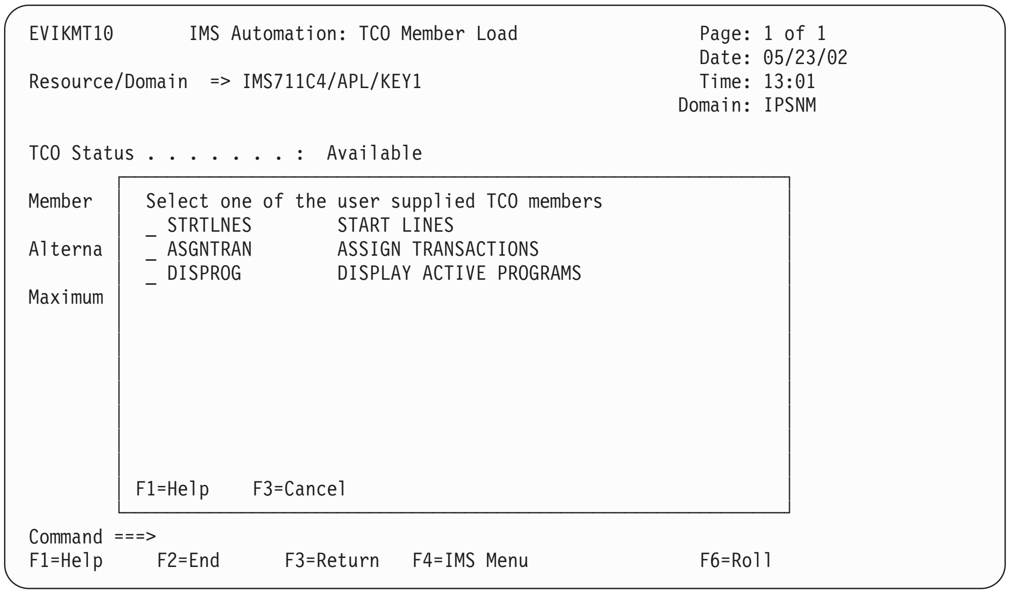
1. Use the user processing option on the Message processing panel to define TCO data set name. The specification is optional.

Keyword	Data
DSN	*data set name*

2. For more information on TCO, refer to the *IMS Operations Guide*.
3. &EHKVAR1 is the name entered from the IMS automation operator interface TCO function.

TCOMEMBERS: Define TCO Members

This keyword serves to create a list of members that appear in a pop-up IMS Automation TCO Member Load panel. The panel is shown in Figure 1 on page 66.



```
EVIKMT10       IMS Automation: TCO Member Load                Page: 1 of 1
                                                              Date: 05/23/02
Resource/Domain  => IMS711C4/APL/KEY1                         Time: 13:01
                                                            Domain: IPSNM

TCO Status . . . . . . . :  Available
          ┌───────────────────────────────────────────────────────────────┐
Member    │ Select one of the user supplied TCO members                   │
          │ _ STRTLNES        START LINES                                 │
Alterna   │ _ ASGNTRAN        ASSIGN TRANSACTIONS                         │
          │ _ DISPROG         DISPLAY ACTIVE PROGRAMS                     │
Maximum   │                                                               │
          │                                                               │
          │                                                               │
          │                                                               │
          │                                                               │
          │                                                               │
          │                                                               │
          │F1=Help    F3=Cancel                                           │
          └───────────────────────────────────────────────────────────────┘
Command ===>
F1=Help     F2=End      F3=Return   F4=IMS Menu               F6=Roll
```

Figure 1. Example IMS Automation TCO Member Load Pop-up Panel

Use user processing on the Message Processing panel:

Keyword	Data
NAME	(*membername*,'*comment*')

Format

```
►►──TCOMEMBERS──NAME=(membername,'comment')──────────────────────────►◄
```

Keyword and Parameter Definitions

membername
: This is the 8-character name of the member previously defined in the IMS TCO member library. The library is associated with the DFSTCF DD statement in the IMS start up JCL.

comment
: This is a comment, up to 32 characters long.

Comments and Usage Notes

For more information on TCO, refer to "Working with TCO Functions" on page 76.

Example of Usage

Keyword	Data
NAME	(STRTLNES,'START LINES)
NAME	(ASGNTRAN,'ASSIGN TRANSACTIONS')

Chapter 11. INGIMS Operator Command

INGIMS is IMS-related SA z/OS command that issues IMS console commands and that can be invoked by operators or via PIPEs.

INGIMS: Issue List of Defined Transactions and View the Output

The INGIMS command lets you:

- Issue any console-enabled IMS type 1 command
- Issue any IMS type 2 command if an IMSPlex name is provided
- Broadcast messages to all or selected IMS users
- Issue a list of defined transactions and view the output
- Display the output of IMS transactions in full-screen or pipeable line mode

For a detailed description of the INGIMS command, see *IBM Tivoli System Automation for z/OS Operator's Commands*.

Chapter 12. Monitor Commands

The following routines should be used as a MONITOR command when defining monitor resources for IMS Automation.

INGRMIDB Routine for the Monitoring of IMS DB2 Connections

This routine should be used as the monitor command for a monitor resource. It is issued after each monitoring interval to check whether IMS is connected to DB2. To achieve this, the IMS command DISPLAY SUBSYS is issued for the DB2 subsystem to query the status of the IMS DB2 connection.

The health status is considered to be NORMAL when the status of the monitored IMS DB2 connection is Connected. A degraded health status is returned in all other cases.

Format

```
►►──INGRMIDB──────────────────────────────────────────────────────────►◄
```

Restrictions

The INGRMIDB routine is assumed to be defined as the monitor command for a monitor resource with the following restrictions:

- The MVS subsystem ID of the DB2 subsystem, *db2_id*, is specified as the monitored object.
- A job name of a subsystem of category IMS with subcategory CTL is specified for the monitor resource.
- The monitor resource is related to a IMS subsystem.

Return Codes

Depending on the monitor results, INGRMIDB ends with the following return codes, which are mapped to the related health status for the monitor resource:

Return Code	Health status	Description
1	BROKEN	No DB2 connection is defined in IMS
2	FAILED	DISPLAY SUBSYS *db2_id* command failed
3	NORMAL	IMS is connected to DB2
4	WARNING	The IMS connection to DB2 is pending
5	MINOR	IMS is not connected to DB2
6	CRITICAL	No DB2 connection is defined in IMS, but recovery is outstanding

INGRMIOL Routine for OLDS Monitoring

This routine should be used as the monitor command for a monitor resource. It is issued after each monitoring interval to check the status of the online log data sets (OLDS) of an IMS control region and to take recovery actions if needed. To achieve this, the IMS command DISPLAY OLDS is issued to analyze the status of the listed OLDS data sets. If needed, OLDS data sets are started or stopped. Any OLDS data sets that have been defined as spare OLDS data sets are started last and stopped first.

The health status of the monitored OLDS data sets is considered to be NORMAL when the number of available OLDS data sets is not less than a given minimum number, and the AUTOMATIC ARCHIVE is switched on. In all other cases a degraded health status is returned.

Format

```
►►──INGRMIOL──────────────────────────────────────────────────────────►◄
```

Restrictions

- The monitor resource must be for an IMS control region that is defined as an IMS subsystem of subcategory CTL.
- OLDS is specified as the monitored object.
- Automation actions in INGRMIOL are only taken if the recovery automation flag for minor resource OLDS is on.

Comments and Usage Notes

The automation settings for OLDS monitoring are taken from user data for the special message ID OLDS, which is defined in the MESSAGES/USER DATA policy item of the IMS control region with the following keyword-data pairs:

Keyword	Data	Description
MINIMUM	*nn*	The minimum number of OLDS data sets that must be in (AVAILABLE + IN USE) status. Half of the number of listed OLDS data sets in the response of the DISPLAY OLDS command is assumed as the default value.
SPARES	(*nn*,*nn*...)	These specified spare OLDS data sets are started last or stopped first when OLDS data sets have to be started or stopped in order to have the minimum number of OLDS data sets in status AVAILABLE.
BACKOUT	*nn*	The maximum number of OLDS that are allowed to be in a BACKOUT status.

All of these settings are optional.

Return Codes

Depending on the monitor results, INGRMIOL ends with the following return codes, which are mapped to the related health status for the monitor resource:

Return Code	Health status	Description
1	BROKEN	The monitor encountered a severe error.
2	FAILED	DISPLAY OLDS failed.
3	NORMAL	No problem found by OLDS monitoring.
4	WARNING	One of the following occurred: • Needed to start spare OLDS to have the minimum in AVAILABLE status • AUTOMATIC ARCHIVE is off
5	MINOR	Could not start enough spare OLDS to have the minimum in AVAILABLE status.
6	CRITICAL	The number of OLDS in BACKOUT status exceeds maximum limit.

INGRMIRE Routine for RECON Monitoring

This routine should be used as the monitor command for a monitor resource. It is issued after each monitoring interval to check the status of the recovery control data sets (RECON) of an IMS control region. To achieve this, the IMS command RMLIST DBRC=’RECON STATUS’ is issued to analyze the status of the listed RECON data sets in the response to this command.

The health status of the monitored RECON data sets is considered to be NORMAL when three RECON data sets are found in the status COPY1, COPY2 and SPARE. A degraded health status is returned in all other cases.

Format

```
►►──INGRMIRE────────────────────────────────────────────────────────────►◄
```

Restrictions

- The monitor resource must be for an IMS control region that is defined as an IMS subsystem of subcategory CTL.
- RECON is specified as the monitored object.

Return Codes

Depending on the monitor results, INGRMIRE ends with the following return codes, which are mapped to the related health status for the monitor resource:

Return Code	Health status	Description
1	BROKEN	The monitor encountered a severe error.
2	FAILED	RMLIST DBRC=’RECON STATUS’ failed.
3	NORMAL	No problem found by RECON monitoring.
4	WARNING	No COPY2 found for RECON.
5	MINOR	No SPARE found for RECON.
6	CRITICAL	Neither COPY2 nor SPARE found, and possibly no COPY1 found for RECON.

INGRMIDC Routine for DC Monitoring

This routine should be used as the monitor command for a monitor resource. It is issued after each monitoring interval to check whether the VTAM ACB of an IMS control region is OPEN and LOGONS are ENABLED. To achieve this, the response to the IMS command DISPLAY ACTIVE DC is analyzed.

The health status is considered to be NORMAL when the VTAM ACB of the IMS control region is OPEN and LOGONS are ENABLED. A degraded health status is returned in all other cases.

Format

```
►►──INGRMIDC──────────────────────────────────────────────────────────►◄
```

Restrictions

- The monitor resource must be for an IMS control region that is defined as an IMS subsystem of subcategory CTL. It should not be used for a DB control region.
- DC is specified as the monitored object.

Return Codes

Depending on the monitor results, INGRMIDC ends with the following return codes, which are mapped to the related health status for the monitor resource.

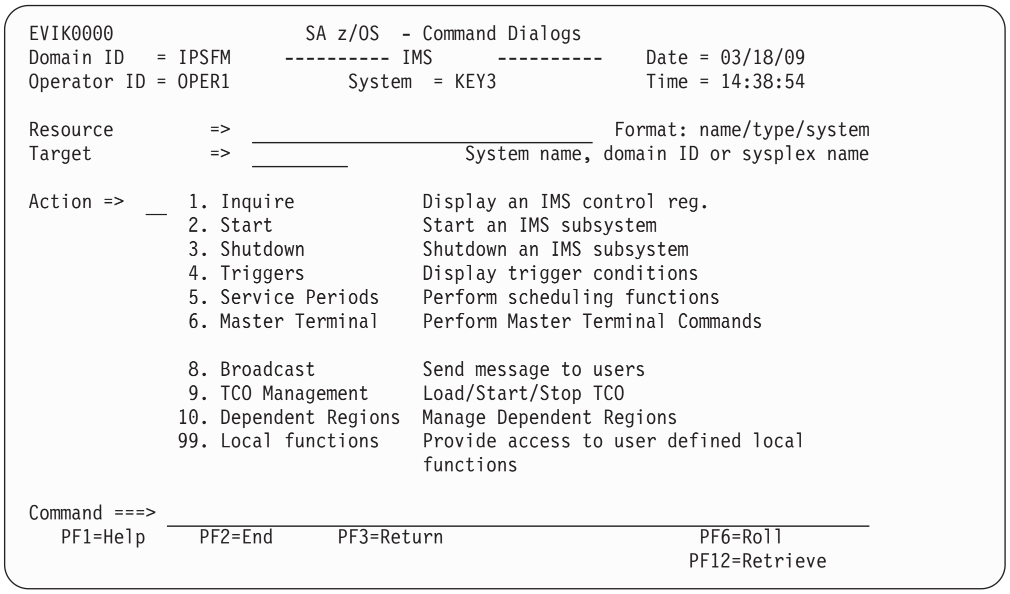
Return Code	Health status	Description
1	BROKEN	The monitor encountered a severe error.
2	FAILED	DISPLAY ACTIVE DC failed.
3	NORMAL	VTAM ACB is OPEN and LOGONS enabled.
4	WARNING	LOGONS are not enabled.

Chapter 13. Using IMS Automation

This chapter explains how to use the IMS Automation panels and to work with subsystems. It is assumed that you have used and are familiar with the SA z/OS operator interface. This chapter describes characteristics that are unique to IMS Automation.

Using the Main Menu

The main menu panel lists all of the tasks available with the operator interface, as shown in Figure 2.



```
EVIK0000                  SA z/OS  - Command Dialogs
Domain ID   = IPSFM     ---------- IMS      ----------    Date = 03/18/09
Operator ID = OPER1           System  = KEY3              Time = 14:38:54

Resource         =>  ________________________________  Format: name/type/system
Target           =>  _________             System name, domain ID or sysplex name

Action =>  __   1. Inquire           Display an IMS control reg.
                2. Start             Start an IMS subsystem
                3. Shutdown          Shutdown an IMS subsystem
                4. Triggers          Display trigger conditions
                5. Service Periods   Perform scheduling functions
                6. Master Terminal   Perform Master Terminal Commands

                8. Broadcast         Send message to users
                9. TCO Management    Load/Start/Stop TCO
               10. Dependent Regions Manage Dependent Regions
               99. Local functions   Provide access to user defined local
                                     functions

Command ===> __________________________________________________________________
   PF1=Help     PF2=End      PF3=Return                         PF6=Roll
                                                               PF12=Retrieve
```

Figure 2. IMS Automation Main Menu

You can specify a resource using the following fields:

Resource
To define one or more IMS Subsystems to be processed, type in the name of a resource. This should be in the format name/type/system. This name will be used for all functions until a new one is entered on one of the Major Option panels or selected on one of the List Selection panels.

