System Automation for z/OS

OPC Automation Programmer's Reference and Operator's Guide

Version 2 Release 3

SC33-7046-05

System Automation for z/OS

OPC Automation Programmer's Reference and Operator's Guide

Version 2 Release 3

SC33-7046-05

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" on page xi.

Eighth Edition (June 2004)

This edition applies to System Automation for z/OS Version 2 Release 3 (5645–006), an IBM licensed program, and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for readers' comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

FAX: (Germany) 07031-16-3456
FAX: (Other countries) (+49)+7031-16-3456

Internet: s390id@de.ibm.com

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

Contents

Figures

Tables

Notices

References in this publication to IBM® products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries:

IBM
MVS
NetView
OS/390
PR/SM
RACF
RMF
RMF
TME
VTAM
z/OS

NetView and TME are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

About This Book

This book describes how to customize and operate OPC Automation. The OPC Automation part of SA z/OS brings together batch and online console automation into a common focal point. OPC Automation automates, simplifies, and standardizes console operations and the management of component, application, and production related tasks.

Note: OPC Automation is now a part of IBM Tivoli Workload Scheduler. For consistency, references to OPC and OPC Automation have been maintained in this book. However, there are some references to Tivoli Workload Scheduler, which you should read as OPC Automation.

Who Should Use This Book

This book is intended for the following user groups:

- System programmers, system designers, and application designers who will customize OPC Automation.

  For these users, all three parts of the book will be of interest.

  Installing and customizing OPC Automation requires a programmer's understanding of NetView, OPC, SA z/OS, and OPC Automation, 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 OPC.

  For operators, a working knowledge of OPC will be assumed.

What's in This Book?

This book contains the following:

Part 1, "Introducing OPC Automation"
Explains some main concepts of SA z/OS and describes the functions of OPC Automation.

Part 2, "Operator's Guide"
Describes the actions that an operator can perform with OPC Automation commands.

Part 3, "Programmer's Reference"
Describes the information needed by system programmers to install and customize the OPC Product Automation of System Automation for z/OS. It also describes the old SA z/OS OPC Automation interfaces. These interfaces are provided for compatibility and may be removed in a future release.

Notation for Format Descriptions

The reference sections of this manual contain format descriptions of commands and of entries in the SA z/OS policy database. The notation used for these descriptions is as follows:

- Items shown in braces { } represent alternatives. You must choose one. For example,

`{A|B|C}`

indicates that you must specify one item only: A, B, or C.

- Items shown in brackets [ ] are optional. You may choose one. For example,

  `[A|B|C]`

  indicates that you may enter A, B, or C, or you may omit the operand.
- A series of three periods (...) indicates that a variable number of items may be included in the list.
- An underscored item shows the default that the system will choose if you do not specify an item. For example,

  `[A|B|C]`

  indicates that if no operand is specified, B is assumed.
- Lowercase italicized items are variables; substitute your own value for them.
- Uppercase items must be entered exactly as shown.
- Parentheses must be entered as shown.
- Where operands can be abbreviated, the abbreviations are shown in capital letters. For example, ALL can be entered as A or ALL.
- Commas are used as delimiters between parameters. The last parameter does not require a comma after it. Because of this, we place the comma in front of a parameter to show that if you add this parameter, you need a comma, as for example in

  ```
  XYZ     [A[,B[,C]]]
  ```

  However, the comma follows the preceding parameter and needs to be on the same line as that parameter.

Related Publications

The System Automation for z/OS Library

The following table shows the information units in the System Automation for z/OS library:

Table 1. System Automation for z/OS Library

Title	Order Number
System Automation for z/OS Planning and Installation	SC33-7038
System Automation for z/OS Customizing and Programming	SC33-7035
System Automation for z/OS Defining Automation Policy	SC33-7039
System Automation for z/OS User's Guide	SC33-7040
System Automation for z/OS Messages and Codes	SC33-7041
System Automation for z/OS Operator's Commands	SC33-7042
System Automation for z/OS Programmer's Reference	SC33-7043
System Automation for z/OS CICS Automation Programmer's Reference and Operator's Guide	SC33-7044
System Automation for z/OS IMS Automation Programmer's Reference and Operator's Guide	SC33-7045
System Automation for z/OS OPC Automation Programmer's Reference and Operator's Guide	SC23-7046
System Automation for z/OS Licensed Program Specifications	SC33-7037

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/servers/eserver/zseries/software/sa

Related Product Information

The following table shows the books in the related product libraries that you may find useful for support of the SA z/OS base program.

Table 2. Related Products Books

Title	Order Number
ISPF User's Guide	SC34-4484
ISPF Dialog Management Guide and Reference	SC34-4266
MVS/ESA MVS Configuration Program Guide and Reference	GC28-1817
MVS/ESA Planning: Dynamic I/O Configuration	GC28-1674
MVS/ESA Support for the Enterprise Systems Connection	GC28-1140
MVS/ESA Planning: APPC Management	GC28-1110
MVS/ESA Application Development Macro Reference	GC28-1822
OS/390: MVS System Commands	GC28-1781
MVS/ESA SPL Application Development Macro Reference	GC28-1857
OS/390 Hardware Configuration Definition: User's Guide	SC28-1848
OS/390 Information Roadmap	GC28-1727
OS/390 Information Transformation	GC28-1985
OS/390 Introduction and Release Guide	GC28-1725
OS/390 JES Commands Summary	GX22-0041
OS/390 Licensed Program Specifications	GC28-1728
OS/390 Printing Softcopy Books	S544-5354
OS/390 Starting Up a Sysplex	GC28-1779
OS/390 Up and Running!	GC28-1726
Planning for the 9032 Model 3 and 9033 Enterprise Systems Connection Director	SA26-6100
Resource Access Control Facility (RACF) Command Language Reference	SC28-0733
S/390 MVS Sysplex Overview -- An Introduction to Data Sharing and Parallelism	GC23-1208
S/390 MVS Sysplex Systems Management	GC23-1209
S/390 Sysplex Hardware and Software Migration	GC23-1210
S/390 MVS Sysplex Application Migration	GC23-1211
S/390 Managing Your Processors	GC38-0452
Tivoli/Enterprise Console User's Guide Volume I	GC31-8334
Tivoli/Enterprise Console User's Guide Volume II	GC31-8335

Table 2. Related Products Books (continued)

Title	Order Number
Tivoli/Enterprise Console Event Integration Facility Guide	GC31-8337
Tivoli NetView for OS/390 Administration Reference	SC31-8222
Tivoli NetView for OS/390 Application Programming Guide	SC31-8223
Tivoli NetView for OS/390 APPN Topology and Accounting Agent	SC31-8224
Tivoli NetView for OS/390 Automation Guide	SC31-8225
Tivoli NetView for OS/390 AON Customization Guide	SC31-8662
Tivoli NetView for OS/390 AON User's Guide	GC31-8661
Tivoli NetView for OS/390 Bridge Implementation	SC31-8238
Tivoli NetView for OS/390 Command Reference Vol. 1	SC31-8227
Tivoli NetView for OS/390 Command Reference Vol. 2	SC31-8735
Tivoli NetView for OS/390 Customization Guide	SC31-8228
Tivoli NetView for OS/390 Customization: Using Assembler	SC31-8229
Tivoli NetView for OS/390 Customization: Using Pipes	SC31-8248
Tivoli NetView for OS/390 Customization: Using PL/I and C	SC31-8230
Tivoli NetView for OS/390 Customization: Using REXX and CLIST Language	SC31-8231
Tivoli NetView for OS/390 Data Mode Reference	SC31-8232
Tivoli NetView for OS/390 Installation: Getting Started	SC31-8767
Tivoli NetView for OS/390 Installation: Migration Guide	SC31-8768
Tivoli NetView for OS/390 Installation: Configuring Graphical Components	SC31-8770
Tivoli NetView for OS/390 Installation: Configuring Additional Components	SC31-8769
Tivoli NetView for OS/390 Messages and Codes	SC31-8237
Tivoli NetView for OS/390 MultiSystem Manager User's Guide	SC31-8607
Tivoli NetView for OS/390 NetView Management Console User's Guide	GC31-8665
Tivoli NetView for OS/390 User's Guide	SC31-8241
Tivoli NetView for OS/390 RODM and GMFHS Programming Guide	SC31-8233
Tivoli NetView for OS/390 Security Reference	SC31-8606
Tivoli NetView for OS/390 SNA Topology Manager and APPN Accounting Manager Implementation Guide	SC31-8239
Tivoli Management Platform Reference Guide	GC31-8324
TSO/E REXX/MVS User's Guide	SC28-1882
TSO/E REXX/MVS Reference	SC28-1883
VM/XA SP GCS Command and Macro Reference	SC23-0433
VSE/SP Unattended Node Support	SC33-6412
VTAM Messages and Codes	SC31-6493
VTAM Network Implementation Guide	SC31-6404
VTAM Network Implementation Guide	SC31-6434

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most 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 access LookAt from the Internet at:
http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/ or from anywhere in z/OS or z/OS.e where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS).

The LookAt Web site also features a mobile edition of LookAt for devices such as Pocket PCs, Palm OS, or Linux-based handhelds. So, if you have a handheld device with wireless access and an Internet browser, you can now access LookAt message information from almost anywhere.

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your *z/OS Collection* (SK3T-4269) or from the LookAt Web site's **Download** link.

Accessing z/OS licensed documents on the Internet

z/OS™ licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

To obtain your IBM Resource Link user ID and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

1. z/OS.e™ customers received a Memo to Licensees, (GI10-0684) that includes this key code.

Part 1. Introducing OPC Automation

This part describes some main concepts of SA z/OS, including some NetView-related information, and gives an overview of the facilities offered by OPC Automation.

Subtopics:

- Principal Concepts of SA z/OS
- Functions of OPC Automation

Chapter 1. Principal Concepts of SA z/OS

This section sketches some fundamentals of SA z/OS. For more detailed information see the SA z/OS documentation.

Automation Policies

System automation primarily deals with starting and stopping applications in accordance with their interrelationships. These interrelationships include dependencies of applications on other applications as well as being a component application of an application complex. Also, system automation supports permanent availability of an application by moving the application to another system in case of an unrecoverable abend (see "Application Groups" on page 8).

All applications and systems that you want to include in automation must be defined to SA z/OS in an automation *policy database*. This database contains the objects to be managed by SA z/OS, and the rules according to which automation of these objects proceeds. You access the policy database from the so-called *customization dialogs*. The customization dialogs are described in *System Automation for z/OS Defining Automation Policy*.

The objects that are defined in the policy database are called *policy objects* or *entries*. Applications and systems, for example, are policy objects. Every policy object belongs to an *entry type* which is identified by a three letter code; thus, applications belong to the entry type APL.

Policy objects have automation-related properties and are associated with one another; these properties and connections are called *policy items*. For example, there is a policy item STARTUP for applications that specifies how SA z/OS is to start the application.

What you enter in the policy database are policy objects. However, the objects that can be automated are not these policy objects, but so-called *resources*, which are automatically generated from the policy objects.

This is especially important in the case of applications, since the resources that correspond to an application always represent a *subsystem*, that is, a combination of the application with a system on which it is intended to run; thus, one application can correspond to several subsystems. These resources are generated when an application is linked to a system in the policy database. Note also that some properties and connections are defined on the application (policy object) level (see "Triggers" on page 7) and handed down to all corresponding resources, while others are specified at the resource level (see "Dependencies, Request Propagation, and Desired State" on page 4), and therefore only apply to that resource.

The names of the resources have the following format:

resource_name/*entry_type*[/*system_name*]

The most common entry types are APL (application), APG (application group), and SYS (system). The system name is omitted when the resource is associated with a sysplex, and not a single system.

The policy database must be converted into an *automation control file* (ACF) in order to be accessible to SA z/OS.

Goal-Driven Automation

A basic concept of SA z/OS is to distinguish between the *desired* state of a resource and (broadly speaking) its *actual* state. Every resource has a desired state, which is either AVAILABLE or UNAVAILABLE; AVAILABLE is the default. This desired state, which is also called the automation *goal*, can be different from the actual state; a resource whose desired state is to be running (AVAILABLE), can actually be down. SA z/OS always tries to keep the actual state in line with the desired state, but sometimes this is not possible.

SA z/OS is called goal *driven* because all requests that can be made to it from the outside refer to the desired state of the target resource. When an operator passes a start request for a resource to SA z/OS, this is a request to set the desired state of the resource to AVAILABLE. It is up to SA z/OS to decide whether (1) this is at all possible, and if so, whether (2) the actual state can be modified accordingly:

1. Making a request does not automatically lead to a change of the desired state of the target resource. Rather, SA z/OS compares the *priority* of the new request with that of the last successful request. Only when the new request has a higher priority does SA z/OS change the desired state of the resource. Note that this presupposes that the old request is still available. For more details on this topic, see "Persistency of Requests and Conflicting Requests" on page 6.
2. The latter decision mainly depends on the *dependencies* between the target resource and other resources, and on the *triggers* that may have been associated with it. Dependencies and triggers are defined in the policy database. For more information, see "Dependencies, Request Propagation, and Desired State," and "Triggers" on page 7.

Dependencies, Request Propagation, and Desired State

One of the main tasks of system automation when starting or stopping a resource is to consider the dependencies that exist between the resource to be started/stopped and other resources. Certain resources can only be started when certain other resources are already running (start dependencies), and certain resources can only be stopped when certain other resources are already down (stop dependencies). Note that start and stop dependencies are in principle independent of each other, although if A can only be started when B is running, then it will, as a rule, not be possible to stop B unless A has been stopped beforehand.

Such dependencies can be specified in the policy database. The only restriction is that the dependent and the supporting resource must belong to the same sysplex (they need *not* reside on the same system). SA z/OS takes dependencies into account when it is requested to start or to stop a resource. By default, it will try to start/stop all resources on which the target resource of the request directly or indirectly depends. The mechanism by which this is accomplished is called *request propagation*. It is best explained by an example.

Example 1: Let A, B, and C be resources so that A can only be started when B is running, and B can only be started when C is running. C is supposed to have no start dependencies. Suppose, furthermore, that A, B, and C are all actually down, and that this conforms to their desired state (which is UNAVAILABLE).

Finally, assume that A, B, and C are not associated with any trigger (for the significance of this, see "Triggers" on page 7), and that there are no requests pending for any of the three resources (see "Persistency of Requests and Conflicting Requests" on page 6).

This situation is displayed in Figure 1. The labels of the arrows specify the dependency type. **MakeAvailable/WhenAvailable** is the format in which SA z/OS specifies that the dependent (lower) resource, which is referred to by **MakeAvailable**, can only be started when the supporting (upper) resource, referred to by **WhenAvailable**, is running.

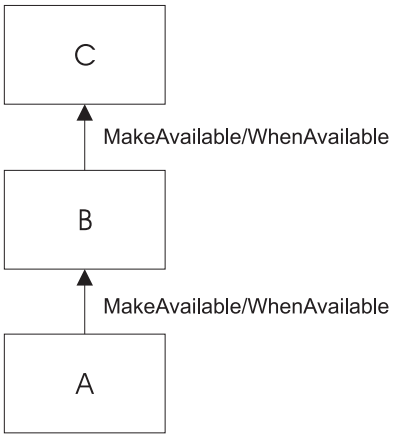


Figure 1. Example of Start Dependencies

When SA z/OS receives a request to start A, the following chain of events will occur:

1. The request is propagated:
   a. Since A can only be started when B is running, a start request is put to B.
   b. Since B can only be started when C is running, a start request is put to C.
2. In response to these requests, the desired state of all three resources is changed to AVAILABLE.
3. SA z/OS tries to change the actual state of the resources according to their desired state:
   a. At first, only C, which has no start dependencies, can be started. B and A cannot be started because C and B are not yet running.
   b. Then B will be started, because C is now available.
   c. Finally, A is started.

The propagated requests are usually called *votes* instead of requests.

In example 1, the request propagation is uniform; the desired state of all three resources is set to AVAILABLE because the condition of the dependency relationships is **WhenAvailable** in both cases. This is not always the case, as the following example shows.

Example 2: Modify example 1 to the effect that B can only be started when C is *unavailable*, and that C is running, in accordance with its desired state AVAILABLE, when the request comes in.

To reflect this modification, the upper arrow label of Figure 1 would have to be changed to **MakeAvailable/WhenDown**. This expresses that

the dependent (lower) resource can only be started when the supporting (upper) resource is unavailable (down).

In example 2, the request must be transformed when propagated from B to C, because in order to start B and then A, C must be down. Therefore, SA z/OS would put a *stop* request to C in this case, and the desired state of C would be set to UNAVAILABLE.

By propagating requests, SA z/OS actively supports the start or stop request. You can also switch off request propagation for a resource. If this were to be done for resource A in example 1, then A would not be started because B is not available, and SA z/OS would do nothing to start B. In this case A would only be started after B had been started, directly or indirectly, through another request.

Persistency of Requests and Conflicting Requests

Requests (and the votes derived from them) are persistent. They are stored in SA z/OS and continue to be taken into account until you explicitly remove them. This implies that there can be more than one request (vote) for the same resource at the same time, and these requests (votes) can be contradictory, as shown in the following example.

Example 3: Expand example 1 by a resource D, also depending on C, which can only be started if C is down. A, B, and C are as in Figure 1 on page 5; D is supposed to be down, and its desired state to be UNAVAILABLE.

Figure 2 contains a graphical presentation of example 3.

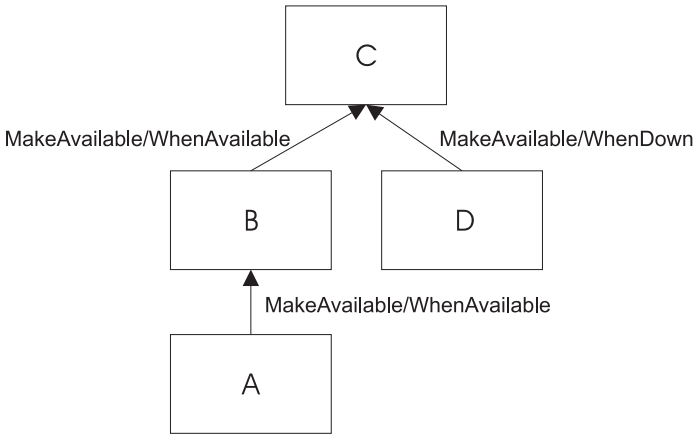


Figure 2. Example of Conflicting Requests

Now assume that first a request to start A and then a request to start D are passed to SA z/OS. The first request results in setting the desired state of C to AVAILABLE. Thereafter the propagation of the start request for D results in a vote to stop C. Since votes are persistent, the previous vote to start C is still existent, and we have two contradictory votes for C. In such a situation, SA z/OS uses the *priority* of the original requests to decide which one of the two votes wins.

When the priority of the old start vote for A is higher than that of the new vote to start D, then the desired state of D will be changed to AVAILABLE, but that of C will remain AVAILABLE; accordingly, SA z/OS will not try to stop C, and thus D cannot be started. If, on the other hand, the vote to stop C has the higher priority,

then the desired state of C is changed to UNAVAILABLE, and SA z/OS will try to stop C in accordance with its desired state, and then to start D. When two contradictory votes have the same priority, a start vote wins over a stop vote.

The persistency concept implies that the losing vote is not automatically discarded. If, for instance, the start request for A wins, the start request for D and the propagated stop vote for C continue to be stored in SA z/OS, and can still be fulfilled after the request for A, and therefore also the start vote for C which was derived from it, have been removed by an operator. After the removal, SA z/OS will determine the desired state of C again and will set it to UNAVAILABLE in response to the stop vote propagated from the start request for D, if no other vote is pending for C. After that, C will be stopped, and then D will be started.

Note that persistency of requests does not apply to successive requests of the same operator. In this case the second request will replace the earlier one.

Triggers

Triggers specify necessary conditions for starting or stopping an application; 'necessary' means that the application can only be started or stopped when the condition is satisfied. Triggers are defined independently of applications. In this way the same trigger can be associated with more than one application. Triggers are defined and linked to an application in the policy database.

The conditions contained in a trigger are either startup conditions or shutdown conditions; there can be more than one startup condition, and also more than one shutdown condition. When a trigger is associated with an application, the resources generated from this application can only be started if *at least one* of the startup conditions in this trigger is satisfied; analogously, they can only be stopped if at least one of the shutdown conditions is fulfilled.

A trigger condition consists of a set of *events*. An SA z/OS event represents an external event that is not under control of SA z/OS, but is relevant to the state of the application associated with the trigger. The information that the external event has or has not occurred is passed to SA z/OS by *setting* or *unsetting* the SA z/OS event; this must be done by an operator or by an automation procedure. A trigger condition is only satisfied when *all* its events are set.

The following example illustrates the use of triggers and their interrelations with dependencies and request propagation.

Example 4: Expand example 1 to the effect that resource C is associated with a trigger that contains only one startup condition. This condition consists of two events, EVENT1 and EVENT2. EVENT1 is set, EVENT2 is unset.

When the request to start A arrives at SA z/OS, it will set off the same sequence of events as with example 1 up to step 2 on page 5. Since, however, the only startup condition of the trigger is not satisfied, C will not be started, and therefore B and A will not be started either. In order to start A, EVENT2 must be set, for example, by an operator. This will lead to a re-evaluation of the startup condition. Since this condition is now satisfied, SA z/OS will start C, and subsequently B and A.

Service Periods

So far we have always assumed that the start or stop requests are made by a human operator. However, SA z/OS also provides the possibility to make start and stop requests at specified points in time independently of human intervention. The objects that are able to do this are called *service periods*. Service periods are defined in the policy database.

A service period is a set of time intervals, so-called *service windows*, during which an application should be available or unavailable. Service periods are defined independently of applications and can then be associated with one or more applications or application groups (see "Application Groups"). When an application is associated with a service period, the service period makes a start request for the application whenever the start time of a service window arrives; this request is canceled when the stop time of the service window arrives. You can also specify service windows during which the application should be unavailable; in this case, a stop request is made at the start, and canceled at the stop time of the service window. The following example is again an expansion of example 1.

Example 5: Resource A of example 1 is associated with a service period that contains at least one service window during which A should be available.

If the start time of this service window arrives, the same sequence of events will occur as with example 1.

An operator can temporarily modify a service period (this is called a *schedule override*). In case of a conflict between a request made by an operator and a request from a service period, the operator request wins when its priority is not lower than that of the service period request.

Application Groups

Modern applications often consist of more than one component, and these different components can be distributed among different systems. SA z/OS provides the possibility to combine different components of an application on one or more systems within a sysplex into an *application group*. This allows you to start and stop a complex application by a single command, and to integrate it into automation processes as a whole.

Example 6: Suppose that resource B of example 1 is an application *group* with the members B1 and B2, and declare A dependent on group B (not on the individual group members), and B dependent on C. You can define B so that every request made to the group as a whole is automatically propagated to every group member.

Figure 3 on page 9 contains a graphical presentation of example 6.

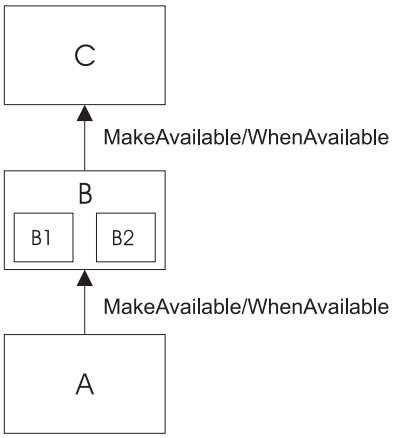


Figure 3. Example of a Request Involving a Group

Then, if you request A to be started, SA z/OS will first, as before, propagate the request to group B and to application C. After C has been started and therefore group B can be started (step 3b on page 5 of example 1), a start vote will be propagated to every member of B. After the desired state of B1 and B2 has been set to AVAILABLE and both resources have been started, B will be considered available, and only then will SA z/OS start A.

In this type of group (which is called BASIC) the group members form a complex entity, and therefore the group is only considered available when *all* its members are available.

The group concept is also used to move applications from their primary system to a backup system when the primary system has failed (group type MOVE). In this case the members of the group are instances of the same application on different systems. In accordance with their purpose, MOVE groups are declared available when *exactly one* of their members is available. You assign preferences to the elements in order to determine which group member is to be started when a start request is put to the group, and which group member takes over when the currently available member is not restartable any more.

SERVER groups are a third type of group. They are a variant of move groups and differ from these mainly in that you can specify how many of its members must be available before the group is considered available. As with move groups, you assign preferences to the members to determine which of them are to be started when a start request is put to the group, and which group members takes over when one of the currently available members is no longer restartable.

Groups can be nested. Suppose, for example, that you have a complex application that you want to be able to move from one system to another. Here you can first define two basic groups G1 and G2, each containing the application on a different system, and then define a move group that contains G1 and G2 as its members.

SA z/OS and the NetView Automation Table

The implementation of SA z/OS is based on NetView®. One important area, where SA z/OS relies on NetView functionality, is the NetView Automation Table (AT). This table serves to automate operator responses to messages that are sent to NetView. It contains instructions of the general form:

When message ABC arrives then issue command XYZ.

Whenever NetView receives a message, it scans the AT. If it finds an entry for the message, it issues the command specified in that entry.

With applications controlled by SA z/OS, the command will typically be one of the generic routines that are shipped with SA z/OS (see *System Automation for z/OS Programmer's Reference*). Many of these routines retrieve information from the ACF and then act according to that information.

A typical example for such information is the MESSAGES/USER DATA policy item of the APPLICATION policy object. Within the MESSAGES/USER DATA policy item, you can associate a command with a message ID (see *System Automation for z/OS Defining Automation Policy*). If you connect this message ID with the generic routine ISSUECMD in the &abrrMAT;, then NetView will execute ISSUECMD when the application sends the message in question to NetView. ISSUECMD, in its turn, will search for the message ID in the ACF entry for this application, and if the message ID is associated there with a command, it will issue this command. For more information on ISSUECMD, see *System Automation for z/OS Programmer's Reference*.

For example, you could associate the message ID `AHL031I`, which is the ID of the startup message sent by the application GTF, with the command `MVS $DMRO'GTF IS NOW UP'` in the MESSAGES/USER DATA policy item for GTF. Then the AT would have to contain an entry like the following:

```
IF MSGID = 'AHL031I'
THEN EXEC(CMD('ISSUECMD AUTOTYP=START') ROUTE(ONE *));
```

Now, when NetView receives the AHL031I message it extracts the job name from the message and calls ISSUECMD. ISSUECMD knows where to find the job name and searches the ACF for the associated application. When it finds GTF, it will look for the AHL031I entry in the MESSAGES/USER DATA policy item and will issue the command that is associated with AHL031I for GTF,

```
MVS $DMRO'GTF IS NOW UP'.
```

For more information on the AT, see *Tivoli NetView for OS/390 Automation Guide*. OPC Automation also has some special generic routines, see Chapter 14, "OPC Automation Common Routines and Data Areas," on page 119.

Chapter 2. Functions of OPC Automation

This chapter describes the basic concept of OPC Automation, explains some aspects of its implementation, and sketches possible uses of OPC Automation.

Basic Concepts

OPC Automation is an extension of SA z/OS that capitalizes on the strengths of NetView, SA z/OS, and OPC by providing the ability to greatly expand job execution, scheduling, monitoring, and alert notification capabilities.

The implementation of OPC Automation requires the introduction of new objects in OPC and in SA z/OS.

Tivoli Workload Scheduler for z/OS Automation consists of two basic functions.

1. Requests from OPC to SA z/OS and associated status updates.
2. Requests from SA z/OS to OPC and associated status updates.

OPC to SA z/OS functions

Tivoli Workload Scheduler needs to be able to request desired state changes to subsystems under the control of System Automation for z/OS. A new request interface has been provided. This interface is designed to work with OPC in a way that is natural for OPC. It takes the form of a batch job that can be used to execute any NetView or SA z/OS command on any NetView interconnected in the enterprise.

The batch job may execute on any system in the sysplex that contains an SA z/OS Agent or NetView Agent running the OPC PPI batch command receiver. This command receiver is a NON-MVS System Automation for z/OS subsystem and may be controlled via the same interfaces as any other System Automation for z/OS subsystem.

The Tivoli Workload Scheduler status change exit has been changed to support WTO'd status changes. The purpose of the exit is to provide information to System Automation for z/OS, which in turn will notify operators via NMC and SDF.

SA z/OS to OPC functions

System Automation for z/OS needs to be able to control the operation of Tivoli Workload Scheduler. OPC operations can be made to wait until a previously requested SA z/OS state change has occurred. SA z/OS will make subsystem, application group, system and system group statuses available to OPC. This is done by reflecting the SA z/OS status in a pair of OPC Special Resources. OPC operations may be coded to wait until the appropriate special resource is in the desired state, thus preventing the batch job stream from proceeding until an OPC request to SA z/OS has been completed.

A new interface, the INGOPC/INGTWS command, has been provided to allow an SA z/OS operator or automation function to request changes to the OPC current plan. This interface may be used for any Controller defined to SA z/OS or any Tracker (defined to SA z/OS) that represents a foreign Controller.

Note: The old SA z/OS OPC commands have been deprecated. They are provided in this release for compatibility purposes and to allow for the migration to the new commands. They will be removed in a future release. The commands concerned are as follows:

1. OPCACMD
   Replaced by INGOPC REQ=LIST.
2. OPCALIST
   Replaced by INGOPC REQ=LIST.
3. OPCAMOD
   Replaced by INGOPC REQ=MOD.
4. OPCAPOST
   Replaced by INGOPC REQ=MOD.

Defining System Automation for z/OS to Tivoli Workload Scheduler

There are no required Tivoli Workload Scheduler definitions. As the interface is a normal batch job as submitted by OPC, there are no extra OPC definitions. You may optionally create a batch job submission workstation that is used to submit the batch jobs to run commands against SA z/OS. The advantage in doing this is that the command processor that the batch job runs can automatically stop the workstation if the SA z/OS Agent or NetView Agent is not up or the PPI interface is not responding. When the SA z/OS Agent or NetView Agent starts up it will automatically restart the previously stopped workstation.

Currently only one PPI receiver non-MVS subsystem is provided in the sample configuration. However, you may start more by specifying a different PPI receiver name for each one to be started. Batch jobs have a parameter that can be used to select the appropriate PPI receiver to execute the commands against. By specifying multiple OPC workstations, one per PPI receiver, OPC can schedule batch jobs to the appropriate receiver and thus get a better throughput of requests. However, only one of the workstations will be restarted when the SA z/OS Agent or NetView Agent starts up. The user can use OPC variable substitution to resolve the name of the PPI receiver from the name of the Workstation the job is assigned to.

Defining Tivoli Workload Scheduler to System Automation for z/OS

The Tivoli Workload Scheduler Installation manual insists that each OPC subsystem in a sysplex is uniquely named. If the old SA z/OS OPC Automation interfaces are not required, then it is possible with the new interfaces to accomplish this requirement. Detailed instructions for defining this configuration can be found in *System Automation for z/OS Defining Automation Policy*, however, basically what is required is a set of AOCCLONE variables that represent the names of the subsystems on each system and a set of subsystem definitions to make use of these CLONE variables.

Tivoli Workload Scheduler consists of a number of MVS and non-MVS subsystems that need to be defined to SA z/OS. These are:

1. The OPC Controller subsystems.
2. The OPC Tracker subsystems.
3. The OPC Server subsystems.
4. The OPC Data Server subsystems.

5. The SA z/OS Batch Job Command Receiver subsystems.
6. The SA z/OS Request receiver subsystems.
7. The SA z/OS Observer subsystem.

Each will be described in turn.

The OPC Controller Subsystems

OPC Controller subsystems may be defined to start simultaneously across multiple systems in a sysplex. However, only one of these subsystems will become the active subsystem. All other Controller subsystems will go into standby mode. Controller subsystems may be defined in a sysplex group, or in a system group that is attached to multiple systems. Controller subsystems require Tracker subsystems to manage the batch job stream.

The OPC Tracker Subsystems

OPC Tracker subsystems may be defined to start simultaneously across multiple systems in a sysplex. Tracker subsystems may be defined in a sysplex group, or in a system group that is attached to multiple systems. Tracker subsystems usually require that JES is running before they can operate correctly.

The OPC Server Subsystems

OPC Server subsystems are usually automatically started by the Controller subsystem that has taken on the ACTIVE role.

The OPC Data Server Subsystems

OPC Data Server or Data Store subsystems contain SYSOUT information collected from JES on behalf of the active Controller.

The SA z/OS Batch Job Command Receiver Subsystems

These subsystems are non-MVS NetView subsystems that run in an SA z/OS Agent or NetView Agent. They provide PPI communications for servicing Batch Job Commands. This allows a batch job to execute a NetView or SA z/OS command and to get the output of the command back at the batch job.

The SA z/OS Request Receiver Subsystems

These subsystems are non-MVS subsystems that run in the SA z/OS NetView Agent. They provide PPI communications to the OPC Controller that allows the OPC Controller to inject requests into the SA z/OS Manager. This allows OPC to control the status of resources being managed by SA z/OS.

System Initialization with OPC Automation

JES starts OPC, which is usually operational at all times, as a task without SA z/OS. OPC Automation then transfers the responsibility of starting OPC from JES to SA z/OS, as described in the following scenario:

- The OPC Tracker has JES as a parent.
- During the IPL process, as soon as JES is running, SA z/OS issues a start command for the Tracker subsystem.
- Once the Tracker has started, SA z/OS issues a start command for the OPC Controller on the control host(s) only.
- Automation continues to initialize the rest of the tasks that are defined to it. OPC Automation restores the status of any OPC-controlled tasks to the last status requested by OPC and waits for OPC to issue new requests.

NetView Interface to OPC Automation

The program-to-program interface (PPI), a high-performance interface, provides synchronization and bi-directional command and message flow between NetView and other applications. OPC provides additional application programming interfaces (APIs), which allow it to be updated by other programs.

The implementation of these interfaces in OPC Automation provides the following capabilities:

- Automation of OPC startup and termination
- Interception of OPC alerts for analysis by the alert operator
- Expansion of the Status Display Facility to provide information about TSO users, batch jobs, critical messages, and OPC errors
- Implementation of a two-way interface between OPC and NetView with SA z/OS:
  - OPC defines and controls interactive applications. Support is provided to start and stop subsystems that are defined to the SA z/OS application.
  - Database tasks can run in both interactive and batch systems with full synchronization between the activities.
  - SA z/OS operators can access OPC calendars and other information as well as update OPC information using the INGOPC or INGTWS command.
- Two user extensions:
  - OPCACOMP allows the start up and shut down of subsystems independently of automation status changes.
  - U*Xxxxxx* allows automation of activities not associated with a specific subsystem.

OPC Automation provides commands and panels that allow a NetView operator to make inquiries and issue requests to OPC without actually logging on to OPC.

OPC Automation Special Resources

OPC Automation is able to globally control the creation and setting of OPC special resources based on the status of SA z/OS monitored subsystems. This will allow OPC to resolve job scheduling dependencies based on the status of SA z/OS managed resources.

The OPC special resources created/set by this function are as follows:

ING.*res_sys*.*res_type*.*res_name*.UP, and
ING.*res_sys*.*res_type*.*res_name*.DOWN

where:

res_sys is the MVS sysid of the system where the subsystem status change was detected. If the resource is a SYSPLEX application group, then the value SYSPLEX is used.

res_type is one of APL, APG, SYS, SYG, or GRP.

res_name is the name of the resource.

UP is a literal that defines the resource as being available only when the resource has an observed status of AVAILABLE and a desired status of AVAILABLE.

DOWN is a literal that defines the resource as being available only when the observed status of the resource is one of the following Automation Manager statuses:

SOFTDOWN, HARDDOWN, STANDBY, UNKNOWN, SYSGONE,

and when its desired status is UNAVAILABLE.

Possible Uses of OPC Automation

In data centers, certain groups assume responsibility for the daily operation of the systems. Frequently, these groups are split into these two areas that perform the following tasks:

- Controlling online systems
- Processing all batch work

User requests for hours of service form the basis for online planning decisions. The time available to process the jobs required for online systems for the next day, as well as requests for other batch work, determines batch processing.

A system using OPC executes the current plan (CP), which contains the information for batch processing. A help desk, hotline, or service-contact point merge user-change requests into the overall schedule. While processing control executes batch processing, operations or master terminal operators control the online systems, thus adding to the confusion. Changes to online availability are frequently manual in nature. For example, instructions to change online availability often consist of slips of paper or phone calls to the operator.

OPC Automation allows changes which influence both batch and online systems through simple OPC dialogs. Because OPC manages both batch and online systems, these changes are needed only in one place. Because the processes are automated with SA z/OS and OPC, no interoperator communications are required. In fact, in a highly automated environment, no operator intervention or awareness of these user-requested changes is necessary.

OPC Automation can automate some of the more complex operator procedures and thereby provide several new functions. The following topics give some examples and scenarios which demonstrate these functions.

Changing Online Hours of Availability

Several possible methods exist for changing the hours of availability of online services. To illustrate these methods, consider the example of a service such as IMS. Assume that the scheduled hours of availability for the IMS online service are 7 a.m. to 6 p.m. In NetView, under control of the current plan, OPC Automation performs the timed start and stop events.

- The help desk gets a request from a user group to extend the IMS hours of availability, for today only, from the original plan of 6 p.m. to 8 p.m. (extended service period).
- The help desk ensures that this extended service is acceptable within the service level agreement for this user group.
- The help desk now makes a change to the OPC current plan to reflect this extended IMS period.
- When the revised scheduled time is reached, now two hours later than usual, OPC executes the operation, requesting that OPC Automation stop IMS.
- OPC Automation requests that SA z/OS application stops IMS.

- Once SA z/OS has successfully stopped IMS, OPC Automation returns an operation-ended status to OPC, fulfilling OPC's dependencies on the online IMS.
- Jobs dependent on the termination of IMS are now released for execution.

No restructuring of the batch processing is necessary if the request is within planned service bounds.

Cycling Individual Online Databases

OPC Automation allows OPC to interact not only with the SA z/OS functions, but also with the MVS and MTO consoles, which enables the scheduling of interrelated sequences of events. For example, it is possible to cycle individual databases rather than the complete online system. Figure 4 shows how this scheduling results in minimum disruptions to online applications.