To display a list of all known IMS resources type a question mark (?) for the resource. You can then select a subsystem from this list (refer to "Selecting and Viewing Subsystems" on page 75). To obtain this list for a sysplex that is not the current one, specify the corresponding sysplex in the **Target** field before you press Enter.

Note: Many of the menu options are not valid for remote sysplexes.

Target This field allows you to specify the system name, the domain ID or the sysplex name where you would like the required IMS function to be performed. To limit the resource list to a specific system, use the */APL/system notation.

| If you specify a sysplex name and then enter a question mark in the
| **Resource** field you call a list of all the IMS control regions in the specified
| sysplex. For the current sysplex, you need not enter the sysplex name.

You can then select one of the following options by entering it in the **Action** field. The options that are marked with an asterisk (*) are not valid for remote sysplexes. For more details about the SA z/OS command that is called for each option, see *IBM Tivoli System Automation for z/OS Operator's Commands*.

1 (Inquire)*
: Use this option to retrieve detailed information for an IMS resource. See "Getting Detailed Status" on page 76.

2 (Start)
: Use this option to initiate the startup process of a resource. This option calls the INGREQ command.

3 (Shutdown)
: Use this option to initiate the shutdown process of a resource. This option calls the INGREQ command.

4 (Triggers)
: Use this option to display the triggers associated with a resource. This option calls the DISPTRG command.

5 (Service Periods)
: Use this option to display or override the schedule that is associated with a resource. This option calls the INGSCHED command.

6 (Master Terminal)*
: Use this option to issue a command to a specific subsystem. This option calls the INGIMS REQ=CMD command.

8 (Broadcast)*
: Use this option to send a message to a specific subsystem. This option calls the INGIMS REQ=BROADCAST command.

9 (TCO Management)*
: Use this option to manage the Time Controlled Operations (TCO) functions of a specific subsystem. See "Working with TCO Functions" on page 76 for details.

10 (Dependent Regions)*
: Use this option to display information about IMS dependent regions that you can issue commands against. See "Displaying IMS Dependent Region Information" on page 77 for details.

99 (Local functions)*
: IMS Automation allows your system programmer to add functions to this operator interface. If functions have been added at your installation, select this option to view a menu of them.

Note: When option 99 is selected from the IMS Automation main menu, the EVIEU000 module is called. This module is not part of SA z/OS, but should be provided by you as a user-defined application.

Important:
Option 99 is only valid for the local sysplex. You cannot access a remote sysplex with any of these functions.

Selecting and Viewing Subsystems

This section describes how to select a resource from a list of available resources and how to display detailed information about a subsystem.

Selecting a Subsystem

You can specify the resource you want to work with by simply entering its name in the **Subsystem** or **Resource** field of the respective panel. On the IMS-specific panels you can call up a list of the available IMS control regions by entering a question mark at position one of the **Subsystem** or **Resource** field. On the panels that belong to basic SA z/OS commands (INGREQ, INGSCHED, DISPTRG), you can use an asterisk (*) as a wildcard.

Figure 3 shows a list of IMS control regions generated with the question mark function.

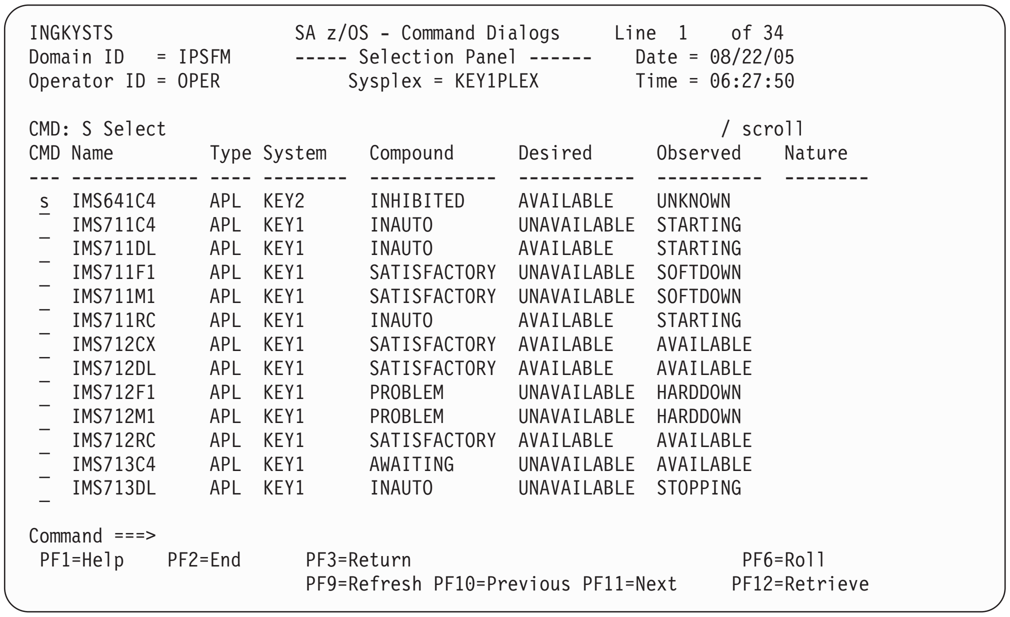

```
INGKYSTS                 SA z/OS - Command Dialogs     Line  1    of 34
Domain ID   = IPSFM      ----- Selection Panel ------    Date = 08/22/05
Operator ID = OPER            Sysplex = KEY1PLEX         Time = 06:27:50

CMD: S Select                                                / scroll
CMD Name         Type System    Compound      Desired      Observed    Nature
--- ------------ ---- --------  ------------  -----------  ----------  --------
 s  IMS641C4     APL  KEY2      INHIBITED     AVAILABLE    UNKNOWN
 _  IMS711C4     APL  KEY1      INAUTO        UNAVAILABLE  STARTING
 _  IMS711DL     APL  KEY1      INAUTO        AVAILABLE    STARTING
 _  IMS711F1     APL  KEY1      SATISFACTORY  UNAVAILABLE  SOFTDOWN
 _  IMS711M1     APL  KEY1      SATISFACTORY  UNAVAILABLE  SOFTDOWN
 _  IMS711RC     APL  KEY1      INAUTO        AVAILABLE    STARTING
 _  IMS712CX     APL  KEY1      SATISFACTORY  AVAILABLE    AVAILABLE
 _  IMS712DL     APL  KEY1      SATISFACTORY  AVAILABLE    AVAILABLE
 _  IMS712F1     APL  KEY1      PROBLEM       UNAVAILABLE  HARDDOWN
 _  IMS712M1     APL  KEY1      PROBLEM       UNAVAILABLE  HARDDOWN
 _  IMS712RC     APL  KEY1      SATISFACTORY  AVAILABLE    AVAILABLE
 _  IMS713C4     APL  KEY1      AWAITING      UNAVAILABLE  AVAILABLE
 _  IMS713DL     APL  KEY1      INAUTO        UNAVAILABLE  STOPPING

Command ===>
 PF1=Help    PF2=End      PF3=Return                              PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next     PF12=Retrieve
```

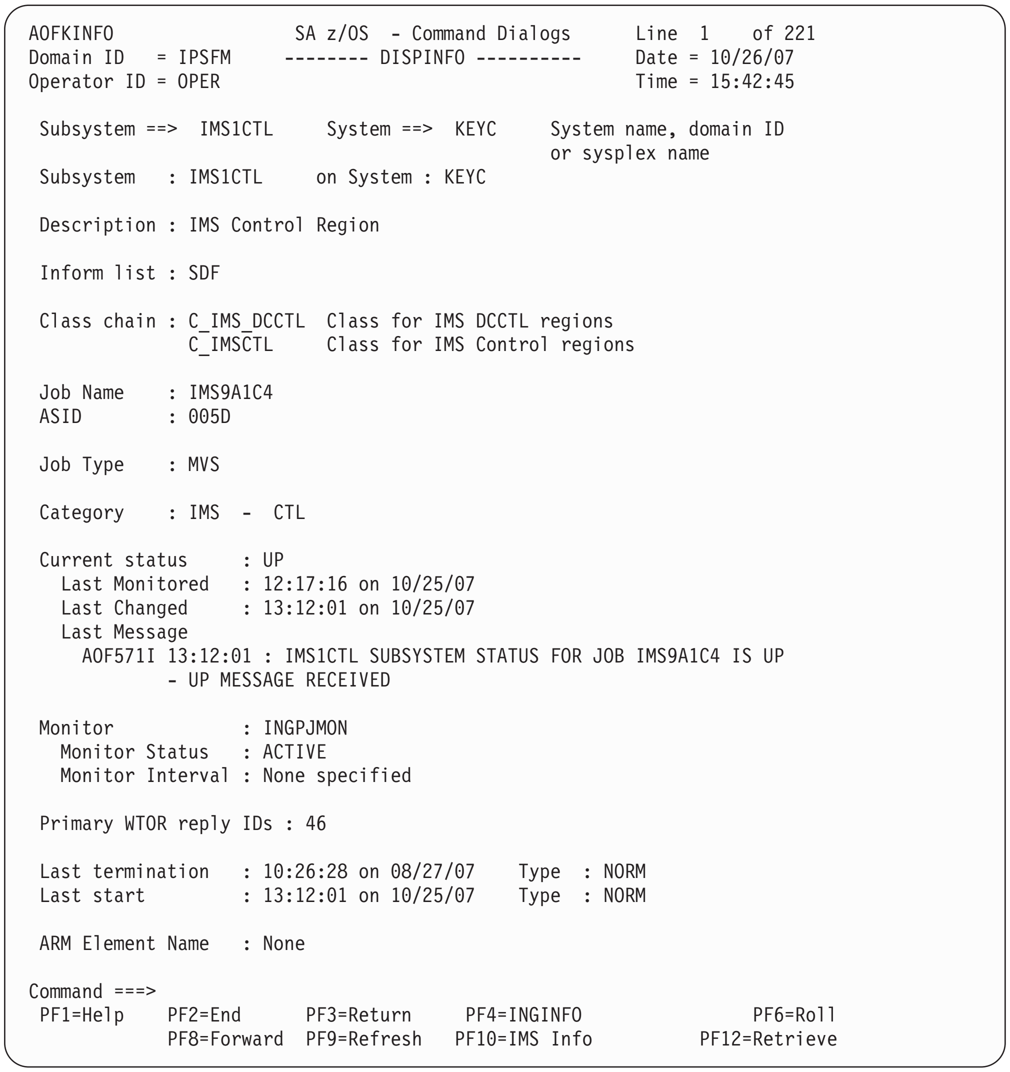
Figure 3. Selection Panel for IMS Resources

The list contains all IMS subsystems of the KEY1PLEX sysplex (see line 3 of the panel) that are defined to SA z/OS. You can use it not only to select a subsystem (by entering s in the **CMD** column), but also to get an overview of the sysplex. Columns 5 through 7, for example, contain status information for the subsystems. For more details on the different status types, see *IBM Tivoli System Automation for z/OS User's Guide*.

You can scroll horizontally through the list to the right by pressing PF11, and to the left by pressing PF10. The first three columns, which make up the resource name of the subsystem, are fixed.

Getting Detailed Status

To view the status of a specified subsystem in detail, select option **1** on the main menu panel and press Enter. This displays the panel shown in Figure 4



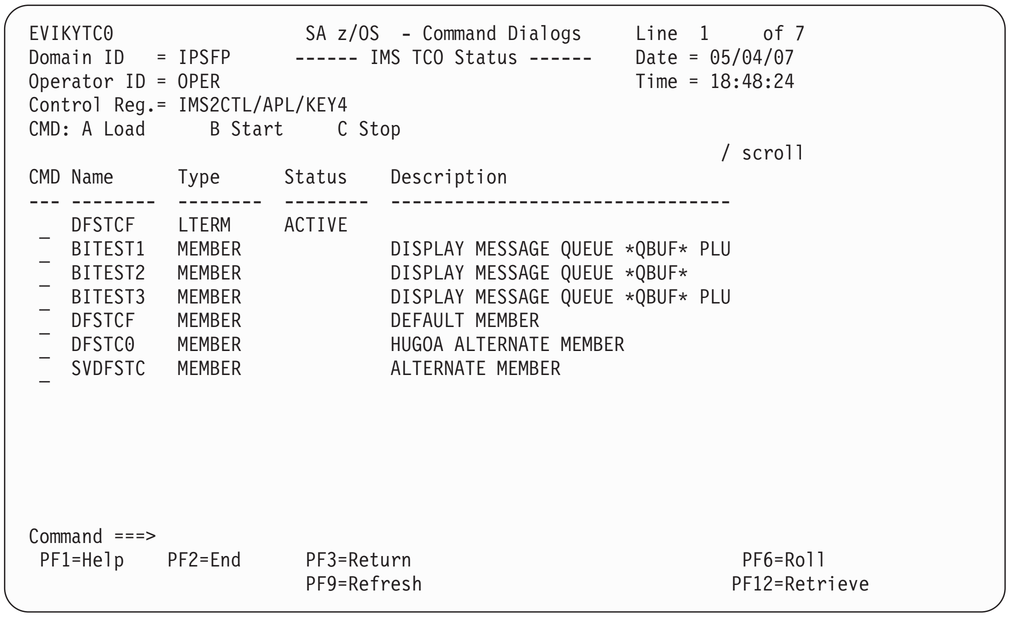
```
AOFKINFO                 SA z/OS  - Command Dialogs      Line  1    of 221
Domain ID   = IPSFM     -------- DISPINFO ----------     Date = 10/26/07
Operator ID = OPER                                       Time = 15:42:45

 Subsystem ==>  IMS1CTL     System ==>  KEYC     System name, domain ID
                                                 or sysplex name
 Subsystem   : IMS1CTL     on System : KEYC

 Description : IMS Control Region

 Inform list : SDF

 Class chain : C_IMS_DCCTL  Class for IMS DCCTL regions
               C_IMSCTL     Class for IMS Control regions

 Job Name    : IMS9A1C4
 ASID        : 005D

 Job Type    : MVS

 Category    : IMS  -  CTL

 Current status     : UP
   Last Monitored   : 12:17:16 on 10/25/07
   Last Changed     : 13:12:01 on 10/25/07
   Last Message
     AOF571I 13:12:01 : IMS1CTL SUBSYSTEM STATUS FOR JOB IMS9A1C4 IS UP
             - UP MESSAGE RECEIVED

 Monitor            : INGPJMON
   Monitor Status   : ACTIVE
   Monitor Interval : None specified

 Primary WTOR reply IDs : 46

 Last termination   : 10:26:28 on 08/27/07    Type  : NORM
 Last start         : 13:12:01 on 10/25/07    Type  : NORM

 ARM Element Name   : None

Command ===>
 PF1=Help    PF2=End      PF3=Return     PF4=INGINFO               PF6=Roll
             PF8=Forward  PF9=Refresh   PF10=IMS Info          PF12=Retrieve
```