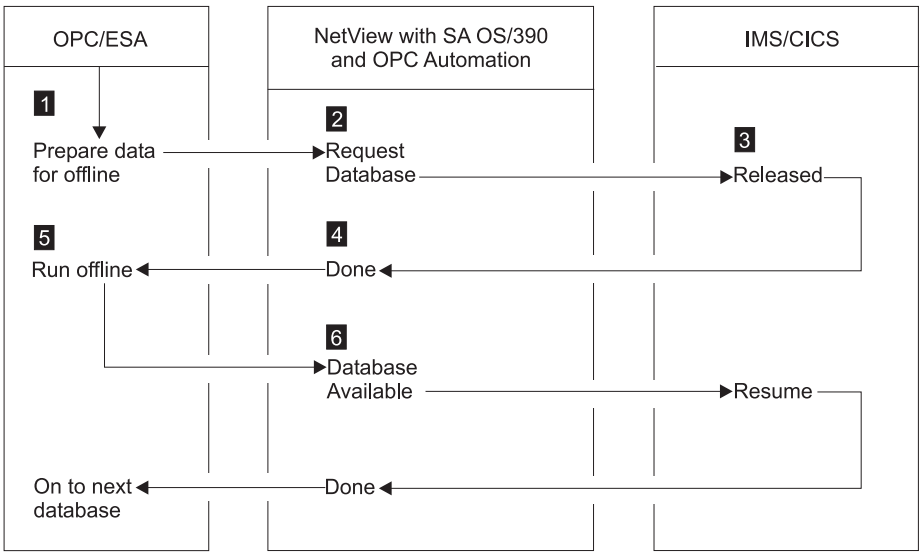


Figure 4. Example of Cycling Individual Online Databases

In Figure 4, the online databases are structured so that you can vary specific ones offline, without an impact to the system, as in the case of databases structured on a geographic or application basis. This process flows as follows:

1. Based on the current plan, OPC begins the READY TO START DATABASE UPDATE job.
2. A request is sent to OPC Automation to issue the command required to vary the subject database offline to allow for batch processing.
3. The request is issued through the MTO interface.
4. OPC Automation ensures that the database is offline.
5. OPC Automation posts the operation as completed in OPC.
6. With the operation completed, OPC dependency starts the batch processing for this database.
7. When the batch process is completed, OPC once again triggers OPC Automation, and the proper MTO command is issued to vary the database online to IMS.

When individual databases accomplish this type of process, the database/data communications (DB/DC) system is always up, and certain small portions of the

data is unavailable for short periods. In some cases, you can restructure the databases to further shorten periods of data unavailability.

OPC Automation does not directly support the preceding example, which requires some user-written modules. (See “Interaction with CICS Automation” on page 141 and “Interaction with IMS Automation” on page 142 for some examples of this type of user-written module.) OPC Automation transports the request to the appropriate system and prepares the information for the user code. OPC Automation then returns the resulting status to the operation in OPC. OPC Automation also ensures that the actions requested are serialized with other requests to that specific target subsystem and that the status of the subsystem is such that it can accept the requests.

Scheduling Time for Testing

Another example is an automated mechanism that prepares a logically partitioned mode (LPAR) on a process resource/system management (PR/SM™) complex for testing periods.

In this example, a system programmer or application developer makes a request through the help desk for testing. The help desk checks that the resources for the test period are available and invokes a prepared OPC-controlled application, updating the information required to set up the time and duration of the test. No other action is needed.

At the proper time, OPC begins execution. It sends the requests to the target system control facility (processor operations) application to set up the LPAR for the test period, and to IML and IPL the PR/SM partition. If the requestor of the test period prepares the test system, so it is ready and waiting at the start of the test period, there is no waiting for an operator to set up the test environment or to structure the system as required by the testing.

Distributing and Updating Data Across Multiple Systems

As centralization of operations and support progresses, preparing data at a central site and then distributing it to other systems becomes necessary. Controlled execution of batch utilities is often required to update the target systems.

Installation of system maintenance provides an example of this type of distribution. Program temporary fixes (PTFs) are installed and tested at a central site. The PTFs are then shipped to target systems and applied with a system modification program (SMP/E). Frequently, a system programmer performs this by logging on to the target system and executing the job streams manually.

Another example is the creation of office system files on a central system, such as electronic telephone directories. These files are then distributed to the target systems.

OPC can control network job entry (NJE) jobs for the distribution of data, and thus controls the execution of the jobs on the target systems, to apply the data, using dependency control, if required.

OPC Automation extends this OPC capability and allows necessary cycling of the target system application once the maintenance is applied successfully. You can schedule this in such a way as to minimize any impact on the end-user community. The following is a typical scenario:

1. A PTF is installed, tested, and found acceptable. This PTF is then applied to all copies of TSO in a multisystem environment.
2. The application is defined to OPC. In most cases, the application is simply updated since it is already defined.
3. OPC presents the batch jobs that control the SMP/E process to the systems programmer for modifications, if required.
4. OPC schedules the transmission of the jobs to the target systems using NJE. The scheduling can use a time when network traffic is low.
5. Once the jobs are in the target system, OPC dependency control is used to schedule the SMP/E job execution.
6. OPC ensures that the SMP/E jobs run correctly. If OPC encounters problems, the OPC application provides backout procedures.
7. After installing the PTFs, OPC selects the appropriate time to issue a request to OPC Automation to restart TSO.
8. Since OPC fully controls the process for this PTF update, you can inquire at any time to see the progress of the operations. If errors or problems occur, OPC Automation informs the SA z/OS notification operator.

Complex Application Recovery

As computer applications become more critical to the daily operation of your enterprise, disaster recovery takes on an added significance. Usually, installations have the necessary equipment and facilities for disaster recovery, but the operational processes are so complicated that the chance of a successful backup in a short period, lasting from many minutes to no more than a few hours, is highly unlikely.

OPC Automation allows full or partial automation of this type of activity between systems and sites. In some cases, changing the NV*xx*-to-NetView domain ID relationship is adequate to transfer the control of the work load to a different system. However, the change may require some manual intervention for synchronization. Chapter 17, “Resynchronization and Recovery Considerations,” on page 157 discusses several scenarios and the process of synchronization.

Although not all steps are required each time, most recoveries consist of three major operational steps that are executed sequentially and provide the following functions:

Step 1: Preparing the recovery system
This may require stopping some or all the applications on the recovery systems, unloading data from disk-storage devices to tape, and reconfiguring the recovery system.

Step 2: Starting the critical applications on the recovery system.
This can include the following:
- Loading databases and applications from tape-to-disk devices
- Starting the recovery system
- Updating data from checkpoint data, logs, or other sources
- Starting critical applications.

Step 3: Returning to the original production system
This is a reversal of the recovery process. These procedures are as complex as the original recovery process, but are scheduled and do not have the urgency of the original recovery.

In the following example, a series of applications need starting on a system after the failure of the original system or possibly even the site. Assume that the installation has prepared properly for this type of problem. This implies tested procedures, current levels of the affected applications and operating environment, and data at the backup site. To simplify this example, assume that the database at the recovery site is adequate for a contingency recovery situation.

- Prior to the need, a series of interdependent recovery applications are defined to OPC, but not scheduled.
- The decision to recover the critical applications at the backup site is made. The scheduler uses normal OPC panels to modify the current plan to schedule the first backup application.
- Before recovery, several factors, which can result in modifications to procedures and JCL, need considering. These modifications are then presented to operators at manual workstations with instructions in the operator instruction files of OPC. They are also presented to systems programmers at JCL workstations.
- The work load on the recovery system is stopped by scheduling a request to SA z/OS to stop all subsystems other than JES.
- Once the subsystems are stopped, a series of jobs are scheduled to transfer data from disk-to-tape to accommodate the requirements of the critical applications that are recovered.
- Depending on the situation, the same system is reused or restructured, and then followed by an IML and IPL of the recovery system. If this is the case, the focal point implementation option of SA z/OS is used to partially or completely automate this phase of the recovery. Regardless of the specifics, the result is an operating system platform ready to accept the recovery environment.
- OPC schedules a series of JES jobs that restore the databases from backup.
- OPC triggers NetView to issue the appropriate commands to start the subsystem.
- In some cases, NetView requires access to MTO functions to issue specific procedures before the DB/DC system can resume transaction processing. If that is the case, user-provided modules are required to fully automate the recovery.

At this point, the recovery is completed. Normal operating procedures should apply to the environment. Because a recovery situation creates an environment where resources are scarce, the actual applications that are offered are frequently a subset of the normal applications. To accommodate this environment, OPC and OPC Automation may need to change the scheduling of some of the applications controlled by OPC.

After recovery occurs and you resolve the problems which forced the original backup, the applications should be moved back to the original system. The scenario for this move is similar to the one above except that this move is planned instead of forced. This allows you to move specific applications one at a time, as opposed to the all-at-once scenario that a critical situation requires. The fact that some of the applications are moved to an already working system makes the takeback more complex than the original recovery.

Give special consideration to any synchronization procedure in an OPC Automation environment. For more information on the synchronization process, see Chapter 17, “Resynchronization and Recovery Considerations,” on page 157.

Part 2. Operator's Guide

This part describes the actions that an operator can perform with OPC Automation commands.

Subtopics:

- Managing the PPI Receivers
- Managing the OPC Current Plan
- Monitoring Using SDF
- NMC Display Support

Chapter 3. Managing the PPI Receivers

This chapter describes how to start up and shut down the NetView PPI receivers. There are two types of receiver subsystems. The ″Request″ receiver is responsible for handling status updates and requests from the OPC exit. The ″Command″ receiver is a new receiver that is responsible for handling commands issued from a batch program.

If requests for automation services from OPC need to be stopped for some reason, then the OPC Request Receiver in the appropriate SA z/OS Agent or NetView Agent should be stopped. Requests are typically starting or stopping a SA z/OS resource and are OPC operations assigned to a workstation with a name like NV**.

If batch interface commands are to be stopped for some reason, then the OPC Command Receiver in the appropriate SA z/OS Agent or NetView Agent should be stopped. Batch commands are typically used to gather information for further processing.

Use the following procedures to start and/or stop the desired PPI receivers. Because there may be more than one Command or Request receiver involved, you should check with your System Programmer to determine the correct subsystems to start and/or stop.

Subtopics:

- Starting and Stopping the Request Receiver
- Starting and Stopping the Command Receivers

Starting and Stopping the Request Receiver

The Request receiver is controlled by SA z/OS. It is defined to SA z/OS as a NON-MVS subsystem. If the system programmer has taken the defaults when customizing OPC Automation, then the name of the Request receiver will be TWSREQSRVR. If this is not the case, then you must find out the SA z/OS subsystem name of the Request receiver from the system programmers.

To start the Request receiver, Issue the INGREQ REQ=START command against the appropriate subsystem, for example:

```
INGREQ TWSREQSRVR/APL/KEY1 REQ=START
```

You should not have to start the Request receiver in normal circumstances because it should automatically start when the SA z/OS Agent registers with the Automation Manager.

To stop the Request receiver, Issue the INGREQ REQ=STOP command against the appropriate subsystem, for example:

```
INGREQ TWSREQSRVR/APL/KEY1 REQ=STOP
```

Starting and Stopping the Command Receivers

The Command receivers are controlled by SA z/OS. They are defined to SA z/OS as NON-MVS subsystems. If the system programmer has taken the defaults when customizing OPC Automation, then there will be only one Command receiver and its name will be TWSCMDSRVR. However, there can be multiple Command receivers and you need to find the names of them from the system programmers.

To start the Command receiver, Issue the INGREQ REQ=START command against the appropriate subsystem, for example:

```
INGREQ TWSCMDSRVR/APL/KEY1 REQ=START
```

You should not have to start the Command receiver in normal circumstances because it should automatically start when the SA z/OS Agent registers with the Automation Manager.

To stop the Command receiver, Issue the INGREQ REQ=STOP command against the appropriate subsystem, for example:

```
INGREQ TWSCMDSRVR/APL/KEY1 REQ=STOP
```

Chapter 4. Managing the OPC Current Plan

A new function has been introduced to allow SA z/OS operators to manage an OPC Current Plan. The operator can list or modify any current plan Application, Operation, Special Resource and Workstation. The operator may list any current plan Calendar. This function is performed via the INGOPC command as described in the *System Automation for z/OS Operator's Commands*.

Selecting the OPC Controller to Access

The INGOPC command allows you to select the OPC Controller to access via the OPC API.

The positional parameter specifies the SA z/OS resource name of the OPC Controller. If the OPC Controller is in a sysplex and may have active and standby Controllers, you may specify an SA z/OS Application Group that contains the set of OPC Controller resources. The INGOPC command will automatically select the active Controller.

Using Multiple Resource Definitions

The Resource Parameter can take multiple arguments contained in brackets and separated by commas, for example:

```
INGOPC (CTL1/APL/SYS1,CTL2/APL/SYS2)
```

In this case both CTL1 and CTL2 OPC Controller subsystems will be scanned to see which is the Active Controller. The Active Controller is the subsystem that is in the AVAILABLE Automation Manager State. SA z/OS OPC Automation will ensure that the standby controller is set to HALTED and thus the Automation Manager State will not be AVAILABLE for standby controllers.

Using Wildcards

The Resource Parameter can take wildcards as defined in the INGLIST command, for example:

```
INGOPC CTL*/APL/*
```

In this case both CTL1 and CTL2 would be scanned as in the example for multiple resource definitions, however, the user specification is shorter.

Using Application Groups

The Resource Parameter may be SA z/OS Application Groups, for example:

```
INGOPC CTLR/APG
```

or:

```
INGOPC CTLG/APG/SYS1
```

Both SYSPLEX and SYSTEM type application groups are allowed. The members of the application groups are found and checked to see if they are OPC type applications. Only the OPC type applications with a subtype of CONTROLLER or TRACKER are checked.

Indirectly Selecting a Controller

In most cases the resource specification will resolve to a single OPC Controller subsystem. However, there are occasions when the Controller subsystem is not present on any system that SA z/OS has access to. In these cases, there must be at least one Tracker on the system or sysplex that SA z/OS is managing. This Tracker must have the ″OPC Control″ policy with the ″Controller ID″ and the ″API LU Name″ policy items specified.

If the INGOPC command cannot resolve the resources that you specify as an active OPC Controller, a final attempt is made to see if any of the resources are an OPC Tracker. The first one found that is in an AVAILABLE Automation Manager state will be used.

The Tracker selected will be used to refer to a remote Controller via the definitions in the ″OPC Control″ policy as specified above.

Displaying the Current Plan

To display the Current Plan, use the INGOPC command with REQ=LIST and specify the type of OPC resource required from:

- APPL for applications
- OP for operations
- SR for special resources
- WS for workstations
- CAL for calendars

Displaying OPC Applications

If you specify TYPE=APPL, this will select OPC Applications for display. To reduce the number of applications that are listed, you can use the optional parameters of AD= and IA= to specify the application ID and the Input Arrival time of the application.

In OUTMODE=LINE, a set of messages will be returned that contains all the data for the selected applications.

The Fullscreen interface is shown in Figure 5.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 6
Domain ID   = IPSFM     -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT            Sysplex = KEY1PLEX         Time = 10:20:33
CMD: A Update    B Operations                                 / scroll
                        Application Occurrance List
                      Input Arrival            Error
CMD  Application Id   Date     Time  Status    Code  Description
---  ---------------- -------- ----- --------- ----- ------------------------
     BIGGERLONGERNAME 02/01/01 10:00 Error
     JKOPCTST1        02/04/17 06:30 Error      jjk1 Test Batch Iface
     JKTEST1          02/04/17 08:00 Completed  JJK2 This is a test
     IEFBR14          02/04/17 08:01 Completed       dummy job
     JKOPCTST1        02/04/18 00:01 Error           Test Batch Iface
     JKTEST1          02/04/18 00:08 Completed       This is a test









Command ===>
 PF1=Help    PF2=End      PF3=Return                 PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 5. OPC Applications Interface Panel

You can scroll left and right with the PF11 (Next) and PF10 (Previous) keys. If more data is available than is displayed on the screen, use the PF8 (Down) or PF7 (Up) keys to scroll the data up or down.

The field descriptions for this panel are:

CMD This accepts the two commands that are available on this panel:

- The A command will invoke the Update panels that allow you to modify the OPC Application.
- The B command will set a filter to display the operations belonging to the OPC Application.

Application ID
The Application ID of the OPC Application.

Input Arrival Date/Time
The Input Arrival Date and Time of the Application. The date and time formats are as specified in the OPC Controller not the formats of the NetView.

Status The OPC status of the application.

Error Code
The Error code of the OPC Application.

Description
The description of the OPC Application.

Figure 6 on page 28 is displayed if you press the PF11 key.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 6
Domain ID   = IPSFM      -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT             Sysplex = KEY1PLEX         Time = 10:26:36
CMD: A Update    B Operations                                    / scroll
                        Application Occurrance List
                     Deadline       Actual Arrival Completion
CMD  Application Id   Date     Time  Date     Time  Date     Time
---  ---------------- -------- ----- -------- ----- -------- -----
     BIGGERLONGERNAME 02/06/04 10:01 02/06/04 06:15   /  /     :
     JKOPCTST1        02/04/17 23:00 02/04/18 07:22   /  /     :
     JKTEST1          02/04/17 11:55 02/08/15 10:02 02/08/26 05:04
     IEFBR14          02/04/17 23:00 02/04/17 05:42 72/00/00 09:37
     JKOPCTST1        02/04/18 23:59 02/04/17 05:42   /  /     :
     JKTEST1          02/04/18 23:00 02/04/17 05:42 02/04/24 05:27




Command ===>
 PF1=Help    PF2=End      PF3=Return                 PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 6. OPC Applications Interface Panel, Screen 2

The field descriptions for this panel are:

CMD This accepts the two commands that are available on this panel:

- The A command will invoke the Update panels that allow you to modify the OPC Application.
- The B command will set a filter to display the operations belonging to the OPC Application.

Application ID
The Application ID of the OPC Application.

Deadline Date/Time
The Date and Time by which this application must have completed.

Actual Arrival Date/Time
The Date and Time when the Application was actually started.

Completion Date/Time
The Date and Time when the Application completed processing.

Figure 7 on page 29 is displayed if you press the PF11 key.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 6
Domain ID   = IPSFM      -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT             Sysplex = KEY1PLEX         Time = 10:29:02
CMD: A Update    B Operations                                  / scroll
                       Application Occurrance List
                    Critical  -------------Operations----------------
CMD  Application Id   W.S. Op#  Number Compl Error Undec. Started Crit.
---  ---------------- ---- ---- ------ ----- ----- ------ ------- -----
     BIGGERLONGERNAME NV01    1      1     0     1      0       1   610
     JKOPCTST1        NV01    1      1     0     1      0       1    10
     JKTEST1                  0      3     3     0      0       6     0
     IEFBR14                  0      1     1     0      0       0     0
     JKOPCTST1        N001    1      1     0     1      0       2     1
     JKTEST1                  0      3     3     0      0       0     0




Command ===>
 PF1=Help    PF2=End      PF3=Return                 PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 7. OPC Applications Interface Panel, Screen 3

The Field descriptions on this panel are as follows:

CMD This accepts the two commands that are available on this panel:

- The A command will invoke the Update panels that allow you to modify the OPC Application.
- The B command will set a filter to display the operations belonging to the OPC Application.

Application ID
The Application ID of the OPC Application.

Critical W.S.
The workstation that is responsible for running an operation in the application that is on the critical path.

Critical OP#
The operation number of the operation in this application that is on the critical path

Operations Number
The number of operations in this application.

Operations Compl
The number of operations that have been completed in this application.

Operations Error
The number of operations that have ended in error in this application.

Operations Undec.
The number of operations that OPC is undecided about in this application.

Operations Started
The number of operations that are running in this application.

Operations Crit.
The Duration in minutes of operations remaining on the critical path.

Displaying OPC Operations

If you specify TYPE=OP, this will select OPC Operations for display. To reduce the number of operations that are listed, you can use the optional parameters of AD=, IA= and OPNO= to specify the application ID, the Input Arrival time of the application and the operation number.

In OUTMODE=LINE, a set of messages will be returned that contains all the data for the selected operations.

The Fullscreen interface is shown in Figure 8.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 3
Domain ID   = IPSFM      -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT             Sysplex = KEY1PLEX         Time = 10:31:53
CMD: A Update                                                  / scroll
                             Operations List
     Op.  -------JES-------                                      Err. Work
CMD  Num. Name     Number   Status    Reason                      Code Stn.
---  ---- -------- -------- --------- ------------------------------ ---- ----
        1 JKTST1   JOB01543 Completed                                   N001
        5 JKTST2   JOB01544 Completed                                   N001
       10 JKTST3   JOB01545 Completed                                   N001
Command ===>
 PF1=Help    PF2=End      PF3=Return                 PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 8. OPC Operations Interface Panel

You can scroll left and right with the PF11 (Next) and PF10 (Previous) keys. If more data is available than is displayed on the screen, use the PF8 (Down) or PF7 (Up) keys to scroll the data up or down.

CMD The A command will invoke the Update panels that allow you to modify the OPC Operation.

Op. Num.
The Operation Number is a unique number within an application that is assigned to an operation.

JES Name
The JES Job Name of the operation.

JES Number
The JES Job Number of the operation if it has been started or submitted.

Status The Status of the operation.

Reason
A reason supplied by OPC for a possible error condition suffered by the operation.

Err. Code
The final error code of the operation.

Work Stn.
The workstation that the operation is assigned to run on.

Figure 9 is displayed if you press the PF11 key.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 3
Domain ID   = IPSFM     -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT            Sysplex = KEY1PLEX         Time = 10:34:34
CMD: A Update                                            / scroll
                              Operations List
     Op.  Job      Planned Start  Planned End    Operation Arr. Deadline
CMD  Num. Name     Date      Time Date      Time Date      Time Date      Time
---  ---- -------- -------- ----- -------- ----- -------- ----- -------- -----
        1 JKTST1   02/04/17 00:08 02/04/17 00:08   /  /     :   02/04/17 11:55
        5 JKTST2   02/04/17 00:08 02/04/17 00:08   /  /     :   02/04/17 11:55
       10 JKTST3   02/04/17 00:08 02/04/17 00:08   /  /     :   02/04/17 11:55
Command ===>
 PF1=Help    PF2=End      PF3=Return                 PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 9. OPC Operations Interface Panel, Screen 2

The field descriptions for this panel are:

CMD The A command will invoke the Update panels that allow you to modify the OPC Operation.

Op. Num.
The Operation Number is a unique number within an application that is assigned to an operation.

Job Name
The JES Job Name of the operation.

Planned Start Date/Time
The Date and Time that OPC planned for the operation to start.

Planned End Date/Time
The Date and Time that OPC planned for the operation to end.

Operation Arr. Date/Time
The Date and Time that the operation actually arrived at the workstation for execution.

Deadline Date/Time
The Date and Time by which this operation must have completed.

Figure 10 on page 32 is displayed if you press the PF11 key.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 3
Domain ID   = IPSFM      -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT             Sysplex = KEY1PLEX         Time = 10:36:33
CMD: A Update                                                 / scroll
                            Operations List
     Op.  Job      Actual Start   Actual End     Est.  Act.       Predecessors
CMD  Num. Name     Date      Time Date      Time Dur.  Dur.  Pri. Num.    Comp
---  ---- -------- -------- ----- -------- ----- ----- ----- ---- ----    ----
        1 JKTST1   02/08/26 05:04 02/08/26 05:04 00:01 00:00    3    0       0
        5 JKTST2   02/08/26 05:04 02/08/26 05:04 00:01 00:00    3    1       1
       10 JKTST3   02/08/26 05:04 02/08/26 05:04 00:01 00:00    3    1       1
Command ===>
 PF1=Help    PF2=End      PF3=Return                 PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 10. OPC Operations Interface Panel, Screen 3

The Field descriptions on this panel are as follows:

CMD The A command will invoke the Update panels that allow you to modify the OPC Operation.

Op. Num.
The Operation Number is a unique number within an application that is assigned to an operation.

Job Name
The JES Job Name of the operation.

Actual Start Date/Time
The Actual Date and Time that the operation started.

Actual End Date/Time
The Actual Date and Time that the operation ended.

Est. Dur.
The Estimated Duration of the operation in hours and minutes as calculated by OPC.

Act. Dur.
The Actual Duration of the operation in hours and minutes.

Pri. The Priority of the operation.

Predecessors Num.
The number of predecessor operations that this operation may wait on.

Predecessors Comp
The number of predecessor operations that have completed.

Figure 11 on page 33 is displayed if you press the PF11 key.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 3
Domain ID   = IPSFM      -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT             Sysplex = KEY1PLEX         Time = 10:38:16
CMD: A Update                                                 / scroll
                            Operations List
     Op.  Job      Job  High -Restart & Cleanup- -------Workstation-------
CMD  Num. Name     Sts  RC   Mode      Status    Name Type     Status  Sub
---  ---- -------- ---- ---- --------- --------- ---- -------- ------- ---
        1 JKTST1   Rel.    0 None                N001 Computer Active
        5 JKTST2   Rel.    0 None                N001 Computer Active
       10 JKTST3   Rel.    0 None                N001 Computer Active









Command ===>
 PF1=Help    PF2=End      PF3=Return                 PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 11. OPC Operations Interface Panel, Screen 4

The field descriptions for this panel are:

CMD The A command will invoke the Update panels that allow you to modify the OPC Operation.

Op. Num.
The Operation Number is a unique number within an application that is assigned to an operation.

Job Name
The JES Job Name of the operation.

Job Sts
The JES Job Status of the job.

High RC
The highest Return Code set by any step in the job.

Restart & Cleanup Mode
The Automatic restart mode for the job.

Restart & Cleanup Status
The Automatic restart status for the job.

Workstation Name
The name of the workstation that controls the job.

Workstation Type
The type of workstation that controls the job.

Workstation Status
The status of the workstation that controls the job.

Workstation Sub.
If the primary workstation is not able to control the operation, this is the name of a substitute workstation that takes over control.

Displaying OPC Special Resources

If you specify TYPE=SR, this will select OPC Special Resources for display. To reduce the number of special resources that are listed, you can use the optional parameter of SRNAME= to specify the special resource name.

In OUTMODE=LINE, a set of messages will be returned that contains all the data for the selected special resources.

The Fullscreen interface is shown in Figure 12.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 40
Domain ID   = IPSFM      -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT             Sysplex = KEY1PLEX         Time = 10:40:48
CMD: A Update                                                 / scroll
                          Special Resources List
                                                    --Actual-- -Default--
CMD  Name                                           Av. Quant. Av. Quant.
---  ---------------------------------------------- --- ------ --- ------
     ING.KEY1.APL.CICS_SA_PPI.DOWN                  No       1 Yes      1
     ING.KEY1.APL.CICS_SA_PPI.UP                    Yes      1 Yes      1
     ING.KEY1.APL.CICSK1G.DOWN                      No       1 Yes      1
     ING.KEY1.APL.CICSK1G.UP                        Yes      1 Yes      1
     ING.KEY1.APL.CICSK1G_PPI.DOWN                  No       1 Yes      1
     ING.KEY1.APL.CICSK1G_PPI.UP                    Yes      1 Yes      1
     ING.KEY1.APL.CICSK1H.DOWN                      No       1 Yes      1
     ING.KEY1.APL.CICSK1H.UP                        Yes      1 Yes      1
     ING.KEY1.APL.CICSK3E.DOWN                      No       1 Yes      1
     ING.KEY1.APL.CICSK3E.UP                        No       1 Yes      1
     ING.KEY1.APL.RMF.DOWN                          No       1 Yes      1
     ING.KEY1.APL.RMF.UP                            Yes      1 Yes      1
Command ===>
 PF1=Help    PF2=End      PF3=Return                 PF5=Filters   PF6=Roll
             PF8=Forward  PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 12. OPC Special Resources Interface Panel

If more data is available than is displayed on the screen, use the PF8 (Down) or PF7 (Up) keys to scroll the data up or down.

The field descriptions for this panel are:

CMD The A command will invoke the Update panels to allow you to modify the OPC Special Resource.

Name The Name of the special resource.

Actual Av.
The Actual Availability of the special resource.

Actual Quant.
The Actual Quantity of the special resource.

Default Av.
The Default Availability of the special resource.

Default Quant.
The Default Quantity of the special resource.

Displaying OPC Workstations

If you specify TYPE=WS, this will select OPC Workstations for display. To reduce the number of workstations that are listed, you can use the optional parameter of WSNAME= to specify the workstation name.

In OUTMODE=LINE, a set of messages will be returned that contains all the data for the selected workstations.

The Fullscreen interface is shown in Figure 13.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 7
Domain ID   = IPSFM      -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT             Sysplex = KEY1PLEX         Time = 10:42:34
CMD: A Update                                                 / scroll
                           Work Stations List
                          Reporting  JCL                  Alt. Para.
CMD  Name Status  Type    Attribute  Prep STC WTO ReRoute WS   Server
---  ---- ------- ------- ---------- ---- --- --- ------- ---- ------
     NV01 Unknown General Automatic  No   No  No  No           No
     NV02 Unknown General Automatic  No   No  No  No           No
     NV03 Unknown General Automatic  No   No  No  No           No
     OPR1 Unknown General Completion No   No  No  No           No
     WTO1 Active  General Automatic  No   No  Yes No           No
     CPU1 Active  Computer Automatic  No   No  No  No           No
     N001 Active  Computer Automatic  No   No  No  No           No




Command ===>
 PF1=Help    PF2=End      PF3=Return                PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 13. OPC Workstations Interface Panel

You can scroll left and right with the PF11 (Next) and PF10 (Previous) keys. If more data is available than is displayed on the screen, use the PF8 (Down) or PF7 (Up) keys to scroll the data up or down.

The field descriptions for this panel are:

CMD The A command will invoke the Update panels to allow the user to modify the OPC Workstation.

Name The Name of the workstation.

Status The Status of the workstation.

Type The Type of the workstation.

Reporting Attribute
The Reporting Attribute of the workstation.

JCL Prep.
If YES then this workstation is used to prepare JCL for submission.

STC If YES then this workstation starts tasks via the started task interface of MVS.

WTO If YES then this workstation can write messages to the system operator.

ReRoute
If YES then when this workstation is not able to process operations, they may be re-routed to the alternate workstation.

Alt. WS
The name of an alternate workstation that operations will be re-routed to when this workstation is unable to process operations.

Para. Server
If YES then this workstation is a parallel server workstation and may run multiple operations simultaneously.

Figure 14 is displayed if you press the PF11 key.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 7
Domain ID   = IPSFM      -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT             Sysplex = KEY1PLEX         Time = 10:45:31
CMD: A Update                                                  / scroll
                              Work Stations List
          --Comp. Ops.-- --Int. Ops.--- -Started- --Ready-- -Waiting-
CMD  Name Num. eDur aDur Num. eDur aDur Num. eDur Num. eDur Num. eDur
---  ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
     NV01    0    0    0    0    0    0    0    0    0    0    0    0
     NV02    0    0    0    0    0    0    0    0    0    0    0    0
     NV03    0    0    0    0    0    0    0    0    0    0    0    0
     OPR1    0    0    0    0    0    0    0    0    0    0    0    0
     WTO1    0    0    0    0    0    0    0    0    0    0    0    0
     CPU1    1    0    0    0    0    0    0    0    0    0    0    0
     N001    6   10    0    0    0    0    0    0    0    0    0    0




Command ===>
 PF1=Help    PF2=End      PF3=Return                 PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 14. OPC Workstations Interface Panel, Screen 2

The field descriptions for this panel are:

CMD The A command will invoke the Update panels to allow the user to modify the OPC Workstation.

Name The Name of the workstation.

Comp. Ops.
Completed Operations assigned to this workstation.

Num. The number of completed operations.

eDur The estimated duration in minutes of completed operations.

aDur The actual duration in minutes of completed operations.

Int. Ops.
Interrupted Operations assigned to this workstation.

Num. The number of interrupted operations.

eDur The estimated duration in minutes of interrupted operations.

aDur The actual duration in minutes of interrupted operations.

Started
Started Operations assigned to this workstation.

Num. The number of started operations.

eDur The estimated duration in minutes of started operations.

Ready Ready Operations assigned to this workstation.

Num. The number of ready operations.

eDur The estimated duration in minutes of ready operations.

Waiting
Waiting Operations assigned to this workstation.

Num. The number of waiting operations.

eDur The estimated duration in minutes of waiting operations.

Displaying OPC Calendars

If you specify TYPE=CAL, this will select OPC Calendars for display. To reduce the number of calendars that are listed, you can use the optional parameter of CALENDAR= to specify the calendar name.

In OUTMODE=LINE, a set of messages will be returned that contains all the data for the selected calendars.

The Fullscreen interface is shown in Figure 15.

```
INGKYST0                SA z/OS - Command Dialogs     Line  1    of 2
Domain ID   = IPSFM     -------- INGOPC    ---------    Date = 09/25/02
Operator ID = KAT            Sysplex = KEY1PLEX         Time = 10:47:14
CMD: No Commands allowed                                        / scroll
                                Calendar List

CMD  Name             Days Shift Description
---  ---------------- ---- ----- ------------------------------
     APC                 8 0000  general APC calendar
     DEFAULT             8 0000  general APC calendar









Command ===>
 PF1=Help    PF2=End      PF3=Return                PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next    PF12=Retrieve
```

Figure 15. OPC Calendar Interface Panel

You can scroll left and right with the PF11 (Next) and PF10 (Previous) keys. If more data is available than is displayed on the screen, use the PF8 (Down) or PF7 (Up) keys to scroll the data up or down.

The field descriptions for this panel are

CMD No commands are currently allowed for the Calendar display.

Name The name of the calendar.

Days The number of days in the calendar definition. Usually 7 days are always present to represent the 7 days of the week.

Shift The time of the start of the shift in HHMM format.

Description
The description of the calendar.

Modifying the Current Plan

To modify the Current Plan, use the INGOPC command with REQ=MOD and specify the type of OPC resource required from:

- APPL for Applications
- OP for Operations
- SR for Special Resources
- WS for Workstations
- CAL for Calendars

Alternatively you can use the modify line commands in the INGOPC display panels.

Line Mode Modifications

You can use the INGOPC command to modify OPC Current Plan resources in line mode. First, you must specify the OPC resource, and then specify the data that is to be modified.

You specify the OPC resource with, selection criteria parameters that are different for each OPC resource type, as shown in Table 3:

Table 3. OPC Resource Type Selection Criteria Parameters

OPC Resource Type	Selection Criteria Parameters
APPL	**AD=** The Application Description of the applications occurrence in the current plan. **IA=** The Applications Input Arrival Time of the applications occurrence in the current plan.
OP	**AD=** The Application Description of the application to which the operation belongs in the current plan. **IA=** The Applications Input Arrival Time of the application to which the operation belongs in the current plan. **OPNO=** The Operation Number of the operation in the current plan.
SR	**SRNAME=** The Special Resource Name in the current plan of the required special resource.
WS	**WSNAME=** The Work Station Name in the current plan of the required work station.

You can specify the data to be modified for a given OPC resource in two ways:

- Firstly, via the command parameter UPDATE=. The data are specified as keyword value pairs separated by an equals sign (=). Pairs of data are separated by the semi-colon character (;). For example:

  ```
  INGOPC ... UPDATE=(PRIORITY=3;STATUS=A)
  ```

- Secondly, via the default safe as passed to the INGOPC command. The data are specified as keyword value pairs separated by " = " (the blanks either side of the equals sign are required). Each pair is contained as a separate message in the default safe. For example:

  ```
  updateStem.0 = 2
  updateStem.1 = 'PRIORITY = 3'
  updateStem.2 = 'STATUS = A'
  'PIPE STEM updateStem. | COLLECT | SAFE * '
  'PIPE NETV INGOPC ......'
  ```

The valid keywords are derived from the OPC-related manual, *Tivoli Workload Scheduler for z/OS Programming Interfaces* (SH19-4545-00). Any keyword as specified in the "Modify Request", "Arguments" section may be used. Table 4 matches the INGOPC TYPE= parameter value to the OPC resource types.

Table 4. INGOPC TYPE= Parameters Matched to OPC Resource Types.

TYPE=	OPC Current Plan Resource	OPC Manual Section
APPL	CPOC	Modify CPOC Arguments
OP	CPOP	Modify CPOP Arguments
SR	CSR	Modify CSR Arguments
WS	CPWS	Modify CPWS Arguments

Modifying OPC Applications via Panel Interaction

From a list of applications (TYPE=APPL) use the "A" (modify) line command against an application. The panel shown in Figure 16 will be displayed:

```
EVJKYRQ1                SA z/OS - Command Dialogs
Domain ID   = IPSFM     ---------- INGOPC   ----------    Date = 09/27/02
Operator ID = KAT                                        Time = 07:22:37
                         Application Modification
  Application =>  BIGGERLONGERNAME
  IA Date/Time=>  0201011000        (YYMMDDHHMM)

  New IA      =>                    (YYMMDDHHMM)
  Deadline    =>                    (YYMMDDHHMM)
  Priority    =>
  Error Code  =>
  Status      =>                    (C or W)
  Group Def.  =>
  JCL Var.Tbl.=>
  Monitor ALL =>                   External Monitor all operations (Y or N)









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

Figure 16. OPC Applications Modification Panel

Fill in the fields to achieve the desired result and press the ENTER key.

Modifying OPC Operations via Panel Interaction

From a list of operations (TYPE=OP) use the ″A″ (modify) line command against an operation. The panel shown in Figure 17 will be displayed:

```
EVJKYRQ2                SA z/OS - Command Dialogs      Page   1 of 3
Domain ID   = IPSFM     ---------- INGOPC   ----------    Date = 09/27/02
Operator ID = KAT                                        Time = 08:57:26
                          Operation Modification
  Application =>  JKTEST1
  IA Date/Time=>  0204170800        (YYMMDDHHMM)
  Operation # =>  1

  Oper. Cmd.  =>                    (EX=Execute/MH=Hold/MR=Release/NP=Nop
                                     UN=Un-Nop)
  Status      =>                    (A/C/E/I/R/S/U/W/*)
  Error Code  =>
  JOB Name    =>
  WS Name     =>                   Workstation job is to run on
  Description =>
  Est. Duratn =>                   Estimated duration of operation (HHMM)
  Parallel Srv=>                   Number of parallel servers used
  R1 Use      =>                   Number of type 1 resources used
  R2 Use      =>                   Number of type 2 resources used
  JCL Class   =>                   Job Class

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

Figure 17. OPC Operations Modification Panel

Fill in the fields to achieve the desired result, then press the ENTER key or press PF11 key to scroll to the next page. If you press the PF11 key, the panel shown in Figure 18 on page 41 is displayed:

```
EVJKYRQ3                SA z/OS - Command Dialogs      Page   2 of 3
Domain ID   = IPSFM     ---------- INGOPC   ----------    Date = 09/27/02
Operator ID = KAT                                        Time = 09:00:46
                        Operation Modification
  Auto. Error =>                Automatic Error Completion (Y or N)
  Auto. Submit=>                Automatic JOB submission   (Y or N)
  Auto. Hold  =>                Automatic JOB hold/release (Y or N)
  Time Depend.=>                Job is dependent on time   (Y or N)
  WLM critical=>                Critical WLM Job           (Y or N)
  WLM policy  =>                WLM Assist Policies        ( /L/D/S/C)
  Cancel Late =>                Cancel job if LATE         (Y or N)
  Highest RC  =>                Highest acceptable Return Code
  Form        =>                Print form name
  OP. IA      =>                Operation Input Arrival    (YYMMDDHHMM)
  OP. Deadline=>                Operation Deadline         (YYMMDDHHMM)
  Re-Route    =>                Re-Route JOB to Alt. WS    (Y or N)
  User Data   =>
  Re-startable=>                Operation is restartable   (Y or N)
  Deadline WTO=>                Issue deadline WTO         (Y or N)
  DSN Clean   =>                Dataset Cleanup Type       (A/I/M/N)

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

Figure 18. OPC Operations Modification Panel, Screen 2

Fill in the fields to achieve the desired result, then press the ENTER key or press PF11 key to scroll to the next page. If you press the PF11 key, the panel shown in Figure 19 is displayed:

```
EVJKYRQ4                SA z/OS - Command Dialogs      Page   3 of 3
Domain ID   = IPSFM     ---------- INGOPC   ----------    Date = 09/27/02
Operator ID = KAT                                        Time = 09:18:07
                        Operation Modification
  Expanded JCL=>                Expanded JCL Option        (Y or N)
  User SYSOUT =>                User SYSOUT Support        (Y or N)
  Ext. Monitor=>                External Monitor           (Y or N)









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

Figure 19. OPC Operations Modification Panel, Screen 3

Fill in the fields to achieve the desired result, then press the ENTER key.