Figure 4. Detailed Subsystem Information Panel for an IMS Control Region

Working with TCO Functions

The master terminal functions provide a fullscreen panel interface to perform Time Controlled Operations (TCO) functions. Select option 9, TCO Management, to display the IMS TCO Status panel, as shown in Figure 5 on page 77.

| **Note:** Use the new IMS Automation command INGIMS to achieve the same
| functionality: for example, INGIMS *resource* REQ=TCO



```
EVIKYTC0                  SA z/OS  - Command Dialogs     Line  1     of 7
Domain ID   = IPSFP      ------ IMS TCO Status ------    Date = 05/04/07
Operator ID = OPER                                       Time = 18:48:24
Control Reg.= IMS2CTL/APL/KEY4
CMD: A Load      B Start     C Stop
                                                                 / scroll
CMD Name      Type      Status    Description
--- --------  --------  --------  --------------------------------
 _  DFSTCF    LTERM     ACTIVE
 _  BITEST1   MEMBER              DISPLAY MESSAGE QUEUE *QBUF* PLU
 _  BITEST2   MEMBER              DISPLAY MESSAGE QUEUE *QBUF*
 _  BITEST3   MEMBER              DISPLAY MESSAGE QUEUE *QBUF* PLU
 _  DFSTCF    MEMBER              DEFAULT MEMBER
 _  DFSTC0    MEMBER              HUGOA ALTERNATE MEMBER
 _  SVDFSTC   MEMBER              ALTERNATE MEMBER






Command ===>
 PF1=Help    PF2=End      PF3=Return                              PF6=Roll
                          PF9=Refresh                            PF12=Retrieve
```

Figure 5. IMS TCO Status Panel

The Time Controlled Operations (TCO) interface lets you issue TCO commands from the IMS Automation interface.

From this panel you can:

- Load a specific member
- Start or stop TCO processing for a logical terminal

Load a Specific Member

Enter command code A against an object of type MEMBER to load that member. The member is loaded into TCO via the IMS control region outstanding reply. The reply text that is used is defined in the ACF via the TCO entry and has a selection of SPEC. Typically it is defined as `DFSTCF LOAD &EHKVAR1 .` in the ACF, where the EHKVAR1 variable is set to the member name that the line command is executed against.

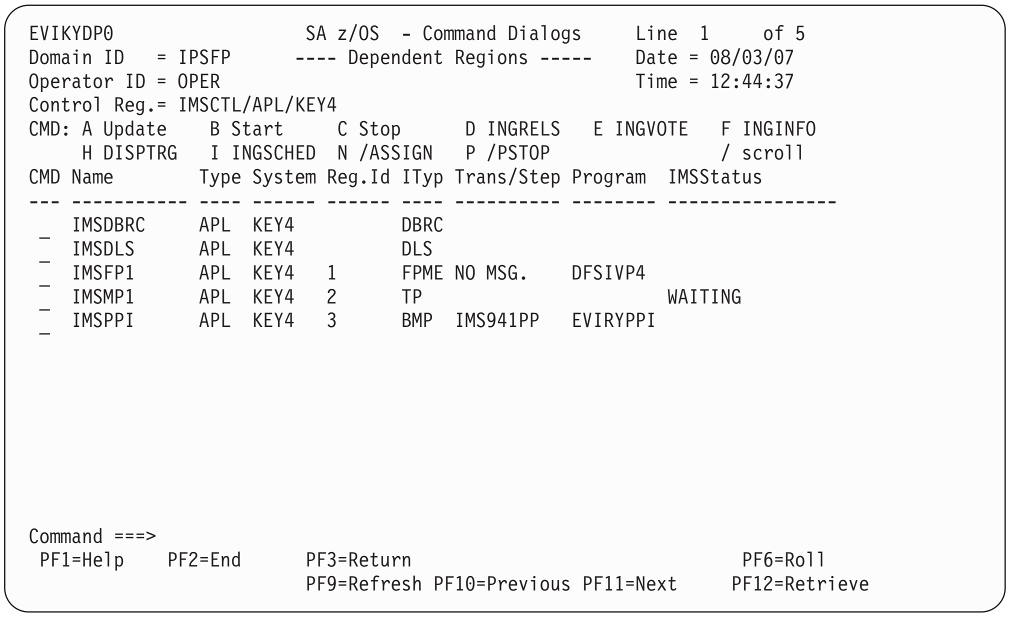
Start or Stop TCO Processing

Enter command code B or C against an object of type LTERM to start or stop TCO processing for that member. This invokes INGIMS with the command of `/START LTERM` *name* or `/STOP LTERM` *name*, where *name* is the name of the logical terminal.

Displaying IMS Dependent Region Information

| Option 10 displays details about the IMS dependent regions that are being
| processed by the specified IMS Control Region, as shown in Figure 6 on page 78.
| All dependent regions that are currently active are displayed. Those that are
| defined to SA z/OS are displayed as SA z/OS resources. The remainder have only
| their job names and IMS information displayed.

| **Note:** Use the new IMS Automation command INGIMS to achieve the same
| functionality: for example, INGIMS *resource* REQ=DEP



```
EVIKYDP0                 SA z/OS  - Command Dialogs     Line  1     of 5
Domain ID   = IPSFP      ---- Dependent Regions -----    Date = 08/03/07
Operator ID = OPER                                       Time = 12:44:37
Control Reg.= IMSCTL/APL/KEY4
CMD: A Update    B Start     C Stop      D INGRELS   E INGVOTE   F INGINFO
     H DISPTRG   I INGSCHED  N /ASSIGN   P /PSTOP               / scroll
CMD Name        Type System Reg.Id ITyp Trans/Step Program  IMSStatus
--- ----------- ---- ------ ------ ---- ---------- -------- ----------------
 _  IMSDBRC     APL  KEY4          DBRC
 _  IMSDLS      APL  KEY4          DLS
 _  IMSFP1      APL  KEY4   1      FPME NO MSG.    DFSIVP4
 _  IMSMP1      APL  KEY4   2      TP                       WAITING
 _  IMSPPI      APL  KEY4   3      BMP  IMS941PP   EVIRYPPI







Command ===>
 PF1=Help    PF2=End      PF3=Return                              PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next     PF12=Retrieve
```

Figure 6. IMS Dependent Regions Panel

For each resource the following information is shown:

Name The name of the resource. This is the job name from the /DIS ACTIVE display if the dependent region is not defined to SA z/OS.

Type The type of the resource. This is blank if the dependent region is not defined to SA z/OS.

System
The name of the system where the resource resides. This is blank if the dependent region is not defined to SA z/OS.

The following fields are displayed on a series of screens that you can move through with the PF10 or PF11 key:

Reg.Id The IMS dependent region number.

ITyp The second type column is the IMS dependent region type.

Trans/Step
Transaction code being processed by region, or NONE, if there are no regions of that type.

Program
Name of the program processing in the region.

IMSStatus
Status of the region, which can be one of the following:

Status	Meaning
UNCERTAIN	The IMS and SA statuses do not match. In particular, IMS does not know about the job name represented by the SA Resource and yet SA sees the resource as Available. This is a possible problem with the IMS dependent region.
ACTIVE-DBCMD	A /DBD or a /DBR command is in progress and waiting for the region to terminate before the /DBD or /DBR can complete.
AVAILABLE	The active threads are available. The region is available to schedule an application.
SCHEDULED	The application program is being scheduled.

Status	Meaning
TERMINATING	The application program is being terminated.
UNAVAILABLE	An active DBT thread is unavailable. An application is using the region, even though the application is not currently scheduled. This region is therefore not available to any other application.
WAITING	The MPP region is waiting for work.
WAIT-AOI	An AO application issued a GMSG call with the WAITAOI subfunction specified, but there are no messages for the AO application to retrieve.
WAIT-BLOCKMOVER	An application control block cannot be loaded because the ACB block mover is busy.
WAIT-CMD/ PENDING	A /DBDUMP, /DBRECOVERY, or /START command is in progress.
WAIT-INPUT	The application program is in WAIT-FOR-INPUT (WFI) mode.
WAIT-INTENT	The application program's intent for a database conflicts with the use of the database by a scheduled program.
WAIT-INTENT/ POOL	Indicates that either the application program's intent for a database conflicts with the use of the database by a scheduled program, or a temporary shortage of DMB, PSB, or PSB work area pool space exists.
WAIT-INTENT SCHD	The IMS transaction scheduler detected an application scheduling intent, for example, Load Balancing.
WAIT-I/O PREVEN	A BMP region that accesses a GSAM database cannot schedule until I/O prevention has completed.
WAIT-MESSAGE	The application program is in a pseudo WAIT-FOR-INPUT (WFI) mode. The application is scheduled and is waiting for a message.
WAIT-POOLSPACE	A temporary shortage of DMB, PSB, or PSB work area pool space exists.
WAIT-SWITCHOVER	The alternate system is tracking the active system.
WAIT-SYNCPOINT	The application in the region is now in SYNC POINT.
WAIT-EPCB POOL	A temporary shortage of EPCB pool space exists.
WAIT-RRS PC	The application program has a protected conversation with an OTMA client that is processing a synchronization point (or sync point). Sync point can continue after the OTMA client issues either an SRRCMIT or SRRBACK call. Alternatively, the application program is part of a cascaded family and is processing a sync point. APPC/OTMA SMQ Enablement uses RRS cascaded transaction support to synchronize the backend and frontend system.

IMS Classes
: One of the classes that is associated with the region. The region can have from one to four classes, whose values range from 1 to 999.

Compound
: The compound status of the resource. This is a summary of all statuses of the resource and provides a single value that tells you whether the resource is OK or it has a problem.

Observed
: The current status of the resource as observed by the automation agent.

Desired
: The status that the automation manager is trying to move the resource to. It can be either available or unavailable.

Automation
: The status representing the automation agents automation for the resource.

Startable
Indicates whether it is possible to start the resource if the automation manager is asked to do so at this point in time.

Health
The health status of the resource. Shows N/A if the resource is not connected to a monitor.

Auto Shows the automation flag that is maintained by the automation manager. No automation is done for the resource by the automation manager when the flag is off.

Hold Shows the hold flag that is maintained by the automation manager.

Starttype
The preset start type that will be used the next time the resource is made available (started). This value is set by INGSET and will override any TYPE value that is specified (or defaulted to) in the next INGREQ start request.

Stoptype
The preset stop type that will be used the next time the resource is made unavailable (shut down). This value is set by INGSET and will override any TYPE value specified (or defaulted to) in the next INGREQ stop request. However, a stop type of FORCE, wherever specified, will always be honored.

Trigger
The trigger that is associated with the resource.

| **Category**
| Shows the category of the resource, always IMS for dependent regions.

| **Subtype**
| The subtype of the resource. Available subtypes are: CTL, TP, DBRC, DLS,
| FP, BMP, FDR.

Description
Descriptive information about the resource.

Different colors are used to indicate when a particular status is viewed as abnormal, as follows:

Color	**Meaning**
Blue	The desired status is UNAVAILABLE.
Pink	The compound status is DENIED or INHIBITED.
Red	The compound status is PROBLEM. The observed status is HARDDOWN or PROBLEM.
Yellow	The compound status is DEGRADED. The observed status is not in line with the desired status.

You can use one of the following command codes to invoke another command dialog:

A Update resource settings (SA z/OS resources only).

B Start resource (make available). SA z/OS resources will invoke INGREQ, non-SA z/OS regions will invoke INGIMS with a command of /START REGION *xxxxx*, where *xxxxx* is the job name.

C Stop resource (make unavailable). SA z/OS resources will invoke INGREQ, non-SA z/OS regions will invoke INGIMS with a command of /STOP REGION JOBNAME *xxxxx*, where *xxxxx* is the job name.

D Show resource relationships. (This invokes INGRELS).

E Show resource requests or votes. (This invokes INGVOTE).

F Show resource details. (This invokes INGINFO).

H Show trigger details for resource. (This invokes DISPTRG).

I Show schedules for resource. (This invokes INGSCHED).

N Invoke the /ASSIGN command. Use the assign command to assign additional classes to the region. It invokes INGIMS with the command /ASSIGN CLASS *xx* REGION *nn*, where *nn* is the region number. You can overtype the command to supply the classes.

P Invoke the /PSTOP command. Use the pstop command to stop a transaction. It invokes INGIMS with the command /PSTOP REGION *nn* TRANSACTION *xx*, where *nn* is the region number and *xx* is taken from the **Trans/Step** column. You can overtype the command before executing it.

Note: This action code will not check whether the control region or the dependent region supports the command. You can adjust the command to suit your purpose if the TRANSACTION format of the command is not suitable.

Additionally you can use the slash character (/) to make the selected line the first line of the display.

You can also use the FIND(F), RFIND(RF), and SORT subcommands on this panel.

Chapter 14. IMS Automation Routines

The following IMS automation routines are available:
- EVIECT0X
- EVISTRCT
- EVIEET00
- EVIEI006
- EVISTRMN

EVIECT0X

Purpose

This routine can be used to perform recovery processing for transaction and program abends in response to the application program abend message DFS554A.

The routine performs the following actions:
- Parses all data from the DFS554A
- Checks the appropriate automation flag
- Checks the code definitions for exclusions from recovery for the message types ABCODETRAN or ABCODEPROG
- Performs threshold checking for the triggering DFS554A message
- Captures received message with severity NORMAL.

The minor resource name that is used for automation flag checking and threshold checking is either TRAN.*tran* or PROG.*progid*.

If allowed, the recovery commands or replies for message DFS554A with the selection names PROG or TRAN are issued.

EVIECT0X should be called from the NetView automation table.

Syntax

```
►►──EVIECT0X──────────────────────────────────────────────────►◄
```

EVISTRCT

Purpose

Captures received message (CQS0205E) with severity CRITICAL.