Modifying OPC Special Resources via Panel Interaction

From a list of special resources (TYPE=OP) use the ″A″ (modify) line command against an special resource. The panel shown in Figure 20 will be displayed:

```
EVJKYRQ5                SA z/OS - Command Dialogs
Domain ID   = IPSFM     ---------- INGOPC   ----------    Date = 09/27/02
Operator ID = KAT                                        Time = 09:21:04
                      Special Resource Modification
  SR Name     =>  ING.KEY1.APL.CICS_SA_PPI.DOWN

  Used For    =>                    (C/P/B/N)
  ON Error    =>                    ( /F/FX/FS/K)
  Deviation   =>
  Available   =>                    (Y or N)
  Quantity    =>

  Default Values
  Available   =>                    (Y or N)
  Quantity    =>









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

Figure 20. OPC Special Resources Modification Panel

Fill in the fields to achieve the desired result and press the ENTER key.

Modifying OPC Workstations via Panel Interaction

From a list of workstations (TYPE=OP) use the ″A″ (modify) line command against an work station. The panel shown in Figure 21 on page 43 will be displayed:

```
EVJKYRQ6                 SA z/OS - Command Dialogs
Domain ID   = IPSFM     ---------- INGOPC   ----------    Date = 09/27/02
Operator ID = KAT                                        Time = 09:23:52
                        Workstation Modification
  Workstation =>  NV01

  Reporting   =>                   (A/C/N/S)
  Par. Servers=>                  Number of parallel Servers
  R1 Resources=>                  Number of type 1 resources
  R2 Resources=>                  Number of type 2 resources
  Status      =>                   (A/F/O)
  START act.  =>                   (R/E/L)
  Alt. WS     =>
  WS Linked   =>                   (L/U/ )









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

Figure 21. OPC Workstations Modification Panel

Fill in the fields to achieve the desired result and press the ENTER key.

Chapter 5. Monitoring using SDF

The Status Display Facility uses color to represent the various subsystem resource statuses such as error, warning, action, or informational states. Typically, a subsystem shown in green on a Status Display Facility status panel indicates that it is up, whereas red indicates a stopped or problem state.

The Status Display Facility status display panels can be tailored to present the status of system components in a hierarchical manner. The hierarchical display of status information is implemented using tree structures. A tree structure always starts with the system name as the root component. The ″leaves″ of the tree are the monitored resources.

Color can be propagated up or down the leaves of the tree structure based on the order of dependencies. The effect of propagation is to consolidate, at the root component, the status of all the monitored resources in that system. In this way, the color of the root component reflects the most important or critical status in a computer operations center. If all the monitored resources are green, the root component (the system) will be green.

OPC Automation provides additional Status Display Facility panels that monitor the events that occur in the following areas for all OPC regions defined to OPC Automation:

Applications in Error
Shows the OPC applications that have encountered an error.

Batch Jobs
Shows the Status of OPC Batch jobs in the system.

TSO Users
Shows the Status of TSO Users in the system.

To use the OPC Automation Status Display Facility panels, enter `SDF` on a NetView panel command line. A panel similar to Figure 22 on page 46 will be displayed.

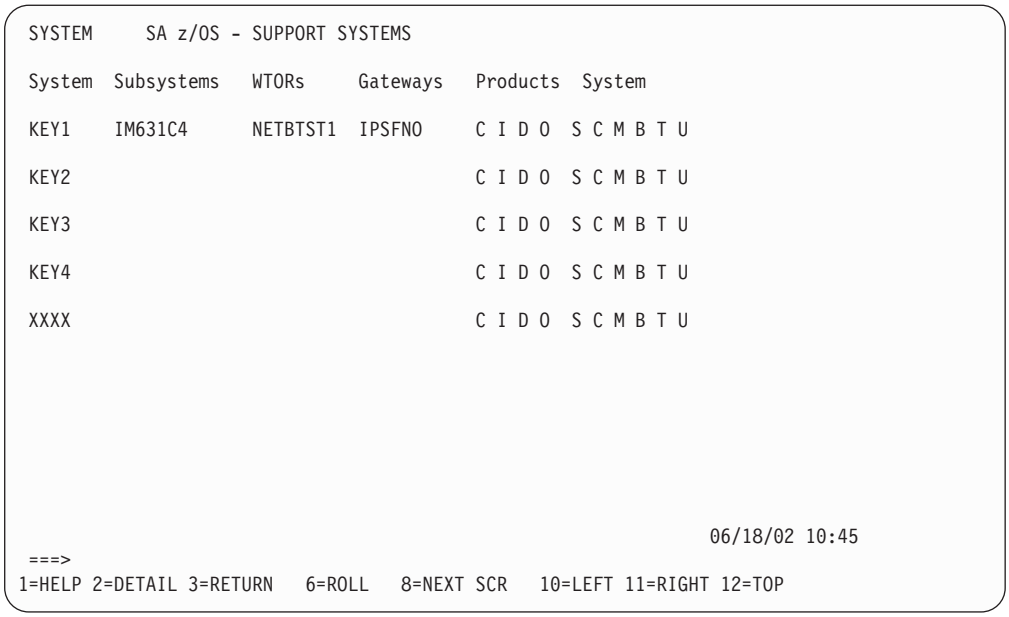


Figure 22. Status Display Facility Main Panel

Note: Sample Status Display Facility panels are provided with OPC Automation. The programmer customizes the panels for your specific environment, so the panels shown here will not look exactly like your panels.

This could be your primary panel that lists the systems and their status. The color of KEY1 through KEY4 will reflect the most critical status of any resource in that system.

If you place the cursor under the letter O on the panel displayed in Figure 22 and press PF8, the panel shown in Figure 23 on page 47 is displayed (assuming you are using the default sample panels).

```
KEY1O                OPC MONITOR PANEL




         Applications in Error

         Batch Jobs

         TSO Users




                                                                  06/18/02 11:25
 ===>
PF1=HELP 2=DETAIL 3=END        6=ROLL 7=UP 8=DN                        12=TOP
```

Figure 23. The OPC Monitor Panel

This shows several categories in which OPC status is important. If the letter O shown on the previous panel was red, then at least one of the items on the OPC Monitor panel will be red.

Chapter 6. NMC Display Support

SA z/OS OPC Product Automation will display the status of OPC applications or operations in error. In addition, the status of TSO users may optionally be displayed.

NMC Resource Definitions

OPC Naming Convention

For OPC status monitoring a new anchor ″OPC″ has been created. This will enable the status of OPC operations to be accessed without the need to know the name of the currently active OPC Controller. OPC resources are represented by objects with a minor name of:

```
jobname
```

TSO Naming Convention

For TSO user monitoring a new anchor ″TSO″ has been created. This likewise enables the status of all TSO users to be accessed from one view. TSO users are represented by objects with a minor name of:

```
systemId_tsoUserName
```

where:

systemId is the name of the system that the TSO user is logged onto.

tsoUserName is the name of the TSO user.

NMC BuildViews for OPC objects

To show the OPC objects on a view off the main tree, use the buildviews statements shown in Figure 24:

```
VIEW=K1.OPC,
  ANNOTATION='OPC Monitoring in Sysplex K1'

WILDCARD=(?,*)

NONSNA=K1.OPC/ANCH*,
  QUERYFIELD=MYNAME
```

Figure 24. Sample OPC Buildviews Statements

To show the TSO objects on a view off the main tree, use the buildviews statements shown in Figure 25 on page 50:

```
VIEW=K1.TSO,
  ANNOTATION='TSO Monitoring in Sysplex K1'

WILDCARD=(?,*)

NONSNA=K1.TSO/ANCH*,
  QUERYFIELD=MYNAME
```

Figure 25. Sample TSO Buildviews Statements

Part 3. Programmer's Reference

This part describes the information needed by system programmers to install and customize the OPC Product Automation of System Automation for z/OS. It also describes the old SA z/OS OPC Automation interfaces. These interfaces are provided for compatibility and may be removed in a future release.

Subtopics:

- Installing OPC Automation
- Submitting NetView Commands from a Batch Job
- The Batch Command Interface
- Using OPC Special Resources
- The Structure of OPC Request Automation
- Automating Applications with OPC Automation
- MESSAGES/USER DATA Entries and USER E-T Pairs for OPC Automation
- OPC Automation Common Routines and Data Areas
- Guidelines for User-Written Operations
- OPC Automation Operator Commands
- Resynchronization and Recovery Considerations

Chapter 7. Installing OPC Automation

This chapter describes the steps to follow when installing.

Enabling and Disabling OPC Automation

SA z/OS OPC Automation may be disabled and enabled by specification of subsystems with an application type of OPC.

To disable SA z/OS OPC Automation, do not specify any OPC applications in the Policy Database for the System that is to have OPC Automation disabled. Disabling occurs on a system by system basis, so by not linking OPC type applications to a system, automatically disables OPC Automation.

Disabling OPC Automation causes the message traps in the SA z/OS NetView to also be disabled. This will speed message processing for those systems that do not participate in OPC functions. Disabling OPC Automation does not prevent execution of the INGOPC command. As long as at least one system in the sysplex contains a Controller or Tracker, the INGOPC command will work.

If you disable SA z/OS OPC Automation for any reason, be sure to unlink the OPC Command Receiver NON-MVS subsystem and the OPC Request Receiver NON-MVS subsystem from systems for which OPC Automation is disabled.

To enable SA z/OS OPC Automation, specify the OPC applications for the Controllers and Trackers in the Policy Database of all systems that have either Controllers and Trackers running on them. Include any Trackers that belong to foreign Controllers. That is Controllers not present anywhere in the sysplex that the Trackers belong to.

Defining System Automation Policy

Several automation policy items are required for correct operation of OPC Automation. These policy items are:

- The Automation Operators that are required for function enablement.
- The required non-MVS subsystem that is the PPI request server. This subsystem provides support for the compatible execution of requests from OPC to SA z/OS.
- The optional new non-MVS subsystem that defines the PPI batch command interface server.
- The definition of OPC Controller, Tracker and Server subsystems.
- The definition of WORKSTATION names to be automatically activated on System Automation for z/OS Agent startup.
- Definition of the status observer subsystem to ensure that SA z/OS status changes are reflected in OPC special resource statuses.

Define SA z/OS Automation Operators

The Automation Operators that are required for correct operation of SA z/OS OPC Automation are listed in Table 5.

Table 5. Automation Operators

Automation Operator	Description	Messages
OPCAMSTR	Main Automation Operator, required to enable SA z/OS OPC Automation	EVJ*
OPCAOPR2	OPC Request execution operator	none
OPCACMDR (new)	OPC Batch Command Execution operator	none

These automation operator definitions can be found in the *SYSPLEX sample PDB definitions under the Auto Operators policy with the name ″OPC_AUTO_OPS″.

Define Optional Workstations

The batch jobs that execute NetView and SA z/OS commands may be submitted by any OPC Computer/Automated Workstation. However, if the NetView PPI receiver is not operational at the time of execution of the batch job, the batch job may optionally set the workstation that submitted it to an inoperative state. If this function is to be used, which is the default for the batch job, then it may be prudent to create additional batch job submission workstations to which these NetView-related batch jobs are assigned. This will allow the rest of the batch job stream to be submitted, whilst holding the NetView-related jobs until NetView starts.

The Workstations that are automatically re-enabled at SA z/OS startup are those defined on the WORKSTATION User message policy for either the Trackers or Controllers defined to SA z/OS. An example of this is shown in Figure 26 on page 55, which shows the definition of workstation N001 against subsystem OPCF, where:

- CODE1 represents the name of the sysplex that the Batch Command Server non-MVS subsystem is running on.
- CODE2 represents the name of the system that the Batch Command Server non-MVS subsystem is running on.
- CODE3 is not used.

This allows the same Controller or Tracker subsystem definition to be run on different systems or sysplexes and the name of the workstation can be different on each sysplex or system combination.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing             Row 1 to 21 of 21
Command ===>                                                  SCROLL===> PAGE

Entry Type  : Application           PolicyDB Name   : SA22_KEYPLEX
Entry Name  : TWS_V810_CONTROLLER   Enterprise Name : KEY1FAMILY

Subsystem  :  OPCF
Message ID  : WORKSTATION

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

Code 1          Code 2          Code 3          Value Returned
*               *               *               N001



****************************** Bottom of data ********************************



 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 26. Defining Workstation User Message Policy

Non-MVS Subsystem Definition for the OPC Request Server

This non-MVS subsystem is required to allow requests from Tivoli Workload Scheduler to System Automation for z/OS. This interface is provided for compatibility with the previous method of handling requests from OPC to SA z/OS.

For new users of System Automation for z/OS V2.2, the sample PDB *SYSPLEX contains a subsystem definition TWS_REQUEST_SERVER. This subsystem definition contains the policy definition for the OPC Request Server.

For existing users of SA z/OS, a sample part EVJCFPPI has been supplied. Please note that this part contains definitions for both the Request Receiver and the Command Receiver, so that this action will only need to be done once. Migrate this part into a dummy SYSTEM on the existing PDBs. This will automatically create the TWS_REQUEST_SERVER subsystem. Link this subsystem to an application group that will connect it to each system that runs a System Automation for z/OS Agent. The recommendation for this group is that it be linked to each system that is defined in the PDB and that it be a SYSTEM type group. This will allow the shutdown of each system to operate independently from other systems.

No relationship definitions are required for this subsystem.

Non-MVS subsystem definition for the OPC Command Server

This non-MVS subsystem is required to allow commands from the new Batch Command Interface to System Automation for z/OS.

For new users of System Automation for z/OS V2.2, the sample PDB *SYSPLEX contains a subsystem definition TWS_COMMAND_SERVER. This subsystem definition contains the policy definition for the OPC Command Server.

For existing users of SA z/OS, a sample part EVJCFPPI has been supplied. Please note that this part contains definitions for both the Request Receiver and the Command Receiver, so that this action will only need to be done once. Migrate this part into a dummy SYSTEM on the existing PDBs. This will automatically create the TWS_COMMAND_SERVER subsystem. Link this subsystem to an application group that will connect it to each system that runs an System Automation for z/OS Agent. The recommendation for this group is that it be linked to each system that is defined in the PDB and that it be a SYSTEM type group. This will allow the shutdown of each system to operate independently from other systems.

No relationship definitions are required for this subsystem.

Define Workstation Domain Entries

Customize the WORKSTATION DOMAINS (ODM entry type) policy objects in the SA z/OS policy database and connect them to systems in the WORKSTATION DOMAINS policy item as required. These policy objects map OPC workstations to NetView domain IDs.

Define Controller Details

Customize the CONTROLLER DETAILS (OCS entry type) policy objects in the SA z/OS policy database and connect them to systems in the CONTROLLER DETAILS policy item as required. These objects specify the location of a controller and can be associated with a set of OPC special resources. See *System Automation for z/OS Defining Automation Policy* for more information.

Define System Details

Customize the OPC SYSTEM DETAILS policy objects (OEN entry type) in the SA z/OS policy database and connect them to systems in the OPC SYSTEM DETAILS policy item as required. These objects contain control information for OPC Automation. See *System Automation for z/OS Defining Automation Policy* for more details.

Define Special Resources Policy

If required, define OPC special resources as OPC SPECIAL RESOURCES policy objects (OSR entry type) in the SA z/OS policy database and link them to CONTROLLER DETAILS objects. See *System Automation for z/OS Defining Automation Policy* for more information.

Define or Modify Subsystem Messages/User Data

You must define each subsystem you wish to automate from OPC to SA z/OS. In many cases, you will also have to define OPCA and OPCACMD entries in the MESSAGES/USER DATA policy item for these subsystems (see "Executing OPC Requests with OPC Automation" on page 96).

Define SDF Statuses

The following SDF Statuses are required for correct operation of DFCRIT and DFUPDT.

```
EVENTR          PRIORITY=50 COLOR=R HIGHLIGHT=N CLEAR=(Y,RV*)
```

```
EVENTP      PRIORITY=150 COLOR=P HIGHLIGHT=N CLEAR=(Y,RV*)
EVENTY      PRIORITY=250 COLOR=Y HIGHLIGHT=N CLEAR=(Y,RV*)
EVENTW      PRIORITY=350 COLOR=W HIGHLIGHT=N CLEAR=(Y,RV*)
EVENTC      PRIORITY=450 COLOR=T HIGHLIGHT=N CLEAR=(Y,RV*)
EVENTL      PRIORITY=550 COLOR=T HIGHLIGHT=R CLEAR=(Y,RV*)
EVENTG      PRIORITY=650 COLOR=G HIGHLIGHT=N CLEAR=(Y,RV*)
EVENTD      PRIORITY=750 COLOR=G CLEAR=(Y,RV*) REQ=NOADD
```

These statuses may need to be defined for back level releases of System Automation for z/OS for both targets and focal points.

Defining the SA z/OS Status Observer

SA z/OS OPC Automation provides a facility that echoes the status of SA z/OS resources in OPC Special Resources. This facility allows the user to define OPC operations that will wait until SA z/OS resources reach a desired state. Currently only two desired states are allowed. The UP state is when the Automation Manager sets the resource to the AVAILABLE state. The DOWN state is when the Automation Manager sets the resource to the UNAVAILABLE state.

The Status observer is implemented as an Automation Agent function. This function, registers with the Automation Manager and receives status changes. It then translates the status changes to the appropriate OPC special resources and issues a MVS subsystem broadcast to all OPC Controllers and Trackers on the system that the Agent is running on.

The SA z/OS Status Observer is defined as a non-mvs SA z/OS subsystem. This subsystem should run on the system that contains the OPC Controller or alternatively an OPC Tracker. Figure 27 shows the relationships required for the definition of the OPC Observer.

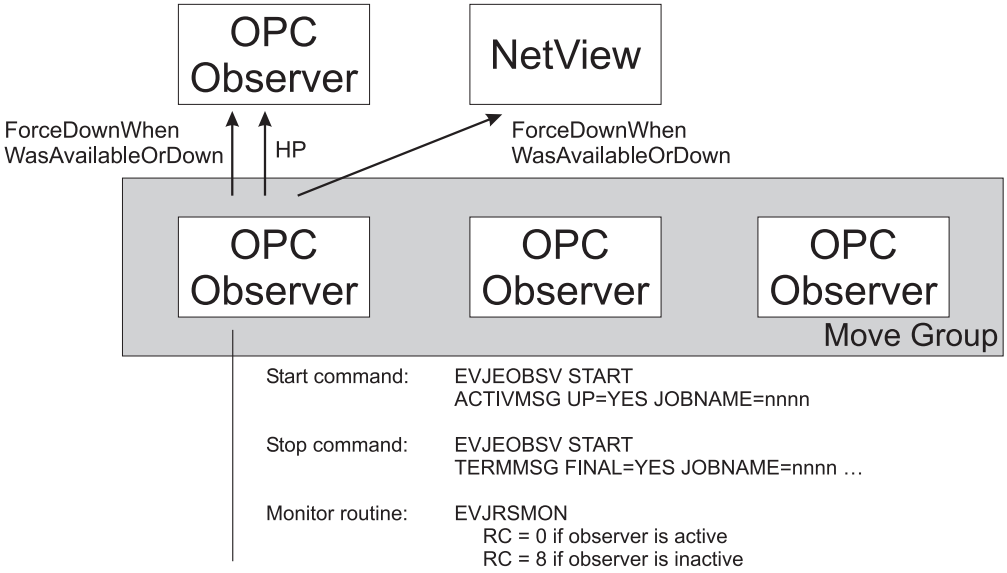


Figure 27. OPC Observer Relationships

Status Observer Definitions

The Application definitions for the OPC Status Observer are as in Table 6:

Table 6. OPC Status Observer Application Policy Definitions

Policy Item	Value(s)
Application Type	STANDARD
Job Name	OPCOBSVR
Job Type	NONMVS
Monitor Routine	EVJRSMON
Periodic Interval	00:10:00 or a suitable interval
Relationships	• forceDown(WhenObservedWasAvailable) to supporting resource sanetview/APL/= • forceDown(WhenObservedHardDown) to supporting resource opcController/APL/= • hasParent to supporting resource opcController/APL/=
Startup	NORM: EVJEOBSV START ACTIVMSG JOBNAME=&SUBSJOB,UP=YES
Shutdown	NORMAL: EVJEOBSV STOP TERMMSG JOBNAME=&SUBSJOB,FINAL=YES

The relationships are defined to ensure that the observer subsystem is forced to autodown if either the OPC Controller or the NetView containing the observer subsystem are unavailable. This will force the OPC Observer resource to unavailable and will allow a move group to select a working OPC Observer to start.

In a single system, these relationships will simply have the effect of cleaning up the status of the OPC Observer subsystem.

For a sysplex, you should define a sysplex move group and put the OPC Observer subsystem in it. Connect the move group to all the systems in the sysplex that have an OPC Controller defined to them. Set the preferences to ensure that the appropriate system is selected that matches the Active OPC Controller. The above definitions will ensure that the OPC Observer will be moved to a new system if either the system, NetView or OPC Controller currently active fails. The definitions will not select a system where the new active OPC Controller resides. It will merely select the next active system based on preferences. You can manually adjust the preferences to move the OPC Observer to the new Active OPC Controller system if required.

Chapter 8. Submitting NetView Commands from a Batch Job

This chapter will describe how to execute NetView commands from a Batch job. This is particularly useful for Tivoli Workload Scheduler, but can be used stand alone without either Tivoli Workload Scheduler or Operations Planning for Control.

Subtopics:

- Sample Batch Job JCL
- Command Statement Syntax
- Valid Command Types
- Command Continuation
- Command Output Re-Direction
- Executing a Command on a Different NetView

Sample Batch Job JCL

A sample batch job can be found in the System Automation for z/OS Installation library SINGSAMP. Member EVJSJ001 contains the sample JCL. The batch job must be run on the same system as the SA z/OS Agent that contains the Command receiver specified by the batch job. In most cases there will be a Command receiver running on every SA z/OS Agent . However, customization of the Command receivers can alter the names of the Command receivers and also the number and configuration of the Command receivers. You should check with your system programmers to determine the correct system and command receiver to use for these batch jobs.

Command Statement Syntax

The commands supplied to the batch job in the //SYSIN ddname have the following syntax:

1. All blank lines are ignored.
2. All lines starting with an asterisk (*) are comment lines and are printed in the output but otherwise ignored.
3. Comments on the end of commands are not allowed.
4. Comments are not allowed between continuation lines.
5. A command can be continued by appending a "-" (dash) to the line.
6. Command output normally goes to //SYSTSPRT.
7. Command output may be redirected to other DDNAMEs via the ">" (right angle bracket) symbol.
8. PIPE > stage is prohibited. Use PIPE QSAM instead.
9. Full Screen commands are not allowed.

Valid Command Types

Any command, clist or Rexx program that issues correlated line messages may be used.

This means almost all NetView commands, all SA z/OS commands that support OUTMODE=LINE and any clist or Rexx program that either issues SAY messages or PIPES the messages to CONSOLE.

The return code from the command can be used to stop the remaining commands from being executed. See the HIGHRC= parameter of the EVJRYCMD procedure definition in Part 3, "Programmer's Reference," on page 51.

Command Continuation

Commands are continued across lines by appending a dash to the end of the command, for example:

```
PIPE NETVIEW LIST STATUS=OPS | -
CONSOLE ONLY
```

Command Output Re-Direction

Normally command output is printed on the //SYSTSPRT DDNAME. However, the output of commands may be re-directed to other DDNAMEs. This is achieved via the ″>″ symbol, for example:

```
PIPE NETVIEW LIST STATUS=OPS | CONSOLE ONLY >MYOUTPUT
```

This allows subsequent steps in the batch job or other batch jobs to use the output of the command for their own purpose.

The DCB characteristics of the output DDNAME should be as follows:

```
LRECL=132,RECFM=FB
```

Executing a Command on a Different NetView

Almost all SA z/OS commands can specify the TARGET= parameter to force the command to execute on the target system. If a command does not have this facility, for example the NetView LIST command, you can use PIPE labels to send the command to the appropriate NetView, for example:

```
PIPE CC dom01: LIST STATUS | CONSOLE ONLY
```

or even

```
PIPE CC dom01/auto1: LIST STATUS=OPS | CONSOLE ONLY
```

Chapter 9. The Batch Command Interface

This chapter describes the OPC batch command interface.

JCL for the Batch Command Interface

Figure 28 on page 62 shows the sample from the product sample library (SINGSAMP).

```
//EVJSJ011 JOB (ACCT),'SAMPLE OPC/TWS JOB'
//**START OF COPYRIGHT NOTICE*****************************************
//*                                                                   *
//* PROPRIETARY STATEMENT:                                            *
//*                                                                   *
//*      5645-006                                                     *
//*      LICENSED MATERIALS - PROPERTY OF IBM                         *
//*      (C) COPYRIGHT IBM CORP. 2002 ALL RIGHTS RESERVED.            *
//*                                                                   *
//*      US GOVERNMENT USERS RESTRICTED RIGHTS -                      *
//*      USE, DUPLICATION OR DISCLOSURE RESTRICTED BY                 *
//*      GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                     *
//*                                                                   *
//*      STATUS= JKYS203                                              *
//*                                                                   *
//**END OF COPYRIGHT NOTICE*******************************************
//*                                                                   *
//*  DESCRIPTION: OPC/TWS SAMPLE BATCH JOB TO EXECUTE COMMMANDS ON    *
//*               SYSTEM AUTOMATION NETVIEWS                          *
//*                                                                   *
//*  CHANGE CODE VRSN DATE    WHO  DESCRIPTION                       *
//*  ----------- ---- ------- ---- ----------------------------------*
//*  $L0=FEATURE,SA22,23JAN02,APC(JJK): INITIAL VERSION               *
//*  $L1=MISOPC ,SA23,02MAR04,APC(JJK): ADD NETVIEW V5 COMMENTS       *
//*********************************************************************
//*-------------------------------------------------------------------***
//S0       EXEC PGM=IEFBR14
//CONCAT   DD  DSN=TEMP.CONCAT.LIST,
//             DISP=(MOD,DELETE,DELETE),
//             UNIT=SYSDA,SPACE=(1,1),AVGREC=M,
//             LRECL=132,RECFM=FB,STORCLAS=SMS
//*-------------------------------------------------------------------***
//*%OPC SCAN
//S1       EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4M,
//         PARM='EVJRYCMD &OWSID SERVER=EVJCMDRV HIGHRC=16'
//STEPLIB  DD  DSN=SYS1.NETV.V130.SEKGLNK1,DISP=SHR   NETVIEW LIBRARY
//*TEPLIB  DD  DSN=SYS1.NETV.V510.SCNMLNKN,DISP=SHR   NETVIEW V5 LIB
//SYSPROC  DD  DSN=SYS1.SAM.V220.SINGNREX,DISP=SHR    SA LIBRARY
//EQQMLIB  DD  DSN=SYS1.TWS.V810.SEQQMSG0,DISP=SHR    TWS LIBRARY
//OUTPUT   DD  SYSOUT=*
//CONCAT   DD  DSN=TEMP.CONCAT.LIST,
//             DISP=(MOD,CATLG),
//             UNIT=SYSDA,SPACE=(1,1),AVGREC=M,
//             LRECL=132,RECFM=FB,STORCLAS=SMS
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  DUMMY
//SYSIN    DD  *
* THIS IS A COMMENT
MVS D A,L >OUTPUT
D NET,MAJNODES >CONCAT

PIPE NETV WHO | -
     NLOC /AUT/ | -
     CONS ONLY
INGLIST */APL/* OUTMODE=LINE >CONCAT
/*
```

Figure 28. Sample JCL for the Batch Command Interface

Please ensure that the appropriate NetView library is assigned to //STEPLIB. This library should contain the DSIPHONE module.

S0 This Step deletes a temporary listing data set that is used for concatenation of output from the command.

%OPC This statement is used to tell OPC to begin scanning for OPC substitution variables and to replace them when found with their contents.

S1 This Step executes a BATCH TSO TMP to run the Rexx command that sends commands to a SA z/OS Agent.

EVJRYCMD This is the Command that runs to process batch commands.

&OWSID This is a Tivoli Workload Scheduler substitution variable that represents the name of the workstation that submitted the job.

//STEPLIB The NetView library SEKGLNK1 (or equivalent) is used to provide the DSIPHONE Rexx function. Please substitute the correct library name for the appropriate release of NetView that is being used by SA z/OS.

If Module EQQYCOM cannot be found in the system LINK LIST, then concatenate the appropriate TWS Load library here.

//SYSPROC The SA z/OS library that contains the EVJRYCMD Rexx procedure.

//EQQMLIB The Tivoli Workload Scheduler for z/OS message library.

//OUTPUT Optional output DDname for command output redirection.

//CONCAT Optional output DDname for command output redirection. In this case a data set that will have successive command output concatenated to it.

//SYSTSPRT Required TSO TMP output DDNAME.

//SYSTSIN Required TSO TMP command input DDNAME. Dummied out because the only command to be executed is in the parameter to the TSO TMP.

//SYSIN Commands to be executed in the SA z/OS Agent.

EVJRYCMD Description

Purpose

The EVJRYCMD is a Rexx procedure that will issue commands to a SA z/OS Agent and will receive the results of those commands.

Parameters

wsid This is a required parameter.

This parameter specifies the name of the Tivoli Workload Scheduler work station that submitted this batch job. This information is used by the command to disable the work station in the event that communications between the batch job and the SA z/OS Agent cannot be established. See "the NOWKSTS parameter" on page 64 to modify this behavior. This parameter must be specified, even if it is not to be used. In the case of the NOWKSTS parameter being specified or the batch job is

submitted manually or by a product other than Tivoli Workload Scheduler, specify any non-blank character sequence.

NOWKSTS This parameter is optional.

This parameter modifies the behavior of the command. In the event of a failure in communications to the SA z/OS Agent, this parameter prevents the command from disabling the Tivoli Workload Scheduler work station as defined by the wsid parameter. If this parameter is not specified, any NetView PPI communications problem will cause the command to issue a TWS WSSTAT command to place the work station offline.

SERVER= This parameter is optional.

The default for this parameter is EVJCMDRV. This parameter specifies the name of the PPI receiver in the SA z/OS Agent NetView to which commands will be sent.

TIMEOUT= This parameter is optional.

The default for this parameter is 60 seconds.

This parameter specifies the time in seconds that the batch job will wait for a command to execute in the System Automation Agent NetView. This timeout is applied separately to each command.

HIGHRC= This parameter is optional.

The default for this parameter is 0 (zero).

This parameter specifies the highest acceptable Return Code for the job. Any return codes from commands that are less then or equal to this value will reset the JCL Step return code to zero. Any command return code that is greater than this value will be passed as the JCL Step return code.

Note: The JCL Step return code will be the highest return code of all the command return codes.

Usage

When the SA z/OS Agent is started, it will automatically issue a WSSTAT command to mark the work station online. The specifications of which work stations to mark online at agent restart is contained in the WORKSTATION message/user data policy for the tracker or controller. Multiple work stations may be defined. Work stations that are assigned to trackers should have their WORKSTATION policy defined to the same trackers that they are assigned to. See "Define Optional Workstations" on page 54.

Each command is submitted in turn and the results of the command are retrieved. These results are then written to either SYSTSPRT or to the output redirection DDNAME.

Chapter 10. Using OPC Special Resources

This chapter describes the OPC Special Resources created by SA z/OS and how to use them.

SA z/OS can dynamically create OPC special resources based on the status of SA z/OS resources. These OPC special resources can in turn be used to control the flow of applications and operations in OPC.

OPC Special Resource Definition

SA z/OS creates OPC Special Resources if allowed to via policy definitions. The definition of the name of these special resources is as defined in "OPC Automation Special Resources" on page 14.

Each SA z/OS Resource (application, application group, system or system group) has two OPC special resources. One tracks the state of the SA z/OS resource in the UP case. The other tracks the state of the SA z/OS resource in the DOWN case. Setting the availability of these two OPC special resources are independent of each other. It is possible for the SA z/OS resource to have both the UP and DOWN OPC special resources UNAVAILABLE. This can occur when the SA z/OS resource is starting for example. It is neither UP or DOWN.

It should not normally be possible to have both OPC special resources AVAILABLE at the same time.

Enabling SA z/OS OPC Special Resources

To enable the SA z/OS OPC Special Resource tracking the following have to be done.

1. Ensure that the SA z/OS Resources that are to be monitored are defined SA z/OS.

   All these policy items are described in *System Automation for z/OS Defining Automation Policy* in the section "Defining Automation for OPC Components":

   a. Set the OPCA PCS Special Resources Policy.

      Use NO if you do not want any Special Resources Set.

      Use ALL if you want ALL SA z/OS Resources echoed as OPC Special Resources (this will create a large number of OPC special resources).

      Use YES if you want a selection of SA z/OS Resources echoed as OPC Special Resources.

   b. If YES was specified above, ensure that the OPC Special Resources policy item is updated with the appropriate resource masks and linked to the appropriate systems via the WHERE USED entry.
2. Ensure that the definitions for the OPC Status Observer are entered into the Automation Policy. The instructions for this are given in "Defining the SA z/OS Status Observer" on page 57.
3. Ensure that OPC option RESOPTS DYNAMICADD(YES) is specified.

Using SA z/OS OPC Special Resources in an Application

Holding an Operation until an SA z/OS Resource Reaches a Desired State

To use a SA z/OS special resource to hold an operation until the appropriate status is achieved do the following:

1. Create the special resource in the OPC Database.

   Use the OPC ISPF dialog to create the special resource. Set the Availability of the special resource to N, as shown in Figure 29.

```
------------------------- CREATING A SPECIAL RESOURCE ------------------------
Option ===>

Select one of the following:

1 INTERVALS  - Specify intervals
2 WS         - Modify default connected workstations

SPECIAL RESOURCE    ===> ING.KEY1.APL.CICSK1G.UP_____________________
TEXT                ===> ________________________________________________
SPECRES GROUP ID    ===> ________
Hiperbatch          ===> N DLF object Y or N
USED FOR            ===> B Planning and control C , P , B or N
ON ERROR            ===> __ On error action F , FS , FX , K or blank



Defaults
  QUANTITY          ===> 1_____ Number available 1-999999
  AVAILABLE         ===> N Available Y or N


F1=HELP       F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
F7=UP         F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 29. Creating a Special Resource

2. Create the Operations in the application:

   The first operator should submit the batch job to execute the requested function.

   The second operation runs at a General Completion Workstation and waits for the appropriate Special Resource, as shown in Figure 30 on page 67.

```
--------------------------------- OPERATIONS ----------------- Row 1 to 2 of 2
Command ===>                                                  Scroll ===> CSR

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details, J - Edit JCL
Enter the TEXT command above to include operation text in this list, or,
enter the GRAPH command to view the list graphically.

Application            : JKOPCTST1        Test Batch Iface

Row  Oper      Duration  Job name  Internal predecessors           Morepreds
cmd  ws   no.  HH.MM.SS                                          -IntExt-
'''' N001 001  00.00.01  SAMPJOB   ___ ___ ___ ___ ___ ___ ___ ___   0 0
'''' GAC1 005  00.05.00  ________  001 ___ ___ ___ ___ ___ ___ ___   0 0
****************************** Bottom of data ********************************




F1=HELP       F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
F7=UP         F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 30. Creating the Operations