Syntax

```
►►──EVISTRCT──────────────────────────────────────────────────►◄
```

Usage

The EVISTRCT automation routine is intended to respond to the following message:

```
CQS0205E STRUCTURE structurename IS FULL
```

EVIEET00

Purpose

This routine is a command to process IMS TCO automation.

This routine should be invoked from the NetView automation table.

Syntax

```
►►──EVIEET00──────────────────────────────────────────────────────────────►◄
```

Usage

The EVIEET00 automation routine is intended to respond to the following messages:

```
DFS3343E CANNOT PROCESS DFSTCF LOAD COMMAND, REASON=xx
DFS3350E TCO ABNORMALLY TERMINATED, SEE DUMP
DFS335I TCO ABNORMALLY TERMINATED, SYSTEM ABEND, SEE DUMP
DFS3613I xxx TCB INITIALIZATION COMPLETE
```

EVIEI006

Purpose

This routine handles the IMS control region restart errors.

This routine should be invoked from the NetView automation table.

Syntax

```
►►──EVIEI006──────────────────────────────────────────────────────────────►◄
```

Usage

The EVIEI006 automation routine is intended to respond to the following messages:

```
DFS166 CHECKPOINT ID NOT ON LOG RE-ENTER RESTART COMMAND
DFS033I DUPLICATE ENTRY ON SIGNON REQUEST, RESTART ABORTED
DFS0618A A RESTART OF A NON-ABNORMALLY TERMINATED SYSTEM MUST SPECIFY EMERGENCY BACKUP OR OVERRIDE.
DFS3131I A COLD START OR EMERGENCY RESTART REQUIRED
DFS3626I RESTART HAS BEEN ABORTED
```

EVISTRMN

Purpose

This routine resets the posted IMS sysplex event to both SDF and NMC.

This routine should be invoked from the NetView automation table.

Syntax

```
►►──EVISTRMN─────────────────────────────────────────────────────────────►◄
```

Usage

The automation routine EVISTRMN is intended to respond to the following message:

```
CQS0206I CQS structurename percentage BELOW THRESHOLD LEVEL
```

Appendix A. CICS Automation and the Program-to-Program Interface

Warning!
CICS Automation uses the program-to-program interface for health checking. If you are considering using the CICS Automation program-to-program interface code for your own purposes, remember that this interface is release-sensitive. The significance of this is that you may need to recompile or make changes to your code to be compatible with new releases of CICS Automation or NetView.

This appendix provides details about the NetView program-to-program interface, as follows:

- "The Program-to-Program Interface"
- "Program-to-Program Interface Components in NetView and CICS" on page 88
- "NetView Requests Using the Program-to-Program Interface" on page 89
- "CICS Requests Using the Program-to-Program Interface" on page 92
- "Programming Interface" on page 94
- "Customizing CICS Definitions" on page 106
- "Definition Members" on page 107
- "Security Checking Using CICS" on page 110

The Program-to-Program Interface

Note that you only need to use the NetView program-to-program interface for health checking and link monitoring as implemented in previous releases.

The NetView program-to-program interface provides the ability to communicate between a NetView application and other address spaces on the same host, such as CICS and IMS. CICS Automation uses this program-to-program interface to:

- Initiate®, from NetView, the execution of a CICS program
- Process a response from this CICS program
- From CICS initiate the execution of a command list or command processor in NetView
- Process a response from this command list or command processor

There are CICS Automation program-to-program interface components in CICS as well as in NetView. Chapter 1, "CICS Automation Concepts," on page 3 provides you with step-by-step procedures that tell you how to install these components so that the interface can be implemented.

NetView Components

There is an optional task (EVENTASK), an initialization member for this optional task, and command processors. The initialization member is described in "EVENTASK: NetView PPI Initialization Member" on page 109. The command processors are described in "EVESNCCI: NetView to CICS Communication Interface" on page 94.

CICS Components

There is a long-running transaction COPC, as well as start and stop transactions to start and stop the CICS program-to-program interface component; there is also a subroutine which is described in "EVESCCCI: CICS to NetView Communication Interface" on page 99, and an initialization member which is described in "EVESPINM: CICS PPI Initialization Member" on page 108.

Communication Components

An ID (RECEIVERID) is defined at both ends of the program-to-program interface, so CICS Automation knows which NetView to sign on to. This RECEIVERID is contained in the initialization members described above. The VTAM applid is used by CICS to sign on to the program-to-program interface. NetView determines which applid relates to which subsystem from the **APPLid** field of the CICS CONTROL policy item.

The CICS Automation program-to-program interface cannot function if the RECEIVERIDs in the initialization members do not match, or if the VTAM applid of the **APPLid** field of the CICS CONTROL policy item does not match the VTAM applid. The customization sections describe how to define the RECEIVERID and the VTAM applid.

Program-to-Program Interface Components in NetView and CICS

Figure 7 illustrates the CICS Automation program-to-program interface components in NetView and in CICS.

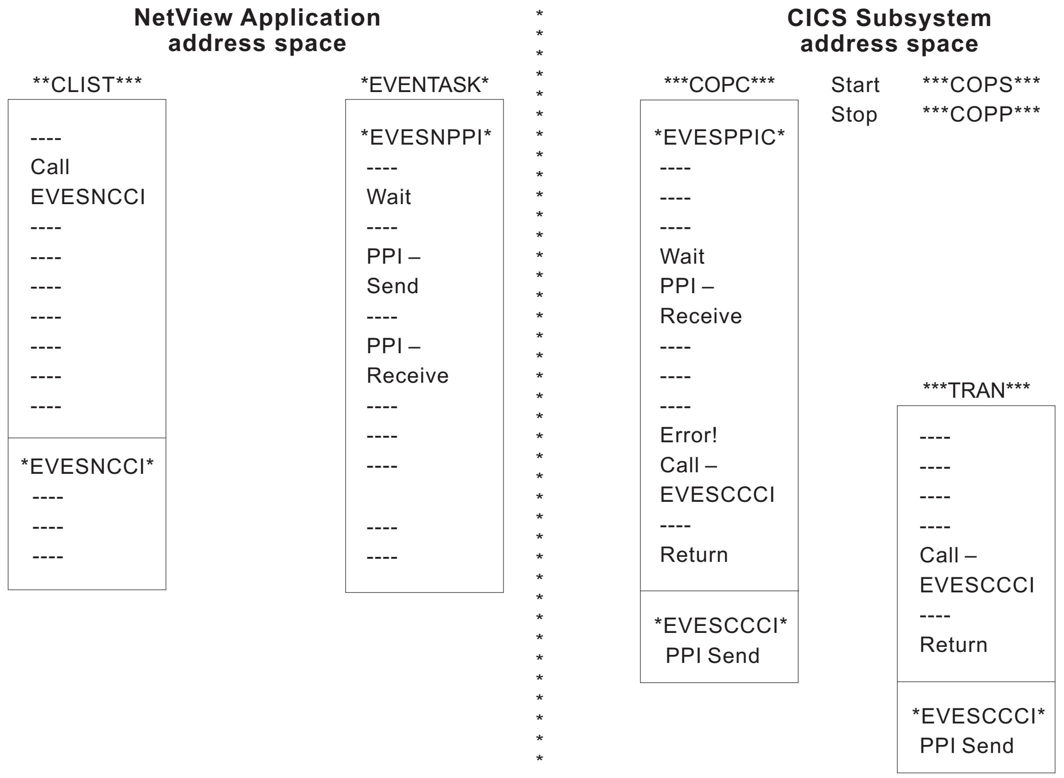


Figure 7. Program-to-Program Interface Components in NetView and CICS

Two of the programs that are shown in Figure 7 have not been previously mentioned:

- EVESNPPI is the NetView subtask program.
- EVESPPIC is the CICS long-running receiver program.

These are internal programs that are not described in this manual.

NetView Requests Using the Program-to-Program Interface

The following requests from NetView are described in this section:
- "CONVERSE from NetView"
- "SEND from NetView" on page 90
- "CANCEL from NetView" on page 91

EVESNCCI is used for these requests. This section only provides an overview of how EVESNCCI works. The programming details, such as command syntax, return codes, and segment support, are provided in "EVESNCCI: NetView to CICS Communication Interface" on page 94.

CONVERSE from NetView

A CONVERSE request from a NetView command list (or command processor) starts a CICS transaction on the same host. The CICS transaction is expected to return a response to a specified NetView task in a named NetView domain. The response can have the form of a RESPONSE, an ACK, or a NACK.

EVESNCCI returns a message and a return code indicating whether the command was successful or not. If the command is accepted, it is forwarded to the EVENTASK optional task.

The EVENTASK optional task uses the program-to-program interface to notify CICS that the transaction is to be started.

The started transaction retrieves the input data. The CICS transaction returns a response to NetView. The response is sent back to the EVENTASK optional task using the program-to-program interface. The processing of the response sent to NetView differs for the various response types, as follows:

A for ACK
The following message is sent to the requestor:

```
EVE128I POSITIVE ACKNOWLEDGEMENT
```

N for NACK
The following message is sent to the requestor:

```
EVE129E response_data
```

Where *response_data* is the data that is returned by the CICS program.

R for RESPONSE
When this request type is used, the NetView program-to-program interface initialization member is interrogated to locate the function requested so that the command list can be identified. The request is then scheduled under the autotask associated with the requested function, after which the data is sent to the requestor.

Figure 8 on page 90 illustrates the flow of events initiated by an EVESNCCI CONVERSE request.

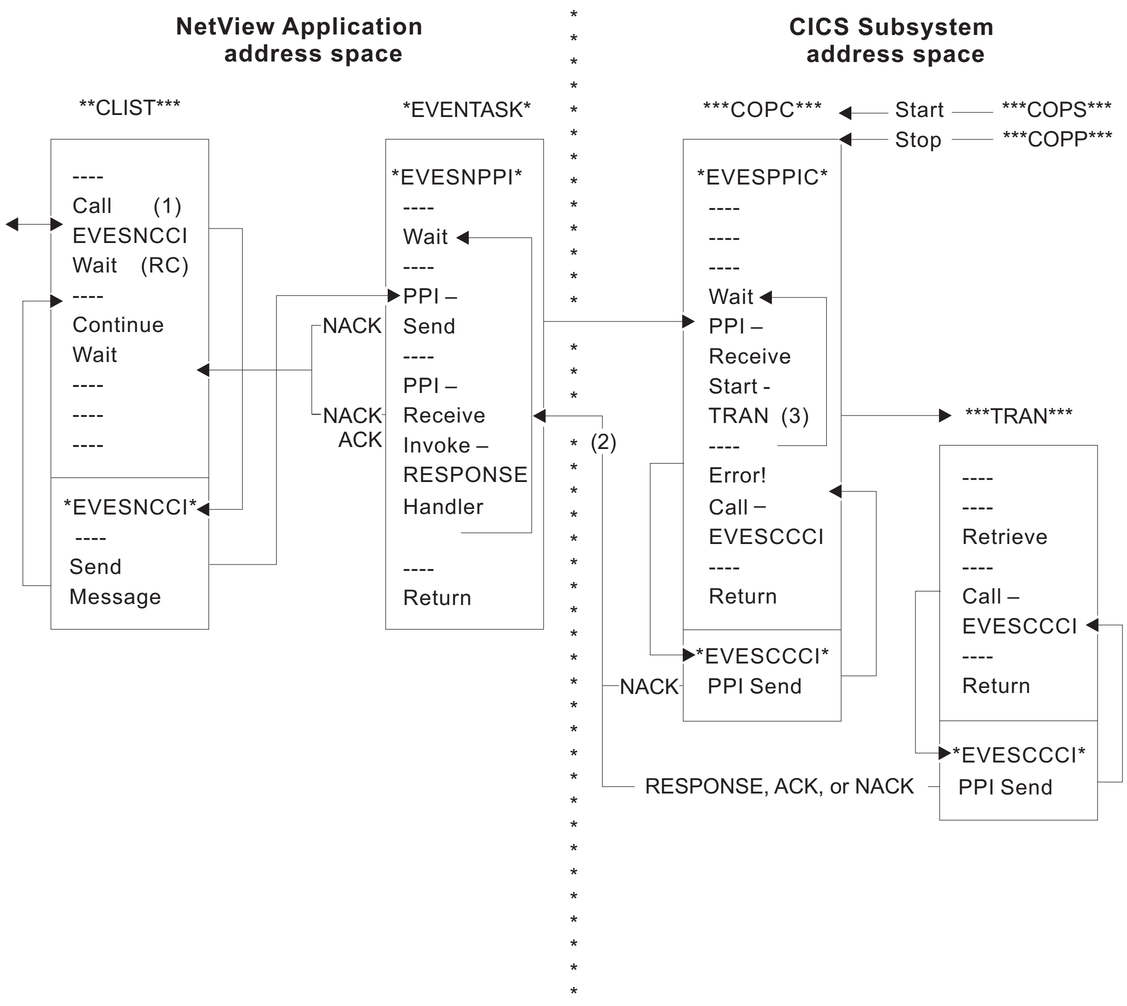


Figure 8. An EVESNCCI CONVERSE Request

Notes:

1. Parameters passed with EVESNCCI indicate the request type (in this case C for CONVERSE), the target CICS, the name of the function to be invoked, data (if required), and the requesting operator ID and domain ID.
2. The responses (RESPONSE, ACK, or NACK) are returned from CICS to the requestor (operator ID and domain ID).
3. The function name is used to locate the transaction name in the CICS initialization member. If the transaction name cannot be found, a NACK response is returned.

SEND from NetView

A SEND request from a NetView command list starts a CICS transaction on the same host. No response is returned. EVESNCCI returns a message and return code indicating whether the command was successful or not. If the command is accepted, it is forwarded to the EVENTASK optional task. This task uses the program-to-program interface to notify CICS to start the transaction.

Errors found by EVESNCCI are returned to the caller. Other errors detected during the processing of a SEND request are not returned. These errors are only logged.

Refer to Figure 9.

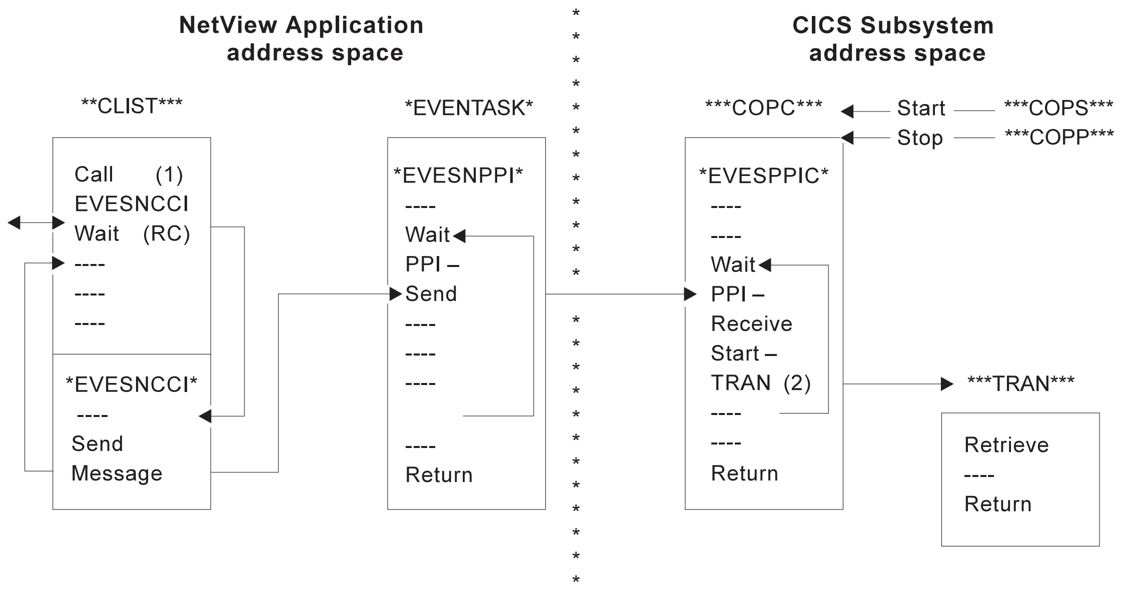


Figure 9. An EVESNCCI SEND Request

Notes:

1. Parameters passed with EVESNCCI indicate the request type (in this case S for SEND), the target CICS, the name of the function to be invoked, and data (if required).
2. The function name is used to locate the transaction name in the CICS initialization member.

CANCEL from NetView

The maximum amount of text that can be specified on the EVESNCCI command is limited to 240 bytes, including the command name EVESNCCI. The maximum amount of data that can be specified varies with the particular EVESNCCI command, but it is always less than 240 bytes. The CICS Automation program-to-program interface implementation provides segment support which allows up to 32656 bytes to be sent from NetView to another program-to-program interface receiver. Segment support is described in “EVESNCCI: NetView to CICS Communication Interface” on page 94 (refer to the SEGMENT= keyword and the usage notes).

The EVESNCCI CANCEL request is used when the segment assembly process must be terminated. This request type causes all saved segments for the specified segment identifier to be freed. If the command is accepted, it is always successful, whether saved segments with the specified segment identifier exist or not.

Users of the segment function are requested to use CANCEL when applicable. This prevents the NetView application system from being filled with unused data.

Errors that are found by EVESNCCI are returned to the caller. Each CANCEL is logged.

CICS Requests Using the Program-to-Program Interface

The following requests from CICS are described in this section:

- CONVERSE
- SEND

EVESCCCI is used for these requests. This section only provides an overview of EVESCCCI. The programming details, such as command syntax and return codes, are provided in "EVESCCCI: CICS to NetView Communication Interface" on page 99.

CONVERSE from CICS

A CONVERSE request from a CICS transaction starts a command list or a command processor in the NetView application system on the same host. The command list or command processor is expected to return a response to the CICS transaction. The response can have the form of a RESPONSE, an ACK, or a NACK.

If the request is accepted, EVESCCCI sends the request to the EVENTASK optional task. The EVENTASK optional task translates the function specification to a command list or command processor name using the EVENTASK initialization member. The autotask under which the command list or processor is to be executed is also obtained from the EVENTASK initialization member.

Errors found by EVESCCCI are returned to the caller. When other errors are detected during the processing of a CONVERSE request, the CICS Automation uses the program-to-program interface to return a NACK response to the CICS transaction. The NACK response data indicates the type of error that occurred. The NACK response is logged via message EVE180E.

Refer to Figure 10 on page 93.

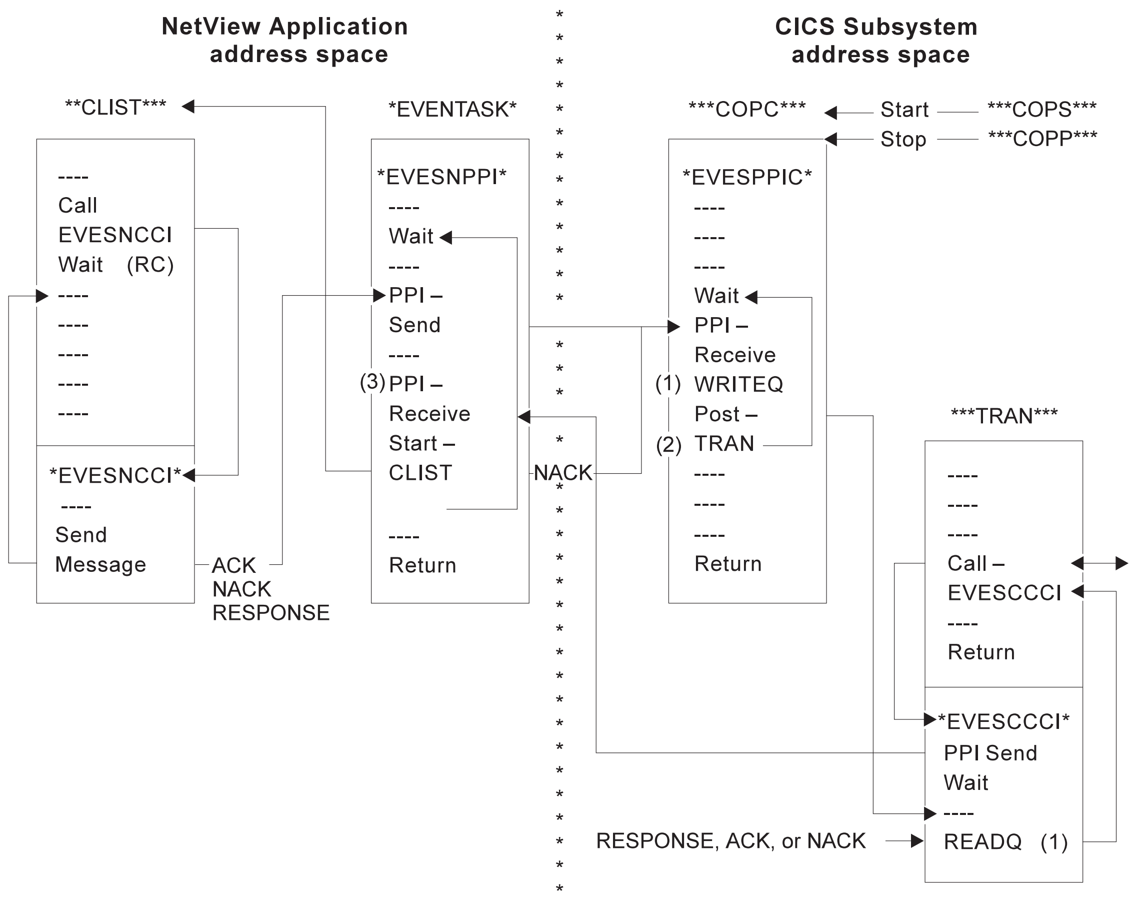


Figure 10. An EVESCCCI CONVERSE Request

Notes:

1. The received data is saved and obtained from temporary storage.
2. The Post-tran is actually a CANCEL of an interval control request.
3. The EVENTASK optional task translates the function specification to a command list or command processor name using the EVENTASK initialization member. The autotask under which the command list or processor is to be executed is also obtained from the EVENTASK initialization member. If the name cannot be obtained or if the scheduling of the command list fails, a NACK response is returned.

SEND from CICS

A SEND request from a CICS transaction starts a command list or a command processor in NetView. No response is returned by the command list or command processor to the CICS transaction.

If the request is accepted, EVESCCCI sends the request to the EVENTASK optional task. The EVENTASK optional task translates the function specification to a command list or command processor name using the EVENTASK initialization member. The autotask under which the command list or processor is to be executed is also obtained from the EVENTASK initialization member. Then, the NetView command processor or command list returns to NetView.

Errors found by EVESCCCI are returned to the caller via a return code and an error code. Other errors detected during the processing of a SEND request are not returned to the requestor. The errors are only logged.

Refer to Figure 11.

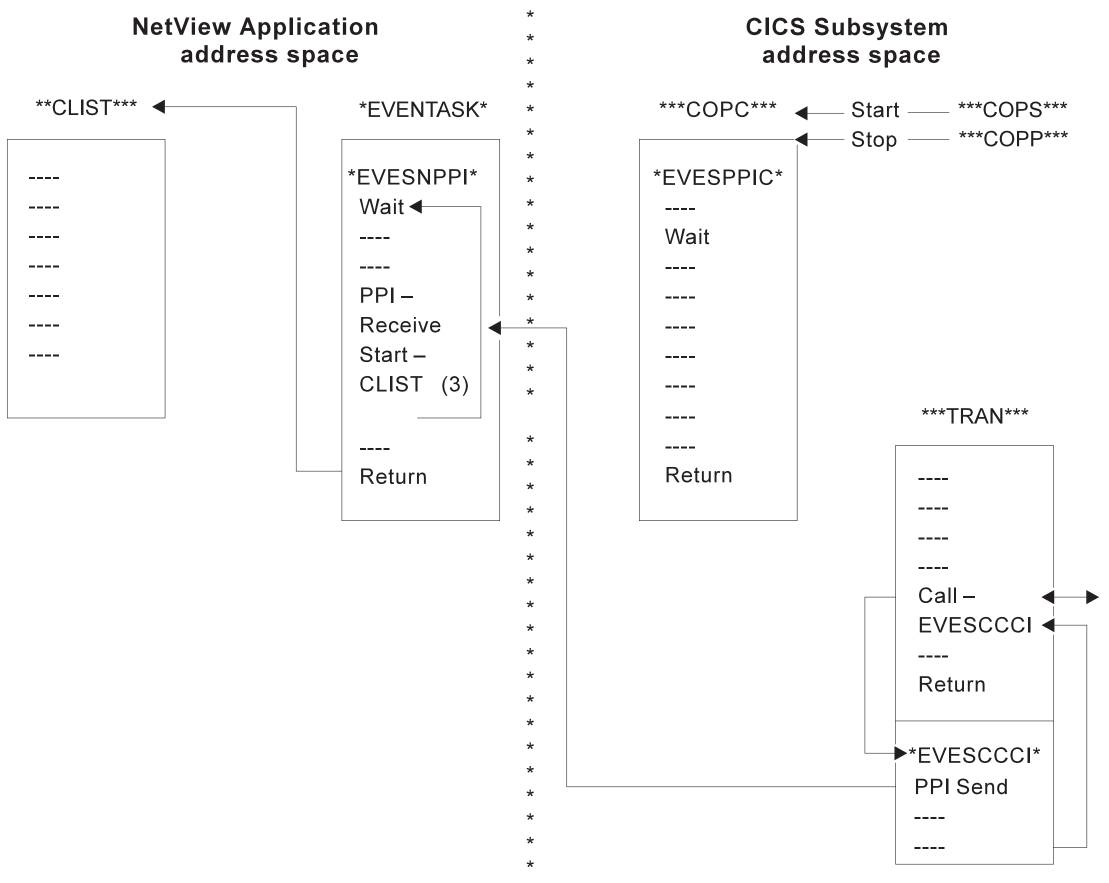


Figure 11. An EVESCCCI SEND Request

Programming Interface

The CICS Automation CICS to NetView program-to-program interface members are:

“EVESNCCI: NetView to CICS Communication Interface”
: Allows you to:
  - Initiate, from NetView, the execution of a CICS transaction.
  - Send a response to a CICS transaction.

“EVESCCCI: CICS to NetView Communication Interface” on page 99.
: Use this subroutine to request, from a CICS transaction, the initiation of a NetView command, a command list, or command processor in NetView.

“EVEMPINT: EVESCCCI Parameter List Copy Book” on page 102.
: The EVEMPINT copy book describes the CICS Automation program-to-program interface parameter list on the CICS side.

EVESNCCI: NetView to CICS Communication Interface

Use this subroutine to:
- Initiate, from NetView, the execution of a CICS transaction.

- Send a response to a CICS transaction.

Syntax

```
EVESNCCI TYPE=C|S|R|A|N|X
         ,NAME=subsnm
         ,FUNC=function
         [,DATA=data]
         [,REQID=reqid]
         [,SEGMENT=segment]
         [,ID=id]
```

Keyword and Parameter Definitions

TYPE=

Specifies the type of command. Valid command types are:

C	For CONVERSE: Starts a process at the other end. A response is expected.
S	For SEND: Starts a process at the other end. No response is expected.
R	For RESPONSE: This is used to send data to the CICS transaction that issued the CONVERSE request.
A	For ACK: Signals the successful completion of a CONVERSE request. No data is provided.
N	For NACK: Signals the unsuccessful completion of a CONVERSE request. Data can optionally be provided (maximum of 100 bytes).
X	For CANCEL: Use this request type to cancel the segment assembly process.

NAME=

Specifies the CICS subsystem name as it is known to CICS Automation.

FUNC=

Specifies the symbolic name of a process to be initiated or to be responded to, such as a CICS transaction or a NetView command list. The relation between a function name and a process name is contained in the EVENTASK and EVESPINM initialization members.

DATA=

Specifies the request or response data. Not accepted for ACK responses. The maximum data length on a NACK response is 100 bytes. If omitted, no data is passed.

Three data delimiter pairs are supported: single quotes, double quotes, or parentheses. These delimiters must be used when the data contains blanks or commas, or starts with one of the data delimiters itself. Data delimiters are stripped off before passing the data on.

REQID=

Specifies the request identification. This field relates the response to the CICS transaction requesting the response. The value is provided on the CICS CONVERSE request and should simply be copied. The request identification may not contain commas or blanks. It is required and only accepted for responses.

SEGMENT=
Use this keyword when the EVESNCCI command must be longer than 240 bytes. The SEGMENT= parameters are:

F	First segment.
M	Neither the first and nor the last segment.
L	Last segment.

When the SEGMENT= keyword is used, the ID= keyword is used to identify the segment chain within a NetView task.

ID=
Identifies the segment chain within a NetView task. This data field is 16 bytes. The segment chain identification may not start with DSI or EVE and may not contain commas or blanks.

Comments and Usage Notes

1. The old parameters DOMAIN= and OPID are ignored.
2. The requested function must be defined in the CICS Automation program-to-program interface initialization members, as shown for the CEMT function:

In NetView (EVENTASK)

```
SERVER=RESPONSE,CEMT,AUTCPPI
```

In CICS (EVESPINM)

```
EVEMPINM TYPE=ENTRY,      DEFINE A FUNCTION
      FUNCTION=CEMT,      FUNCTION NAME
      TRANSID=COMT        TRANSACTION NAME
```

3. The maximum amount of text that can be specified on the EVESNCCI command is limited to 240 bytes, including the command name EVESNCCI.
4. The maximum amount of text that can be specified with a segment chain is 32,656 bytes.
5. EVESNCCI saves the segments until the final segment is received. After receiving the final segment, EVESNCCI assembles the segments into one block which is sent to the specified receiver using the EVENTASK optional task.

   Note: It is assumed that the user of the segment function provides the segments in the correct order and that the segment identification is unique within the NetView task.
6. If an internal error is found during processing of a segment request, such as a GETMAIN failure, all existing segments are deleted.
7. All CANCEL commands (initiated when TYPE=X is used) are logged together with their results. The CANCEL command is always successful. Users of the segment function are requested to use TYPE=X when applicable to prevent NetView from being filled with unused data.
8. The following matrix shows the required (R), optional (O), and invalid (¬) keywords for the various command types:

TYPE=	NAME	FUNC	DATA	OPID	DOMA	REQI	SEGM	ID
C	R	R	O	O	O	¬	O[1]	O[4]

TYPE=	NAME	FUNC	DATA	OPID	DOMA	REQI	SEGM	ID
S	R	R	O	¬	¬	¬	O[1]	O[4]
R	R	R	O	¬	¬	R	O[1]	O[4]
N	R	R	O[2]	¬	¬	R	¬	¬
A	R	R	¬	¬	¬	R	¬	¬
X	¬	¬	¬	¬	¬	¬	¬	R
none	¬	¬	O	¬	¬	¬	R[3]	R[4]

Notes:
1. Only SEGMENT=F can be used with these TYPEs.
2. The length of the data must be less than 101 bytes.
3. Only SEGMENT=M or SEGMENT=L can be used if no TYPE is specified.
4. When a SEGMENT is specified, an ID must also be given.

9. EVESNCCI returns the following return codes and error messages to the caller. Some of the messages are written to the NetView log. Message EVE122E is also sent to the authorized receiver.

RC=0 EVE120I COMMAND ACCEPTED FOR *subsys*, APPLID = *applid*

The command has been forwarded to the EVENTASK optional task, where *subsys* is the symbolic name by which this CICS subsystem is known to CICS Automation, and *applid* is the generic VTAM application identifier of the target CICS subsystem.

RC=4 EVE121E ERROR ON DSI*xxx* REQUEST IN EVESNCCI, RC=*ccc*

The command has not been forwarded to the EVENTASK optional task, where *xxx* identifies the DSI request (such as GET, FRE, FIND, PUSH, or POP), and *ccc* is the return code returned by the DSI*xxx* function.

RC=8 EVE122E EVENTASK TASK NOT ACTIVE

The command has not been forwarded to the EVENTASK optional task.

RC=12 EVE123E INPUT ERROR AT DISPLACEMENT *ddd*, CODE=*ccc*

The command has not been forwarded to the EVENTASK optional task, where *ddd* contains the location in the command string where the error was detected, and *ccc* is one of the following:

004	Unrecognized keyword.
008	Syntax error.
012	Operand error.
016	Duplicate keyword.
020	Conflicting keyword.
024	Required keyword(s) omitted.
028	Incorrect data length. The data length on an ACK response was not zero, or the data length on a NACK response was larger than 100 bytes.

RC=16 EVE124E SEGMENT ERROR, CODE = *ccc*

The command has not been forwarded to the EVENTASK optional task. All existing segments that have the current ID are deleted, except when the segment-chain was corrupted, where *ccc* is one of the following:

004 SEGMENT SEQUENCE ERROR. A middle or last segment has been offered while no first segment with an identical ID was available, or a first segment has been offered while another first segment with the same ID already exists.

008 TOO MUCH DATA. In a series of segments with identical ID the total amount of data exceeds 32656 bytes.

012 SEGMENT-CHAIN CORRUPTED. Storage used for saving segment data has been overwritten.

RC=20 EVE125E NO STORAGE AVAILABLE ON DSI*xxx* REQUEST IN EVESNCCI

The command has not been forwarded to the EVENTASK optional task.

RC=24 ERROR ON EVESX*nnn* CALL IN EVESNCCI, RC = *ccc*

The command has not been forwarded to the EVENTASK optional task. EVESNCCI calls the EVESX001 (CICSSEC) and EVESX022 (CICSQRY) command processors. A non-zero return code from either of these command processors results in this return code and error message. This error implies (normally) that either the caller is not authorized for the function on the specified subsystem, or the specified subsystem is not defined.

10. When other errors are detected during the processing of a CONVERSE request, CICS Automation program-to-program interface returns a NACK response to the requestor. The NACK response data indicates the type of error that occurred. The errors are also logged.

    The following NACK responses can be expected:

```
EVE129E text
        EVE122E task TASK NOT ACTIVE
        EVE136E ERROR ON PPI REQUEST rrr, RC = ccc
        EVE137E NETVIEW SUBSYSTEM NOT AVAILABLE
        EVE141E INCORRECT MQS BUFFER RECEIVED IN progname
        EVE142E FUNCTION funcname NOT FOUND IN membname
        EVE171E procname : ERROR IN progname(tran), CODE = cccc
        EVE175E procname : FUNCTION funcname NOT FOUND IN EVESPINM
        EVE181E procname : ERROR ON TRANSACTION START tran FOR FUNCTION
                funcname
```

Examples of Usage

Example 1

```
"EVESNCCI TYPE=C,"||,
         "NAME=CICS1,"||,
         "FUNC=CEMT,"||,
         "DATA='I PR(E*)'"
```

This command starts a CEMT transaction in the CICS *subsystem* known to CICS Automation as CICS1.

Example 2

```
"EVESNCCI TYPE=R,"||,
        "NAME=CICS2,"||,
        "FUNC=LMT,"||,
        "DATA=(1st segment of LMT response),"||,
        "REQID=1234567890123456,"||,
        "SEGMENT=F,"||,
        "ID=QAZWSXEDCRFVTGBY"

"EVESNCCI SEGMENT=M,"||,
        "DATA=(2nd segment of LMT response),"||,
        "ID=QAZWSXEDCRFVTGBY"

"EVESNCCI SEGMENT=L,"||,
        "DATA=(3rd segment of LMT response),"||,
        "ID=QAZWSXEDCRFVTGBY"
```

This is a link monitor response, which is in 3 segments. Error logic is not included.

EVESCCCI: CICS to NetView Communication Interface

Use this subroutine to request, from a CICS transaction, the initiation of a NetView command, a command list, or command processor in NetView. There are three types of EVESCCCI requests:

1. A CONVERSE request, which expects a response from NetView.
2. A SEND request, which does not expect a response from NetView.
3. RESPONSE.

The request is initiated by setting up a parameter list and linking to the EVESCCCI routine, as shown in the following assembler example:

EXEC CICS LINK PROGRAM(EVESCCCI) COMMAREA(*area*) LENGTH('60')

The parameter list is located in *area*. The following is an example of a parameter list built for a CONVERSE request (see Table 8 on page 101 for a SEND request parameter list example):

Table 6. EVESCCCI CONVERSE Request Parameter List

Name	Disp.	Length	Contents and description
OUTIDENT	000	08	“POUT0001”
OUTREQID	008	16	Not used (set by EVESCCCI)
OUTFNAME	024	08	*function*
OUTRTYPE	032	01	“C”
	033	01	Reserved (binary zeros)
OUTWAITC	034	02	CONVERSE wait time in seconds
OUTDATAL	036	04	Length of request data area (binary)
OUTDATAA	040	04	Address of request data area

Note: Sample copy books for this parameter list are included in the CICS Automation sample library. Refer to "EVEMPINT: EVESCCCI Parameter List Copy Book" on page 102 for field descriptions and important usage information.

If the request data address is located below the line, the response area (if any) is allocated below the line as well. If no data is passed on the CONVERSE request, the length field (OUTDATAL) must be zero. The address field is still used to determine the location of the response area in this case.

If the request is accepted, EVESCCCI sends the request to the NetView EVENTASK optional task, which translates the function into a command list or command processor.

EVESCCCI waits *nn* seconds (see OUTWAITC) for the response to arrive. If the OUTWAITC value is zero, the default wait time is 30 seconds. Valid specifications in the OUTWAITC field range from 1 up to and including 999 (seconds).

If the response does not arrive within the expected interval, the timeout return code is passed to the caller.

EVESCCCI returns the following fields to the caller of the CONVERSE request.

Table 7. EVESCCCI Fields Returned to Caller from CONVERSE Request

Name	Disp.	Length	Contents and description
OUTRESPA	044	4	Address of response area
OUTRESPL	048	4	Length of response area
OUTRTRNC	052	4	Binary return code: **0** Successful request **4** Timeout on CONVERSE **8** Program-to-program interface not available (see OUTABNDC for error code) **12** Incorrect parameter list **16** Internal processing error (see OUTABNDC for error code)
OUTABNDC	056	4	Error code (character) • OUTRTRNC = 8 **C003** Program-to-program interface not active **C015** CICS LOAD/LINK failure **C017** No CICS storage available **C2XX** Program-to-program interface request error • OUTRTRNC = 16 **A...** CICS abend codes **C012** CICS READQ failure **C016** CICS POST failure **C018** CICS FREEMAIN failure **C960** Incorrect TS item length **C961** RQE chain corrupted

Note: Refer to "EVEMPINT: EVESCCCI Parameter List Copy Book" on page 102 for field descriptions and important usage information.

The response area contains the response (if any) on the CONVERSE request. It may contain a RESPONSE, an ACK, or a NACK. The area has the format as described by the EVEMPINT copy book. **It is the responsibility of the caller of EVESCCCI to free the response area via EXEC CICS FREEMAIN.**

The response area format is similar to the transaction input data passed on a CONVERSE or SEND request from NetView:

Name	Disp.	Length	Contents and description
INTIDENT	000	08	"PINT0001"
INTREQID	008	16	*domainid*||*opid*
INTFNAME	024	08	*function*
INTRTYPE	032	01	Response type: "R", "A", or "N"
	033	03	Reserved (binary zeros)
INTDATAL	036	04	Length of *data* (binary). Zero for ACK response
INTSDATA	040	*nn*	*data*

Errors found by EVESCCCI are returned to the caller via a return code and an error code. When other errors are detected during the processing of a CONVERSE request, CICS Automation returns a NACK response to the CICS transaction. The NACK response data indicates the type of error that occurred. The NACK response is logged via message EVE180E. The following NACK responses can be expected:

```
EVE180E text
        EVE121E ERROR ON DISxxx REQUEST IN progname, RC = ccc
        EVE122E tttttttt TASK NOT ACTIVE
        EVE125E NO STORAGE AVAILABLE ON DSIxxx REQUEST IN progname
        EVE142E FUNCTION funcname NOT FOUND IN membname
```

The following is an example of a parameter list built by a SEND request:

Table 8. EVESCCCI SEND Request Parameter List

Name	Disp.	Length	Contents and description
OUTIDENT	000	08	"POUT0001"
OUTREQID	008	16	Not used (set by EVESCCCI)
OUTFNAME	024	08	*function*
OUTRTYPE	032	01	"S"
	033	01	Reserved (binary zeros)
	034	02	Not used for SEND (binary zeros)
OUTDATAL	036	04	Length of request data area (binary)
OUTDATAA	040	04	Address of request data area

Note: Refer to "EVEMPINT: EVESCCCI Parameter List Copy Book" on page 102 for field descriptions and important usage information.

If no data is passed on the SEND request, the length field (OUTDATAL) must be 0 (zero).

On a SEND request, errors found by EVESCCCI are returned to the caller via a return code and an error code. Other errors detected during the processing of a SEND request are not returned to the requestor. The errors are only logged.

EVEMPINT: EVESCCCI Parameter List Copy Book

The EVEMPINT copy book describes the CICS Automation program-to-program interface parameter list on the CICS side. It consists of two parts: one part describes the input data for CICS transactions, and the other part describes the format of output requests from CICS transactions.

The following data area is passed to a started CICS transaction as result of a NetView CONVERSE or SEND request. The same data area is passed in the response area as result of a NetView RESPONSE, ACK, or NACK response. See "EVESNCCI: NetView to CICS Communication Interface" on page 94.

Note: The started transaction must obtain the input data using an EXEC CICS RETRIEVE command.

Name	**Disp.**	**Length**	**Contents and description**
INTIDENT	000	08	Provides an eye-catcher and a block format level.
INTREQID	008	16	This field contains the request identifier which is used to relate a response to a specific request. For a CONVERSE request from NetView, it contains the *domainid*||*opid* concatenation specified on the EVESNCCI command. For responses from NetView, it contains the request identifier allocated by the EVESCCCI routine on a CICS CONVERSE request. This field is not used for a SEND request.
INTFNAME	024	08	Contains the function name specified with the EVESNCCI FUNCTION= keyword.
INTRTYPE	032	01	The response type: C for CONVERSE, S for SEND, R for RESPONSE, A for ACK, and N for NACK.
	033	03	Reserved (binary zeros).
INTDATAL	036	04	Contains the length of the request or response data. This field may have a 0 (zero) value. For an ACK response, this field is 0 (zero).
INTSDATA	040	*nn*	The request or response data, if any. The length of the response data is contained in the INTDATAL field.

The following fields are set by the caller of the EVESCCCI routine.

Name	**Disp.**	**Length**	**Contents and description**
OUTIDENT	000	08	Provides an eye-catcher and a block format level. Must be set to POUT0001.
OUTREQID	008	16	This field is used as a request identifier to relate a response to a specific CONVERSE request. For CICS CONVERSE requests this field is set by the EVESCCCI routine. For CICS responses, the field must be set by the caller (copied from INTREQID). This field is not used for SEND requests.
OUTFNAME	024	08	Specifies the function name. The EVENTASK initialization member must contain a SERVER=REQUEST entry (for CONVERSE or SEND) or a SERVER=RESPONSE entry (for RESPONSE) specification. For responses, this field is normally copied from the transaction input data (INTFNAME field) **Note:** If the name is less than eight characters, pad it with blanks.

Name	Disp.	Length	Contents and description
OUTRTYPE	032	01	Specifies the type of command as REQUEST, SEND, RESPONSE, ACK, or NACK. REQUEST (also referred to as CONVERSE) and SEND are requests. RESPONSE, ACK, and NACK are responses. REQUEST starts a command processor or a command list in NetView. A response is expected. SEND also starts a command processor or a command list in NetView, but no response is expected. RESPONSE is used to send data to a NetView task. ACK signals the successful completion of a CONVERSE request. No data is provided. NACK signals the unsuccessful completion of a CONVERSE request. Data may optionally be provided (maximum of 100 bytes). It is recommended for health checking.
	033	01	Reserved (binary zeros)
OUTWAITC	034	02	Specifies the number of seconds (binary value) to wait for a response on a CONVERSE request. If the response does not arrive within the specified time interval, the timeout return code is passed to the caller. If binary zero is specified, the default wait interval (30 seconds) is used. The maximum binary value that can be specified is 999. The field must contain binary zeros for all requests other than CONVERSE.
OUTDATAL	036	04	This field must be set to the length of the CONVERSE or SEND request data area or to the length of the RESPONSE or NACK response area. The maximum length of a CONVERSE or SEND request area or a RESPONSE response area is 32656 bytes. The maximum length of a NACK response area is 100 bytes.
OUTDATAA	040	04	Specifies the address of the CONVERSE or SEND request area or the address of the RESPONSE or NACK response area. If the request data address is located below the line, the response area (if any) is allocated below the line as well. If no data is passed on the CONVERSE request, the length field (OUTDATAL) must be zero. The address field is still used to determine the location of the response area in this case.

The following fields are set by EVESCCCI upon return to the caller:

Name	Disp.	Length	Contents and description
OUTRESPA	044	04	Address of response area allocated by EVESCCCI. It may contain a RESPONSE, ACK, or NACK response. This field is only returned on a successful CONVERSE request (OUTRTRNC=0). **It is the responsibility of the caller of EVESCCCI to free the response area using EXEC CICS FREEMAIN.**
OUTRESPL	048	04	Contains the length of the response area addressed by the OUTRESPA field.
OUTRTRNC	052	04	Contains the return code which will have one of the following binary values: **000** Successful request. **004** Timeout on CONVERSE. **008** Program-to-program interface not available (see OUTABNDC for error codes). A transaction dump is provided depending on the error code. **012** Incorrect parameter list. No transaction dump is provided. **016** Internal processing error (see OUTABNDC for error codes). A transaction dump is provided.

Name	Disp.	Length	Contents and description
OUTABNDC	056	04	This field contains an error code for return codes 008 and 016. For return code 008 the field will have one of the following character values: **C003** The CICS component of the program-to-program interface is not active. Use the COPS transaction to start it. No transaction dump is provided. An error message is logged. **C015** A CICS LOAD of or LINK to a required module was not successful. Ensure that EVESPERR and EVESPMSG have been properly installed and are enabled. A transaction dump is provided. An error message is logged. **C017** No CICS storage is available. No transaction dump is provided. **C2xx** A program-to-program interface request error occurred, where *xx* contains the program-to-program interface request return code. For error codes C220, C222, C223, C225, C231, C233, C236, C240, and C290, a transaction dump is provided and an error message is logged. For return code 016 the field will have one of the following character values: **A***** A CICS abend has occurred. A transaction dump is provided. An error message is logged for most A*** errors. **C012** An unexpected error has occurred on a READQ command. A transaction dump is provided. An error message is logged. **C016** An unexpected error has occurred on a POST command. A transaction dump is provided. An error message is logged. **C018** An unexpected error has occurred on a FREEMAIN command. A transaction dump is provided. **C961** Internal error. Incorrect TS item length. A transaction dump is provided. An error message is logged. **C962** Internal error. The RQE chain is corrupted. A transaction dump is provided. An error message is logged.

When a RESPONSE response (R) is sent, *function* is used to locate the name of a command processor or command list in the EVENTASK initialization member. This command is scheduled under a task (also defined in the initialization member) with the following command text:

Name	Disp.	Length	Contents and description
	000	8	Command processor or command list name
	008	8	" " (8 blanks)
RHDRCVID	016	8	Program-to-program interface receiver identification
RHDSNDID	024	8	Generic applid (Program-to-program interface sender identification)
RHDDOMID	032	8	*domainid*
RHDTSKID	040	8	*opid*
RHDPRCNM	048	8	Program-to-program interface sender's JOB or STC name
RHDFNAME	056	8	*function*
RHDRTYPE	064	1	"R"
	065	7	"*******" (7 asterisks)
RHDSDATA	072	n	Response data (n = OUTDATAL)

Examples of Usage

Example 1

This assembler example shows the processing of a CICS CONVERSE request.

```
****************************************************************
*        ISSUE A CONVERSE REQUEST                              *
****************************************************************
*
         XC    CISPOUTP,CISPOUTP   ZERO PARAMETER LIST
         LA    R6,CISPOUTP         ADDRESS PARAMETER LIST
         USING OUTDSECT,R6         ESTABLISH ADDRESSABILITY
         SPACE 1
         MVC   OUTIDENT,=C'POUT0001' SET EYE CATCHER/BLOCK LEVEL
         MVC   OUTFNAME,=CL8'TESTCLST' SET FUNCTION NAME
         MVI   OUTRTYPE,OUTRTYPC   SET CONVERSE REQUEST TYPE
         LA    R1,L'CONVTEXT       LENGTH OF VARIABLE DATA
         ST    R1,OUTDATAL         SET LENGTH OF VARIABLE DATA
         LA    R1,CONVTEXT         ADDRESS OF VARIABLE DATA
         ST    R1,OUTDATAA         SET ADDRESS OF CONVERSE DATA
         MVC   CISHWORD,=Y(OUTHL)  LENGTH OF PARAMETER LIST
*
         EXEC  CICS LINK,          REQUEST PROGRAM LINK, TO            *
               PROGRAM('EVESCCCI'), CPDS COMMUNICATION INTERFACE       *
               COMMAREA(CISPOUTP), PARAMETER LIST                      *
               LENGTH(CISHWORD)    PARAMETER LIST LENGTH
*
         L     R15,OUTRTRNC        PICK UP THE RETURN CODE
*        Process the return code and error code!
*
         L     R4,OUTRESPA         ADDRESS RESPONSE AREA
         USING INTDSECT,R4         ESTABLISH ADDRESSABILITY
*
         LA    R14,INTSDATA        ADDRESS RESPONSE DATA
         L     R15,INTDATAL        LENGTH  RESPONSE DATA
*        Do some meaningful processing!
*
CONVTEXT DC    C'CONVERSE TEXT FROM CICS'
*
         DFHEISTG ,
*
CISPOUTP DS    CL(OUTHL)           RESPONSE PARAMETER LIST
CISHWORD DS    H                   PARAMETER BLOCK LENGTH
*
         DFHEIEND ,
*
         EVEMPINT ,                DEFINE INTERFACE DSECT
```

Example 2

This assembler example shows the processing of a NetView CONVERSE request in CICS.

```
****************************************************************
*       RETRIEVE TRANSACTION INPUT DATA                        *
****************************************************************
*
        EXEC  CICS RETRIEVE,      GET INPUT DATA               *
              SET(R4),            RETURN ADDRESS HERE          *
              LENGTH(CISHWORD)    RETURN LENGTH HERE
*
        USING INTDSECT,R4         ESTABLISH ADDRESSABILITY
*
        LA    R14,INTSDATA        ADDRESS REQUEST DATA
        L     R15,INTDATAL        LENGTH  REQUEST DATA
*       Do some meaningful processing!
*
****************************************************************
*       RETURN A 'NORMAL' RESPONSE                             *
****************************************************************
*
        XC    CISPOUTP,CISPOUTP   ZERO PARAMETER LIST
        LA    R6,CISPOUTP         ADDRESS PARAMETER LIST
        USING OUTDSECT,R6         ESTABLISH ADDRESSABILITY
*
        MVC   OUTIDENT,=C'POUT0001' SET EYE CATCHER/BLOCK LEVEL
        MVC   OUTREQID,INTREQID   SET REQUEST IDENTIFIER
        MVC   OUTFNAME,INTFNAME   SET FUNCTION NAME
        MVI   OUTRTYPE,OUTRTYPR   SET RESPONSE RESPONSE TYPE
        LA    R1,L'RESPTEXT       LENGTH OF VARIABLE DATA
        ST    R1,OUTDATAL         SET LENGTH OF VARIABLE DATA
        LA    R1,RESPTEXT         ADDRESS OF VARIABLE DATA
        ST    R1,OUTDATAA         SET ADDRESS OF RESPONSE DATA
        MVC   CISHWORD,=Y(OUTHL)  LENGTH OF PARAMETER LIST
*
        EXEC  CICS LINK,          REQUEST PROGRAM LINK, TO     *
              PROGRAM('EVESCCCI'), CPDS COMMUNICATION INTERFACE *
              COMMAREA(CISPOUTP), PARAMETER LIST               *
              LENGTH(CISHWORD)    PARAMETER LIST LENGTH
*
        L     R15,OUTRTRNC        PICK UP THE RETURN CODE
*       Process the return code and error code!
*
RESPTEXT DC    C'RESPONSE TEXT FROM CICS'
*
        DFHEISTG ,
*
CISPOUTP DS    CL(OUTHL)          RESPONSE PARAMETER LIST
CISHWORD DS    H                  LENGTH OF PARAMETER BLOCK
*
        DFHEIEND ,
*
        EVEMPINT ,                DEFINE INTERFACE DSECT
```

Customizing CICS Definitions

Perform the following additional customization to use the NetView program-to-program interface.

These customization steps are only required if you want to use health checking or link monitoring as implemented in previous releases.

Step 1: Modifications to Program-to-Program Interface Initialization

Step 1a (Optional): Member EVENTASK

This member defines the name of the PPI receiver in NetView for requests from CICS, the default buffer queue limit, and the names and programs of the requests to be executed in NetView. The defaults as specified in this member will work for most installations. If you want to change any of the parameters, see "EVENTASK: NetView PPI Initialization Member" on page 109 for details.

Step 1b (Optional): Member EVESPINM

A sample member is delivered in SINGSAMP (see "EVESPINM: CICS PPI Initialization Member" on page 108 for details). It only needs to be modified if:

- The RECEIVERID is changed. This value must be the same as the value defined in "Step 1: Modifications to Program-to-Program Interface Initialization."
- You are using the program-to-program interface for your own transactions.

In these cases, do the following:

1. Edit EVESPINM. `USERID=YES` means that CICS security checking is required, `USERID=NO` means that CICS security checking is not required.
2. Assemble the program-to-program interface initialization member.
3. Place the assembled member in one of the libraries in the CICS DFHRPL chain.

Step 2: Define a NetView PPI Receiver Task

A PPI receiver is required to allow CICS subsystems to communicate with NetView.

Step 3: Define a CICS PPI Receiver Task

A PPI receiver is required to allow NetView to communicate with CICS subsystems. This is an optional step. It enables operator control of the PPI receivers in each CICS subsystem.

Note: A CICS PPI subsystem may be defined for any CICS subsystem for which the operator wishes to start or stop the PPI receivers in the CICS address space.

Definition Members

The definition members are:

"EVESPINM: CICS PPI Initialization Member" on page 108.
This member is used on the CICS side of the program-to-program interface to:

1. Set the program-to-program interface buffer queue limit. This is the number of outstanding buffers that can be stored in the receiver buffer queue.
2. Define the program-to-program interface receiver identifier for the EVESNPPI NetView subtask program.
3. Provide the identifier of the console on which the long-running COPC transaction is started.
4. Define the relationships between function names and CICS transaction names.

"EVENTASK: NetView PPI Initialization Member" on page 109
This member is used on the NetView side of the program-to-program interface to:

1. Set the program-to-program interface buffer queue limit. This is the number of outstanding buffers that can be stored in the receiver buffer queue.
2. Define request programs and autotasks to be used for specific functions.
3. Define response programs and autotasks to be used for specific functions.
4. Define the program-to-program interface receiver identifier for the EVESNPPI NetView subtask program.

EVESPINM: CICS PPI Initialization Member

This member is used on the CICS side of the program-to-program interface to:

1. Set the program-to-program interface buffer queue limit. This is the number of outstanding buffers that can be stored in the receiver buffer queue.
2. Define the program-to-program interface receiver identifier for the EVESNPPI NetView subtask program.
3. Provide the identifier of the console on which the long-running COPC transaction is started.
4. Define the relationships between function names and CICS transaction names.

Note: There is a corresponding initialization member on the NetView side. See "EVENTASK: NetView PPI Initialization Member" on page 109.

A sample for this member is delivered in the SINGSAMP data set.

Keyword and Parameter Definitions

TYPE=
Indicates the type of entry this is. Valid types are:

INITIAL	The first EVEMPINM type specified. Only one INITIAL entry can be specified.
ENTRY	This type associates a function with a CICS transaction.
FINAL	Indicates that this is the final entry. Only one FINAL entry can be specified.

BUFFQL=
Specifies the buffer queue limit for the CICS receiver side of the program-to-program interface to NetView. A minimum value of 1 and a maximum value of 15 can be specified. If this keyword is omitted, a default value of 3 is assumed. This keyword is only valid with TYPE=INITIAL.

RECEIVERID=
Specifies the identifier of the NetView receiver. If this keyword is omitted, NETVCPPI is assumed. This keyword is only valid with TYPE=INITIAL.

USERID=[YES|NO]
Specifies that the transaction will be invoked with the NetView user ID that invoked the PPI process. The default is NO.

CONSOLE=
Specifies the 1- to 4-character terminal identifier of the console on which the

long-running COPC transaction is started. If this specification is omitted, COPC is started without a terminal. This keyword is only valid with TYPE=INITIAL.

FUNCTION=
The name of the function to be executed. The function name can be from 1 to 8 characters and must not start with the characters EVE.

TRANSID=
The name of the CICS transaction associated with this function. This transaction will be executed when the function is requested.

Comments and Usage Notes

1. A function name may not start with EVE.
2. EVESPINM must be link-edited into one of the CICS DFHRPL libraries.
3. At least one valid TYPE=ENTRY must be specified.
4. There must be a TYPE=ENTRY definition for each function that uses the CICS Automation program-to-program interface. The corresponding NetView side initialization member entry is as follows:
   ```
   SERVER=RESPONSE,CEMT,AUTCPPI
   ```
   Where CEMT is the function.
5. If you are running CICS Automation in more that one NetView domain on the same MVS system, then you need to provide unique RECEIVERIDs in this member. This must also be changed in the corresponding NetView program-to-program interface initialization member. See "EVENTASK: NetView PPI Initialization Member," which explains where the matching RECEIVERID is changed for that member.

EVENTASK: NetView PPI Initialization Member

This member is used on the NetView side of the program-to-program interface to:

1. Set the program-to-program interface buffer queue limit. This is the number of outstanding buffers that can be stored in the receiver buffer queue.
2. Define request programs and autotasks to be used for specific functions.
3. Define response programs and autotasks to be used for specific functions.
4. Define the program-to-program interface receiver identifier for the EVESNPPI NetView subtask program.

Note: There is a corresponding initialization member on the CICS side. See "EVESPINM: CICS PPI Initialization Member" on page 108.

A sample for this member is delivered in the SINGNPRM data set.

Keyword and Parameter Definitions

BUFFQL
This is a 2- or 3-digit numeric value. The minimum value is 10 and the maximum is 999. If this entry is omitted, a value of 15 is assumed.

SERVER=
These entries define:
1. Whether this function is a REQUEST or a RESPONSE. A REQUEST is used to identify a receiver program to be invoked if NetView gets a CONVERSE or SEND from CICS. A RESPONSE is used to identify a sender program to be invoked if CICS sends a RESPONSE. See "EVESCCCI: CICS to NetView Communication Interface" on page 99.

2. The function, such as CEMT.
3. The operator ID that the program runs under, for example, AUTCPPI.
4. The command list or command processor that is used for this function.

RECEIVERID=
The program-to-program interface receiver identifier for the NetView side program-to-program interface subtask program. If omitted, NETVCPPI is assumed.

Comments and Usage Notes

1. A function name may not start with EVE.
2. At least one valid SERVER must be specified.
3. There must be a SERVER entry for each function that uses the CICS Automation program-to-program interface. The corresponding CICS side initialization member entry looks like this:

   ```
   EVEMPINM TYPE=ENTRY,      DEFINE A FUNCTION
         FUNCTION=CEMT,      FUNCTION NAME
         TRANSID=COMT        TRANSACTION NAME
   ```
4. If you are running CICS Automation in more that one NetView domain on the same MVS system, then you need to provide unique RECEIVERIDs in this member. This must also be changed in the corresponding CICS program-to-program interface initialization member EVESPINM. See "EVESPINM: CICS PPI Initialization Member" on page 108.

Security Checking Using CICS

You can use CICS-supplied security to restrict which operators can access defined resources within a CICS environment.

The security check works by using the NetView operator ID that invoked the CICS Automation function. When the function to be performed is invoked in the NetView environment, the invoking operator ID is passed to the CICS system on which the action will be taken. The appropriate transaction or function is invoked, and the NetView operator ID is used in all CICS security checks.

To use this security, you must:

- Define all NetView operators which will invoke CICS functions to RACF® (or your SAF-compliant security system). This will include:

  Regular NetView operators

  NetView autotasks which perform CICS-related actions. These autotasks include those autotasks specifically defined for CICS Automation use, and may include the autotasks which process shutdown functions or resychronization functions.
- Define RACF surrogate authorization for CICS.
- Connect the NetView operators to the CICS resources which they will need to access, such as transactions, programs and files. This connection is done through your SAF security manager (such as RACF).
- Enable the security by modifying the EVESPINM member and specifying USERID=YES to enable extended support. For more information on EVESPINM, see "EVESPINM: CICS PPI Initialization Member" on page 108.
- Enable non-terminal transaction security in CICS by modifying the CICS SIT to specify XTRAN=YES and XUSER=YES. Additional CICS definitions may require similar modification, such as PLTPIUSER.

Note: In order to perform any of the basic functions of CICS Automation, such as displaying subsystem information, an operator must be authorized to use the ACF command. For this command, see *IBM Tivoli System Automation for z/OS Operator's Commands*.

Appendix B. Notices

Programming Interface Information

This book documents programming interfaces that allow the customer to write programs to obtain the services of IBM Tivoli System Automation for z/OS.

Trademarks

Glossary of IMS, CICS, and Other Terms

This glossary defines special IMS and CICS terms used in this book and words used with other than their everyday meaning. In some cases, a definition may not be the only one applicable to a term, but gives the particular sense in which it is used in this book.

A

abend. Abnormal end of task.

ACB. Access method control block (VTAM and VSAM).

access method. A technique for moving data between main storage and input/output devices.

ACK. Acknowledgement.

ANSI. American National Standards Institute.

AOST. Automated Operator Station Task.

APAR. See authorized program analysis report.

application program. (1) A program written for or by a user that applies to the user's work. (2) In data communication, a program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

ASCII. American National Standard Code for Information Interchange.

automation. Computer system control of operation processes.

authorized program analysis report (APAR). A request for correction of a problem caused by a defect in a current unaltered release of a program.

B

batch. An accumulation of data to be processed.

batch message processing. In IMS/VS, a batch processing program that accesses online data bases and message queues.

BMP. Batch Message Processing Region.

C

CCTL. Coordinator Controller.

CEC. See central electronic complex.

CEMT. The CICS master terminal transaction.

central electronic complex (CEC). A conglomeration of several processors and other devices in one or more physical units. This usually means several processors running under the control of a single MVS/ESA operating system. For example, a 3090 model 400® processor complex can run as a 4-processor CEC, or can be partitioned into the equivalent of two 3090 model 200s, each of which runs as a CEC with its own operating system.

central processing complex (CPC). A conglomeration of several processors and other devices in one or more physical units. This usually means several processors running under the control of a single MVS/ESA operating system. For example, a 3090 model 400 processor complex can run as a four-processor CPC, or it can be partitioned into the equivalent of two 3090 model 200s, each of which runs as a CPC with its own operating system.

CICS. Customer Information Control System.

CLIST. See command list.

CNM. Communications Network Management.

command. In IMS and CICS, an instruction similar in format to a high-level programming language statement. CICS commands invariably include the verb EXECUTE (or EXEC). They may be issued by an application program to make use of CICS facilities.

command-language statement. In CICS, synonym for command.

command list (CLIST). A list of commands and statements designed to perform a specific function for the user. Command lists can be written in REXX or in NetView Command List Language.

concurrent. Pertaining to the occurrence of two or more activities within a given interval of time.

CPC. See central processing complex.

D

database. A collection of data fundamental to a system.

database backout. The function of removing changes made to user data sets by in-flight transactions.

database recovery. The function of restoring the user data sets, starting with a backup copy and applying all changes made to each data set after the backup was taken.

data security. The protection of data against unauthorized disclosure, transfer, modifications, or destruction, whether accidental or intentional.

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

DBCTL. Data Base Control.

DEDB. Data Entry Data Base.

DLISAS. Data Language Interface Separate Address Space (IMS Batch).

domain. In IMS, a set of subsystems on a specific NetView domain defined by the system programmer in the control file.

E

end user. In IMS or CICS, anyone using IMS or CICS to do a job, usually by interacting with an application program (transaction) by means of a terminal.

exception. An abnormal condition such as an I/O error encountered in processing a data set or a file, or using any resource.

F

Fast Path Message Region. In IMS, a region that executes programs that require good response characteristics and that have large transaction volumes. Message processing is grouped for load balancing and synchronized for database integrity and recovery.

focal point system. In IMS, a system in which multiple subsystems are interconnected. One subsystem serves as a focal point of control, and the others are referred to as intermediate or distributed systems.

H

HM. Help Message.

HSBID. Hot Standby Identifier.

HSSP. High-Speed Sequential Processing.

I

initial program load (IPL). The initialization procedure that causes an operating system to commence operation.

initialization. (1) Actions performed by IMS or CICS to construct the environment in the IMS or CICS region to enable IMS or CICS applications to be run. (2) A process started by SA z/OS to construct the environment that automation will occur in.

installation. (1) A particular computing system, in terms of the work it does and the people who manage it, operate it, apply it to problems, service it and use the work it produces. (2) The task of making a program ready to do useful work. This task includes generating a program, initializing it, and applying PTFs to it.

INSTALL/IVP. Install/Installation Verification Procedure.

Integrated Resource Lock Manager (IRLM). In IMS automation, this facility is used as a lock manager, both as a single lock manager and in a data sharing environment.

intercommunication facilities. A generic term covering intersystem communication (ISC) and multiregion operation (MRO).

interregion communication (IRC). The method by which CICS provides communication between a CICS region and another region in the same processor. Used for multiregion operation.

intersystem communication (ISC). Communication between separate systems by means of SNA networking facilities or by means of the application-to-application facilities of an SNA access method. ISC links IMS or CICS systems, and may be used for user application to user application communication, or for transparently executing IMS or CICS functions on a remote IMS or CICS system.

IPL. See initial program load.

IRC. See interregion communication.

IRLM. See Integrated Resource Lock Manager.

ISC. See intersystem communication.

IVP. Installation Verification Procedure.

K

keyword. (1) A symbol that identifies a parameter. (2) A part of a command operand that consists of a specific character string. (3) An operand in a CEDA definition. Key-sequenced data set: a VSAM database organization.

L

local. In data communication, pertaining to devices that are attached to a controlling unit by cables, rather than data links.

local device. A device, such as a terminal, whose control unit is directly attached to a computer's data channel. No data link or control unit is used. Contrast with remote device.

local system. In CICS intercommunication, the CICS system from whose point-of-view intercommunication is being discussed.

lock manager. Feature of IMS automation responsible for serializing the recovery process in areas where multiple subsystems can invoke recovery actions.

M

member. See partitioned data set.

MPP. Message Processing Program.

MSC. See Multiple Systems Coupling.

MSDB. Main Storage Data Base.

MTO. Master Terminal Operator.

Multiple Systems Coupling (MSC). An IMS/VS feature that permits geographically dispersed IMS/VS systems to communicate with each other.

N

NACK. Negative Acknowledgement.

NCCF. See Network Communications Control Facility.

network. (1) An interconnected group of nodes. (2) The assembly of equipment through which connections are made between data stations.

network configuration. In SNA, the group of links, nodes, machine features, devices, and programs that make up a data processing system, a network, or a communication system.

Network Communications Control Facility (NCCF). IBM licensed program consisting of a base for command processors that can monitor, control, and improve network operations.

non-XRF (non-XRF IMS). Represent IMS in a non-XRF configuration.

NPDA. Network Problem Determination Aid/Application.

O

OLDS. Online Log Data Set.

online. (1) Pertaining to a user's ability to interact with a computer. (2) Pertaining to a user's access to a computer via a terminal.

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

P

panel. In IMS automation, the set of information displayed on a single screen of the user interface.

parameter. (ISO) A variable that is given a constant value for a specified application and that may denote the application.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data. Synonymous with program library.

PDS. See partitioned data set.

preprocessor. Routine in IMS automation that enables the programmer to define unique GLOBALV names to store the state value of certain processes.

processor. (ISO) In a computer, a functional unit that interprets and executes instructions.

program temporary fix (PTF). A temporary solution or bypass of a problem diagnosed by IBM field engineering as the result of a defect in a current unaltered release of the program.

PTF. See program temporary fix.

PUT. Program update tape.

R

RACF. See Resource Access Control Facility.

RDS. Restart Data Set.

RECON. Recovery Control.

recovery routine. A routine that is entered when an error occurs during the performance of an associated operation. It isolates the error, assesses the extent of the error, and attempts to correct the error and resume operation.

remote. In data communication, pertaining to devices that are connected to a data processing system through a data link.

remote device. A device, such as a terminal, connected to a data processing system through a data link.

remote system. In IMS or CICS intercommunication, a system that the local IMS or CICS system accesses via intersystem communication or multiregion operation.

resource. Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs.

Resource Access Control Facility (RACF). A licensed program that provides for access control by identifying and verifying users to the system, authorizing access to DASD data sets, logging detected unauthorized access attempts, and logging detected accesses to protected data sets.

RMF™. Resource Management Facility.

roll. In IMS automation, the option to begin or rollover to another NetView session. This action is assigned to the PF6 key.

S

SDF. Status Display Facility. The display facility for SA z/OS.

security. Prevention of access to or use of data or programs without authorization.

service. The carrying out of effective problem determination, diagnosis, and repair on a data processing system or software product.

single-point-of-control. Feature of IMS automation enabling the operator to monitor and control IMS subsystems from a single NetView console.

SIT. See system initialization table.

SLDS. System Log Data Set.

SMU. Security Maintenance Utility.

SNA. See systems network architecture.

software. (ISO) Programs, procedures, rules, and any associated documentation pertaining to the operation of a computer.

startup. The operation of starting up IMS or CICS by the system operator.

status code. In IMS/VS, a two-character code in the program communication block (PCB) mask that indicates the results of a DL/1 call.

subsystem. (1) A secondary or subordinate system. (2) A resource defined to SA z/OS and CICS Automation.

system. In IMS or CICS, an assembly of hardware and software capable of providing the facilities of IMS or CICS for a particular installation.

system initialization table (SIT). A table containing user-specified data that will control a system initialization process.

systems network architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

T

task. (1) (ISO) A basic unit of work to be accomplished by a computer. (2) Under IMS or CICS, the execution of a transaction for a particular user.

TCO. Timer-Controlled Operations.

terminal. (1) A point in a system or communication network at which data can either enter or leave. (2) In IMS or CICS, a device, often equipped with a keyboard and some kind of display, capable of sending and receiving information over a communication channel.

terminal operator. The user of a terminal.

transaction. A transaction may be regarded as a unit of processing (consisting of one or more application programs) initiated by a single request, often from a terminal. A transaction may require the initiation of one or more tasks for its execution.

U

update. To modify a file with current information.

V

VSCR. Virtual Storage Constraint Relief.

VTAM. An acronym for the Virtual Telecommunications Access Method. This is one of the ways CICS communicates with terminals.

W

WTOR. Write To Operator with Reply.

X

XRF. Extended recovery facility, a software function that minimizes the effects of various failures on the end users.

Index