3. The Special resource for operation 005 is as shown in Figure 31.

```
------------------------------ SPECIAL RESOURCES ------------- Row 1 to 1 of 1
Command ===>                                                  Scroll ===> CSR

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete

Operation              : GAC1 005

Row  Special                                      Qty    Shr Keep On
cmd  Resource                                            Ex  Error
'''' ING.KEY1.APL.CICSK1G.UP_____________________ 1_____ S   _
****************************** Bottom of data ********************************









F1=HELP       F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
F7=UP         F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 31. Special Resource for Operation 005

4. Make sure you define the Special Resource to OPC with an initial Availability of N.

In this case Operation Number 005 will wait until SA z/OS sets the special resource ING.KEY1.APL.CICSK1G.UP to Y. This will only happen if the observed and desired status of CICSK1G/APL/KEY1 are both AVAILABLE.

Starting or Stopping an SA z/OS Resource

To start or stop a SA z/OS Resource, use the previous section as a base.

1. In the JCL for job SAMPJOB copy the sample job EVJSJ001 from SINGSAMP and tailor it for your environment.

2. Put the INGREQ command in the //SYSIN and specify the OUTMODE=LINE option for INGREQ.
3. If you are starting a resource (APL or APG) update the Special Resource for the operation 005 to reflect the name of the resource specified in the INGREQ command. e.g. for INGREQ TEST/APG/SYS1 use special resource ING.SYS1.APG.TEST.UP.
4. If you are stopping a resource (APL or APG) update the Special Resource for the operation 005 to reflect the name of the resource specified in the INGREQ command. e.g. for INGREQ TEST/APG/SYS1 use special resource ING.SYS1.APG.TEST.DOWN.
5. Remember to create the special resource in the OPC Database via option 1.6 in the OPC dialogs.

The operation is now ready for LTP and Current Plan planning functions.

When OPC submits the first operation, the JCL will run the Batch Command Interface and execute the INGREQ command on the SA z/OS Agent. SA z/OS will then process the command and issue the appropriate orders to the Agents to achieve the desired status. When the resource has achieved the desired status, SA z/OS will notify OPC that this has occurred by updating the special resource. This will cause operation number 005 to be marked Complete - especially if you assign it to a General Non-Reporting work station. Operations and Applications that have a predecessor of operation number 005 will now be able to run.

Chapter 11. The Structure of OPC Request Automation

This chapter explains the structure of OPC request automation in some detail.

Flow Overview

OPC Automation is an interface between NetView, OPC, and SA z/OS. These components provide the facilities which make up the interface. This section provides an introduction to these components and their interactions.

Initialization

Initialization involves the following two sequences:

1. Initialization of the OPC components.
2. Initialization of OPC Automation functions in each NetView. "Startup of OPC-Controlled Subsystems" on page 78 describes this. OPC Automation initialization includes the automated recovery sequences described in "Automated Recovery" on page 87.

Request Flow

This section contains a detailed description of the flow of a request from OPC to NetView and the return confirmation. This flow provides an explanation of the involved modules. "Request and Confirmation Transaction Flow" on page 79 summarizes this request.

Figure 32 on page 70 uses a request to start RMF, located in a NetView domain NVREG with a workstation definition of NV04. This request is an operation in an OPC-defined application known as MAINT.

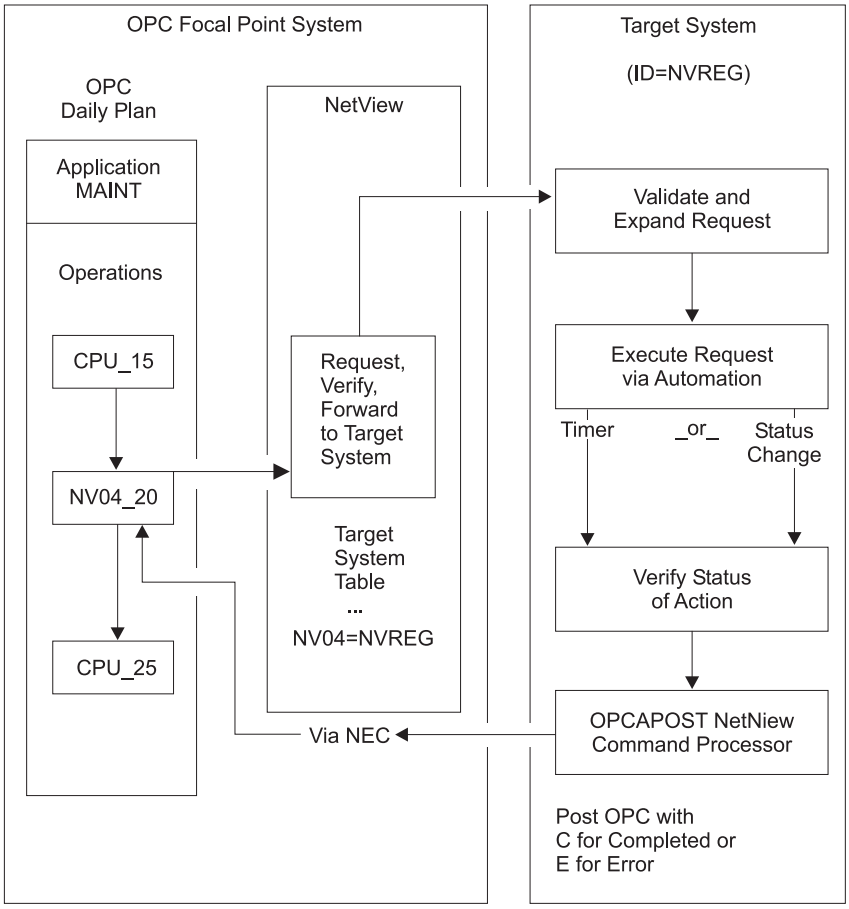


Figure 32. NetView-OPC Interface Flow. Syntax and definition errors, target system availability, recovery, and resynchronization via OPC API and NetView program-to-program interface are not shown in this example.

Using dependency control to ensure an orderly flow of operations, OPC defines the OPC-controlled application named MAINT. OPC defines the application on an automatic general workstation, specifying the NetView to which the request is sent. NV*xx* specifies a NetView automatic general workstation with a NetView domain index of *xx*, which is resolved in the Controller NetView into the target NetView domain ID through the definitions in the SA z/OS policy database. OPC can define the NV*xx* workstation with all regular specifications, such as parallel servers and special resources.

In the MAINT example in Figure 32, OPC defines the last batch application processed before starting RMF with an operation number of 15. Once this completes properly, the normal OPC dependency control readies the NV04_20 operation on the NV04 workstation. This signifies that the request contained within the operation description field is sent to the NetView with a domain ID of NVREG.

OPC Automation uses the NetView PPI to transfer the request from OPC to NetView. This transfer is through the EQQUX007 exit in the OPC/ESA controller.

EQQUX007 Exit

Each change of status on any workstation causes OPC to call user exit 7 (EQQUX007). The OPC Automation user exit EVJUX007 calls modules EVJ07001 and EVJ07004.

- EVJ07001 sends automation commands and data across the NetView PPI for all status changes on NV*xx* workstations.
- EVJ07004 WTO's Operation status information for any operation that ends in Error or is changed from Error state to any other OPC state. This information is used to update SDF and NMC monitoring of OPC operations.

Figure 33 shows the flow of the EVJ07001 exit.

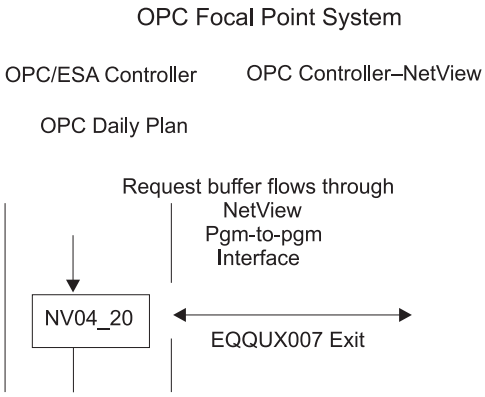


Figure 33. EQQUX007 Exit

When an NV*xx* workstation moves to the R status, the workstation generates a request buffer. Fields pointed to by registers in the EQQUX007 exit provide all of the data for the request buffer. For the layout of the fields in the request buffer, see Table 9 on page 133 and Table 10 on page 133.

OPC Automation supplied EQQUX007 exit logic verifies that all fields exist (except the optional request parameter fields). If this exit logic determines that any field is missing or the value is not valid, it issues an error WTO and changes the operation to E status, with an error code indicating a user-definition error. Since the EQQUX007 exit contains no capability to directly change the status of an OPC operation when an error code is posted to OPC, the EQQUX007 exit uses the EQQUSINT module to respond.

If the information is correct, OPC builds the request buffer and calls the CNMCNETV module, which is the NetView program-to-program interface module. This module transfers the request to the Controller-NetView, where OPC verifies the return codes from the call function to ensure that there are no errors. If OPC detects errors, the EQQUSINT module changes the status to E (ended-in-error), with the error code on the basis of the PPI module return code. The module issues a WTO and completes processing the EQQUX007 logic. OPC Automation then restores registers and returns control to OPC.

If the OPC Automation EQQUX007 exit is unable to load the CNMCNETV module or use it to send data, it directs OPC to mark the requested operation in error, with an error code of UNTV. OPC Automation will attempt to reset operations which have ended in a UNTV error, subject to a user-defined time limitation, whenever the OPC controller is restarted.

Program-to-Program (PPI) Interface Dispatcher

The NetView program-to-program interface passes the request buffer to the PPI dispatcher task in the SA z/OS application. The PPI dispatcher task (EVJTOPPI), a

NetView subtask, receives the requests for an SA z/OS action from the buffers from the EQQUX007 exit. Figure 34 shows this flow.

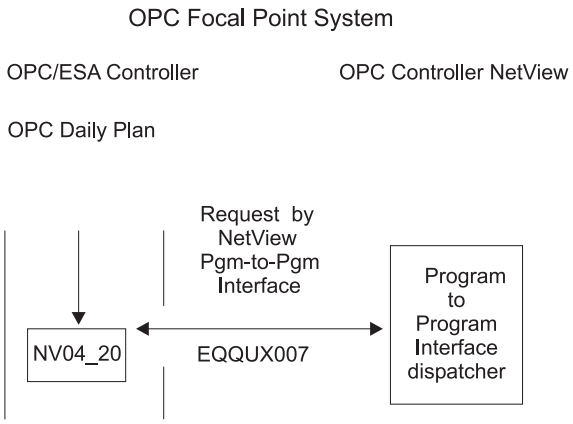


Figure 34. PPI Dispatcher

On the basis of the sending task identifier, the PPI dispatcher determines the function in SA z/OS that is sent. For OPC Automation, the dispatcher selects the verify function.

Verify Module (EVJESPVY)

The verify module, which runs on a NetView autotask, runs only in the Controller NetView. This module receives the action request buffer from the PPI dispatcher task. Figure 35 shows this process.

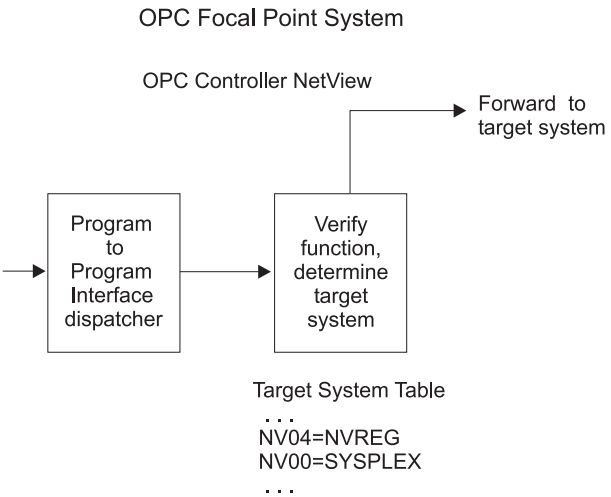


Figure 35. Verify Module

The verify module uses the NV*xx* index to obtain the destination NetView domain ID from the SA z/OS policy database. If the relevant NV*xx* index specifies SYSPLEX, then all SA z/OS systems in the local sysplex are queried for the status of the application associated with the job name of the request. The destination is determined to be the system which has the application in the most active state.

If the destination NetView and the requesting NetView are the same, OPC Automation logs the request buffer and invokes the request module. If the

destination NetView and the requesting NetView are different, OPC Automation sends the request to the proper NetView domain by message forwarding.

If OPC Automation does not find the NV*xx* index then OPC Automation issues a message, posts the operation status to E (ended-in-error, U003), and logs the results. No communications can occur with this workstation until the definition is corrected. On the domain where the OPC controller is running, the workstation must be defined in the WORKSTATION DOMAINS policy object (ODM entry type). You must manually reset operations that are posted-in-error since OPC Automation carries out no automated recovery for definition errors.

If NV*xx* is associated with the sysplex on which the OPC controller is running (SYSPLEX keyword in the WORKSTATIONS DOMAIN entry) and OPC Automation does not find the job defined to any online SA z/OS in the local sysplex, then OPC Automation issues a message, posts the operation status to E (ended-in-error, S998), and logs the results. To cater for the situation where all domains where the job runs are offline, the operation will be retried if a gateway connection to another SA z/OS becomes active.

If OPC Automation successfully forwards messages, it logs the request buffer and returns control to the module. If OPC Automation cannot send the request, it issues an error message and logs it to indicate communication loss with the requested NetView domain. OPC Automation then posts the operation status to E (ended-in-error, S999) due to loss of contact. When OPC Automation re-establishes communications with this NetView domain, it checks for all outstanding errors because of loss of communications on this workstation. If OPC Automation finds any of these errors, it resets the OPC-operation status to R (ready), which re-invokes the EQQUX007 exit.

Request Module (EVJESPRQ)

The arrival of a request from the verify module drives the request module in the Tracker NetView. OPC Automation installs the request module on each system running an OPC/ESA Tracker. Figure 36 on page 74 shows the flow of this process.

The main function of the request module is to translate the OPC-generated request into a subsystem related command, or to schedule a user-defined function that is not related to a subsystem. The control flow of the module is shown in Figure 36 on page 74.

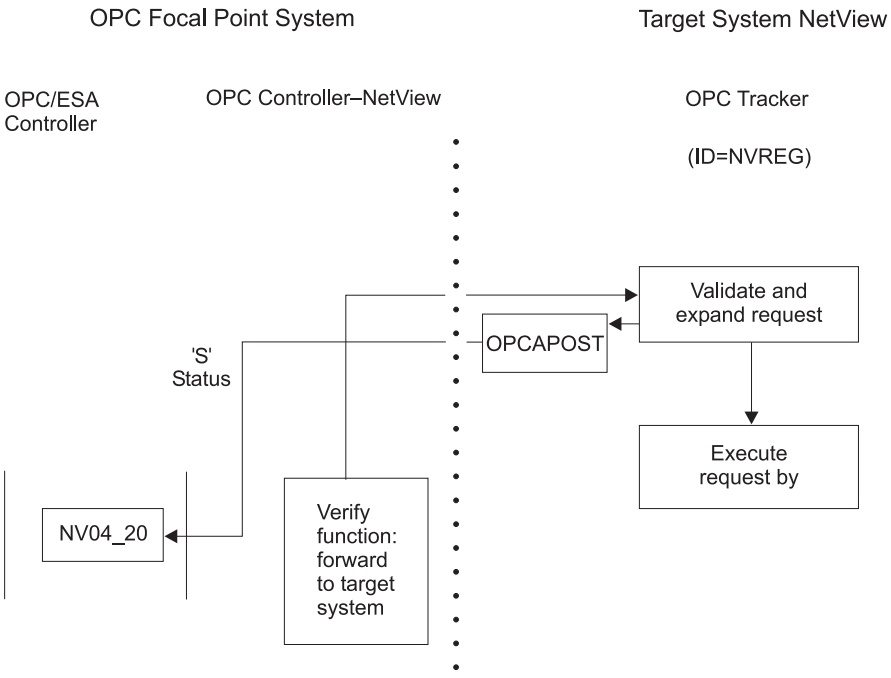


Figure 36. Request Module

If required, the request module uses definitions in the SA z/OS policy database to create the command that initiates the function requested. The policy database contains entries (OPCA, OPCACMD keywords; see Chapter 13, "MESSAGES/USER DATA Entries and USER E-T Pairs for OPC Automation," on page 103) from which the command text and the parameter syntax for the actual request are obtained. If any of these entries are not found, the processing cannot continue. The OPCAPOST module posts an error to OPC, which logs the error and issues a WTO. Since this is a user-definition error, OPC attempts no automation recovery. The user must correct the definitions and reset the operations in error.

In Figure 36, the request module translates the requested action in the buffer to the SA z/OS command required to start RMF. The SA z/OS command then starts RMF.

Except for starting, stopping, or recycling SA z/OS-controlled subsystems, other functions may require user programming. To support these functions, OPC Automation provides a user exit capability. For a detailed description of user responsibilities required to handle a user call, see Chapter 15, "Guidelines for User-Written Operations," on page 135.

For subsystem-related operations, the OPCAPOST command processor posts to OPC if the required entries are found in the policy database. OPC changes the status from R (ready) to S (started). OPC Automation then issues a timer request on the basis of the delay specified in the policy database (OPCA keyword, see "OPCA" on page 109), issues the command, and checks the return code. Then the request module terminates.

A change of subsystem status calls the status-change exit module. If the status change does not occur, the timer-driven module executes when the timer interval expires. This ensures that a request resulting in an unexpected status processes. For

example, if OPC requests a START operation, and the subsystem fails to start due to a JCL error or other problem, then the OPCAPOST module posts OPC with an error status.

OPC Automation dynamically generates the OPC request by using definitions in the policy database and dynamic substitution of command fragments on the basis of the parameters.

Status Change Module (EVJESPSC)

SA z/OS calls the status-change module for each change of status. This module determines whether a status change is the result of a previous OPC Automation request. If the status change is not the result of a previous request, OPC Automation ignores the status change. Figure 37 shows the flow of this process.

If an outstanding request for the changed subsystem exists, and the new status is compliant with the expected status, OPC Automation cancels the timer. OPCAPOST updates the OPC operation status to C (completed) status.

With the timer values properly set and the operation processing normally, the change of status should always occur before the timer interval expires.

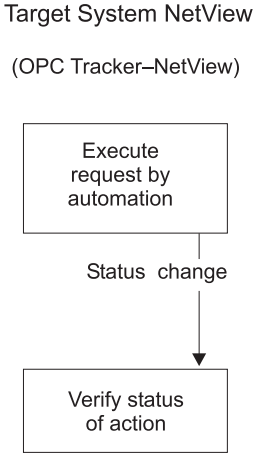


Figure 37. Status Change Module

Timer Module (EVJESPTE)

Under normal conditions, a request passed to SA z/OS results in the desired status change before the timer expires, and OPC Automation purges the timer. When this sequence does not occur, and the timer remains at the end of the timer interval, SA z/OS drives the timer module.

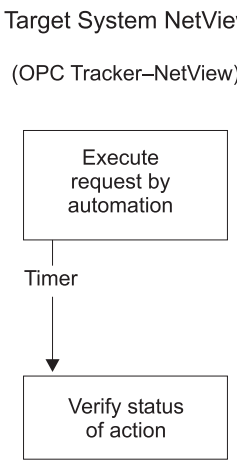


Figure 38. Timer Module

For subsystem related functions, the SA z/OS status file provides the current status, and EVJESPTE compares this with the expected status. If a match is obtained, the OPCAPOST command processor posts a C (completed) status to OPC. If EVJESPTE determines a mismatch between the current and expected status, OPCAPOST posts an error to OPC for review by the OPC administrator. Figure 38 shows the flow of this process.

OPCAPOST Command Processor

The OPCAPOST command processor calls EQQUSINT, which passes the completion code to the OPC Tracker in this system. The OPC Tracker forwards the completion code to the system running the Controller through the mechanism used by OPC/ESA. Figure 39 shows the flow of this process.

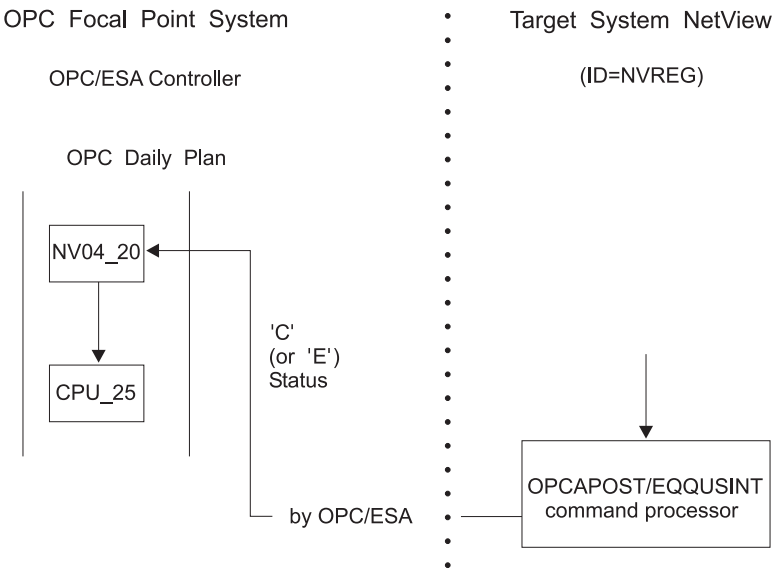


Figure 39. OPCAPOST Command Processor

Other functions use the OPCAPOST command. See “OPCAPOST” on page 130 for documentation on the syntax.

This module completes the processing for this specific OPC operation. If the request executes successfully, OPC Automation sets the OPC operation status to C

(completed) and normal OPC-dependency control allows the next operation to start. See the CPU_25 batch job in Figure 39 on page 76. If the operation completes in error, OPC Automation sets an E status and a 4-character return code. The application does not continue processing until some intervention occurs. An operator or OPC Automation's recovery can sometimes provide this intervention.

Automated Operator Tasks

OPC is an SA z/OS-controlled subsystem. Normal definitions in the SA z/OS policy database can describe OPC. In addition, SA z/OS defines an automated operator task (called *automated function* by SA z/OS) for the OPC Controller in the system containing the Controller, as well as one for the OPC Tracker in each system. These automated operator tasks perform the OPC-requested functions in the SA z/OS application.

OPC Automation requires actions in a specific order. Changes in this order can result in unpredictable and undesirable results. To ensure that a proper sequence of processing is maintained, you must complete the actions in a single-thread fashion. In OPC, this is the responsibility of the user and is achieved through dependency control or critical resource specifications.

NetView maintains this control by ensuring that actions are executed sequentially through the use of automated operator tasks. Specify only one automated operator task for the OPC Controller functions and only one for the Tracker functions. Stipulating any additional automated operator tasks for OPC Automation results in loss of synchronization. This, in turn, can create an uncontrolled environment, requiring a substantial amount of operator/system programmer effort to recover, and additional loss of synchronization until a single automated operator task for the Tracker and Controller functions is reinstated. When OPC Automation detects any violations, it checks for out-of-sequence requests and stops processing for a specific application through an error code to OPC Automation.

However, separate automated operator tasks for Controller and Tracker are required. Running OPC Automation on the Controller system with a single automated operator task specified for both Controller and Tracker functions results in a lockout condition. Consider this especially on backup systems, which do not normally run Controller functions. If you specify only one automated operator task for both systems, each task runs properly until they become an active backup system and lock.

For the automated operator task OPCAMSTR, the operator ID must be AUTOPCP. For the automated operator task OPCAOPR2, you may specify whatever operator ID meets your installation standards. However, do not change the OPC Automation operator task names OPCAMSTR and OPCAOPR2.

Initialization

SA z/OS initialization involves two phases:

- The first starts OPC components so the scheduling process is active.
- The second restores the status of any OPC-controlled tasks to the last status requested by OPC and waits for OPC to issue new requests.

Startup of OPC Components

The first phase involves the initialization of the OPC components. In normal mode of operations, OPC remains operational at all times. Without the SA z/OS

application, OPC starts as a JES task. With OPC Automation, the responsibility of starting OPC transfers from JES to the SA z/OS application. Figure 40 shows an example of the startup of OPC/ESA during an IPL process.

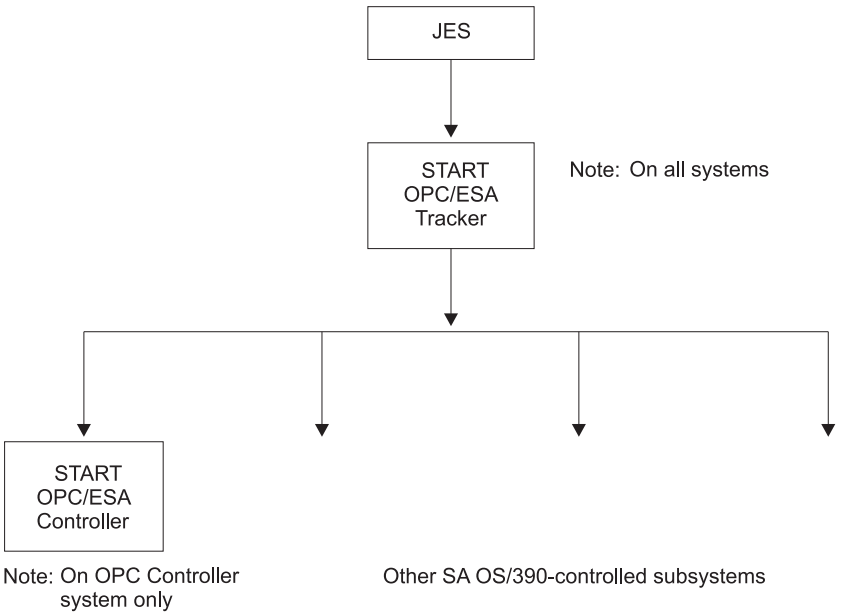


Figure 40. OPC/ESA Startup During IPL Process

The following scenario describes this type of environment:

- The OPC/ESA Tracker has JES as a parent. A small portion of OPC/ESA starts before JES. During system IPL, the master scheduler invokes this program (EQQUNIT).
- During the IPL process, JES issues a start command for the OPC/ESA Tracker task as soon as it is running as part of the normal SA z/OS-controlled flow.
- Once OPC/ESA Tracker starts, the SA z/OS application issues a start command for the OPC/ESA controller on the control host only.
- The SA z/OS application continues to initialize the rest of the tasks that are defined to it.

This completes the initialization phase.

Startup of OPC-Controlled Subsystems

After the SA z/OS application has completed initializing its defined tasks, the startup phase of the OPC-controlled subsystems starts.

OPC Automation uses a status file record for each subsystem defined to it. This record keeps information such as the last completed action, any request in progress, or the last processed request if no request is processing. The status file record provides a means of maintaining this information across NetView failures and restarts.

During the initialization of OPC Automation, its initialization module runs. This module carries out several functions that result in every OPC Automation subsystem resynchronizing to a known status. The initialization module also sets OPC Automation status-record-locking flags to a null value.

During OPC Automation startup, OPC Automation examines the SA z/OS database for OPC Automation entries. If new entries are found, OPC Automation creates status file records and initializes them to a null value (never a used status). OPC Automation attempts no action for these subsystems until it receives a request from OPC. This allows coding entries into the policy database before defining the subsystems in the rest of SA z/OS or OPC. For existing OPC Automation status file entries, OPC Automation resets the timer and completion flags to a null value, which allows handling of new requests.

On the system where the OPC Controller runs, OPC Automation initialization takes an additional step. This step drives the automated recovery function and determines whether OPC has any requests which ended-in-error (S999 or UNTV) because of the unavailability of NetView. If any ended-in-error requests are found, OPC Automation resets the operations.

Initialization Module (EVJESPIN)

The initialization module carries out two functions.

- OPC Automation uses the first function during initialization as previously described.
- An operator command accesses the second. This function also builds and resynchronizes OPC Automation status file records dynamically. For a description of the uses of the initialization command, refer to "EVJESPIN — Initialization" on page 149.

Request and Confirmation Transaction Flow

Figure 41 on page 80 shows the flow from an OPC application requested action through to NetView and the return confirmation of the action. This example illustrates the request to start the resource management facility (RMF™), located in a remote host with a NetView domain identifier of NVREG. OPC contains a representation of this host with a workstation definition of NV04. The request to start RMF is part of an OPC application known as MAINT. In Figure 41 on page 80, the jobname is specified as RMF and the operation text is START.

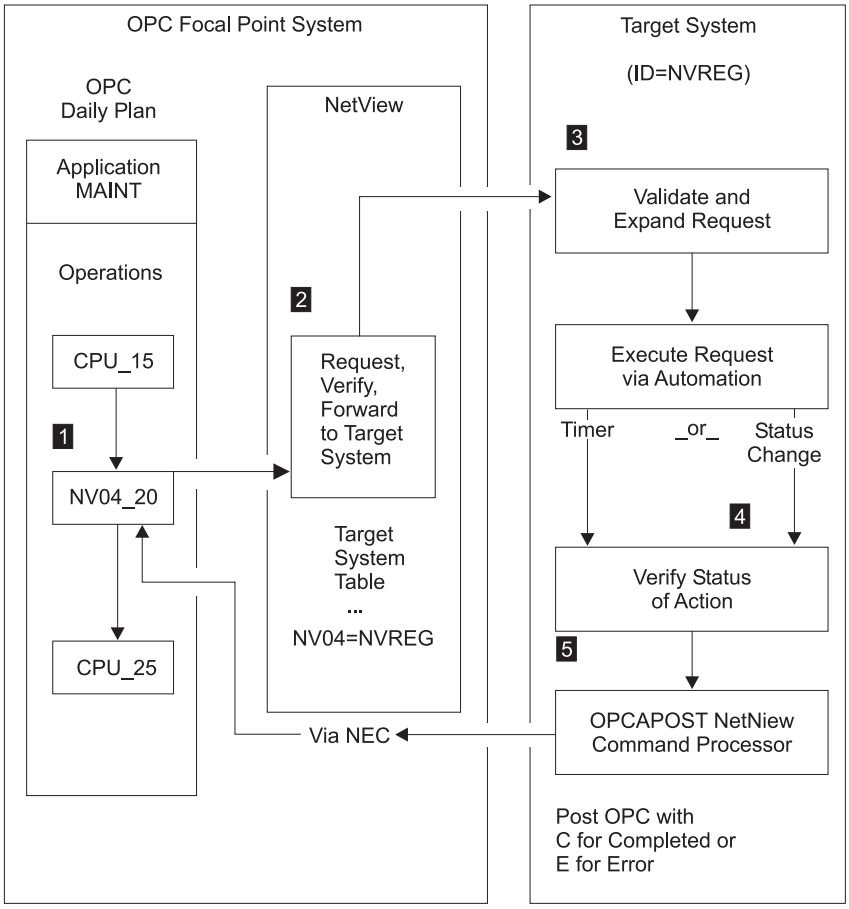


Figure 41. NetView–OPC Interface Flow. Syntax and definition errors, target system availability, recovery, and resynchronization via OPC API and NetView PPI are not shown in this example.

The OPC application named MAINT is defined to OPC, using dependency control, to ensure an orderly flow of operations. NV04 defines an OPC automatic general workstation which is resolved by NetView into the target NetView domain ID through OPC Automation parameter definitions.

Figure 41 shows CPU_15 as the last batch step which needs processing prior to starting RMF. Once this completes properly, OPC dependency control makes the NV04_20 operation ready on the NV04 workstation. This causes the creation of a request buffer for the request to start RMF which is then forwarded to NetView Domain NVREG. For the request buffer, see "Request Buffers and OPC Automation Log Entries" on page 81.

OPC Automation uses the NetView PPI to transfer the request buffer from OPC to NetView. This transfer of the request from OPC to NetView is through the use of the status change exit (exit 7) in OPC.

1 The NetView PPI passes the request buffer to the PPI dispatcher task in SA z/OS. This task dispatches the request to the OPC Automation verify routine, which translates the workstation name into the NetView domain ID through definitions in the SA z/OS policy database.

2 The request is forwarded to the appropriate NetView domain for execution.

3 The target NetView translates the request text into MVS console commands using information stored in the SA z/OS policy database. In Figure 41 on page 80, the request module translates the request buffer into a command to start RMF. This start command must be specified with the OPCACMD keyword in the MESSAGES/USER DATA policy item of the RMF application (for details, see "OPC Requests and MESSAGES/USER DATA Keywords" on page 96). It can be an MVS START command, or the INGREQ SA z/OS command (which is recommended). In the latter case, the command could be:

```
INGREQ RMF REQ=START,TYPE=NORM,SOURCE=EXTERNAL
```

Functions other than START and STOP of subsystems based on SA z/OS may require user programming.

The command is dynamically generated using definitions in the SA z/OS policy database. A check determines whether the command is properly accepted. During this process, WTOs and the OPCAPOST command report errors. OPCAPOST sends an error indication back to OPC. If the command is issued correctly, a timer request is made. The timer intercepts a condition, where the request does not execute in a reasonable amount of time, which is user-selectable.

4 A change-of-status SA z/OS function intercepts all changes-of-status. This allows the completion of outstanding requests as soon as the request is executed.

5 When the request is completed, the OPCAPOST command processor is invoked. OPCAPOST calls EQQUSINT which passes the completion code to the OPC Tracker on this system. The OPC Tracker forwards the completion code to the OPC Controller.

In a user-supported function, the timer and completion validation are a user responsibility. Once the user code determines that the function is completed, it should call the OPCACOMP function. This function assures that actions are accomplished in the correct sequence, performs some housekeeping, returns a good or bad completion code, and calls the OPCAPOST command processor.

This terminates the processing for this specific OPC operation. If the request is executed without problems, the operations status is set to C (completed), and normal OPC dependency control allows the next operation to start. See the CPU_25 batch job in Figure 41 on page 80.

If the operation completes in error, an E status and a 4-character return code is set, and the application does not continue processing until a person or OPC intervenes.

Errors reset by OPC Automation are the result of regained availability of a target NetView domain to which communications are lost. The error codes are set to:

- U*xxx* when human intervention is required.
- S*xxx* when OPC Automation attempts to recover. This occurs when an operation that did not complete properly is resolved and completed.

Request Buffers and OPC Automation Log Entries

Requests are specified to OPC Automation in the **Operation text** field of the **Operations** panel (see "Defining Applications for OPC Automation in OPC" on page 90 for details). The EQQUX007 exit logic creates a request buffer that contains the request and all additional information that is needed

- To pass the request to its proper destination, and

- To inform the requesting workstation of the result of the request.

Before processing takes place, the request buffers received by OPC Automation from OPC are copied to the NetView log for tracking purposes, such as verification of correct operations and error logging.

Figure 42 shows the request buffer log entry for the example of Figure 41 on page 80, if we suppose that an OPC controller running on domain NVDOM has made the request to OPC Automation to perform START for the RMF subsystem. When this action is completed, OPC Automation changes the status of the NV04_20 operation in the MAINT application to C (completed).

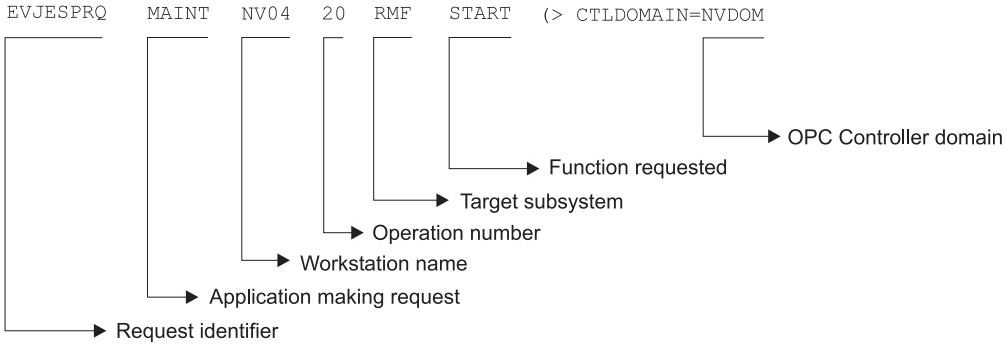


Figure 42. NetView Log Entry of an OPC Generated Request

The operation number is needed in order to inform OPC Automation which operation in OPC must be notified of the result of the request. In the example, the request contains no parameters. When a request does contain parameters they are inserted in the log entry between the request type (function) and the domain name of the requesting OPC controller. For more information on request parameters, see "Request Parameters and the &EHKVAR*i* Variables" on page 98.

Request Handling in the OPC Controller System

In an OPC-controlled application, defining specific parameters for an operation generates a request. These parameters are defined for an operation on an NV*xx* workstation, where NV*xx* represents a NetView domain. When the daily planned execution of OPC makes this NV*xx* workstation ready, OPC Automation starts through the EQQUX007 OPC/ESA user exit. See Figure 43 on page 83 for an illustration of this flow.

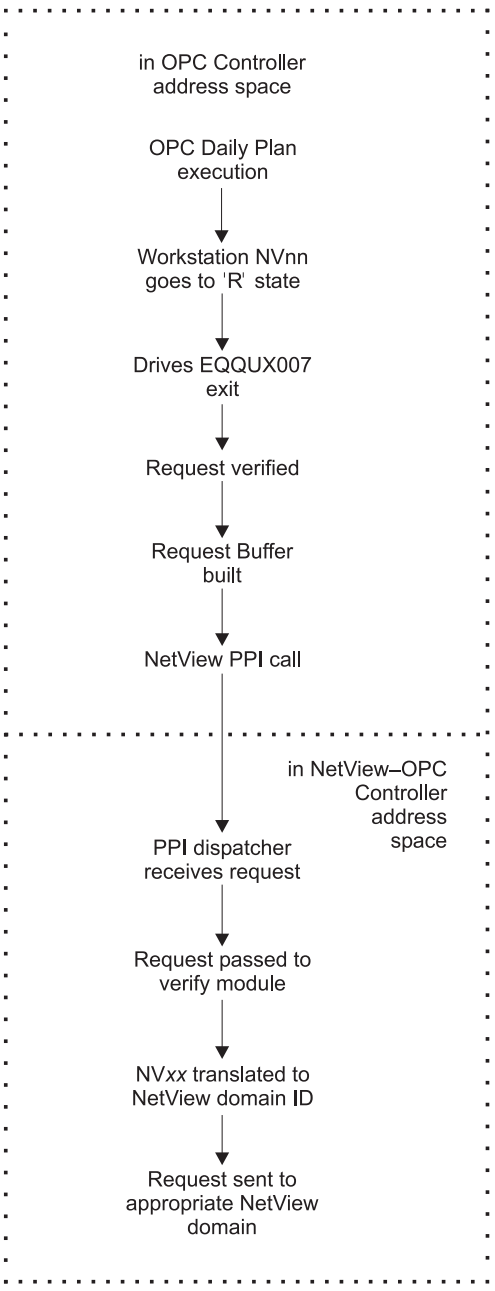


Figure 43. Request Handling in the OPC Controller Processor

Each OPC change of status drives the OPC/ESA EQQUX007 exit. OPC Automation logic in the exit intercepts a change of status to R (ready) for an NV*xx* workstation. OPC Automation logic in the exit intercepts a change of status and verifies the request for required fields, correct lengths, and appropriate use of alphanumerics. However, OPC Automation does not validate fields for specific content. From the information in the OPC control blocks, OPC Automation logic builds a request buffer to identify the request, application, and workstation in OPC.

Once OPC Automation builds this request buffer, the PPI calls NetView. The PPI dispatcher in NetView receives the request buffer and sends it to the appropriate module. In this case, it is the OPC Automation verify module, which runs under the NetView Controller automated-operator task.

If OPC Automation is unable to call the NetView interface module, it marks the operation with a status of E and an error code of UNTV. OPC Automation will tell OPC to reset operations which have ended with a UNTV error every time OPC or SA z/OS is restarted, provided these errors occurred less than a user-specified time interval. This time interval is defined in the **Operation reset delay** field of the OPC SYSTEM DETAILS policy object; see *System Automation for z/OS Defining Automation Policy*.

The verify module translates the NV*xx* identifier to a real NetView domain ID, and sends the request buffer to the appropriate domain through forwarding functions of SA z/OS. If the request is destined for the same system as the one that the OPC Controller is on, OPC Automation transfers the request to the request module running under the NetView Tracker automated operator task.

Request Handling in the OPC Tracker System

Figure 44 describes the flow of OPC Automation for a request invoking a base function of the SA z/OS-defined subsystem.

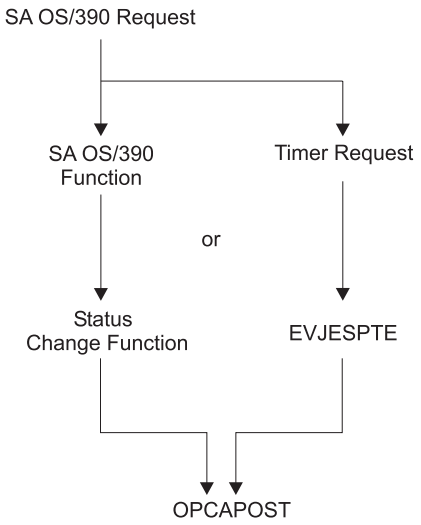


Figure 44. Request Flow for a Base SA z/OS Function

The request module (EVJESPRQ), a part of the Tracker portion of OPC Automation, runs under the NetView Tracker automated operator task. This module issues the command associated with the request that is contained in the request buffer. The OPCACMD entry for each subsystem (see "OPC Requests and MESSAGES/USER DATA Keywords" on page 96) defines the actual command. This permits the coding of generic requests in the application descriptions in OPC. This request flow also allows customizing requests for each system and avoids changes in the target systems reflected in the focal-point system.

The user is responsible for controlling requests and not issuing multiple requests to the same subsystem on a given target. Use dependency control or critical resource definitions in OPC to control the sequencing of requests.

EVJESPRQ uses a timer and completion flag as a locking mechanism to ensure that there is only one outstanding request for a subsystem at a given time. If OPC Automation receives a request before completing a previous request, OPC Automation posts the operation to OPC with a return code of U005, indicating a user sequence error. You must then use a manual recovery function to synchronize

OPC and OPC Automation. The initialization command (EVJESPIN) contains the following two parameters for this purpose:
- RESET
- SYNC

The request module sets a timer for every command issued. This ensures that no lockout condition occurs. The timer value specified in the OPCA entry of the policy database should be large enough to accommodate the longest interval of time that the requested function may take under normal operating conditions.

The occurrence of the subsystem status change invokes the status change module (EVJESPSC). If the module completes before the expiration of the timer, this avoids delaying the process. With a properly set timer delay, the status change function should always gain control before the expiration of the timer. When the status change module gains control, the module cancels the timer if it is still outstanding. If the timer interval expires before the timer is cancelled, OPC Automation logs this event and indicates performance or other problems.

The timer module (EVJESPTE) and the status change module update the OPC Automation status file record and use the OPCAPOST command processor to post the appropriate status and return code to OPC.

Completion and Timer Flags

Both the status change and timer modules check for each other's completion. If one function completes first, the second function exits to avoid false double posting to OPC. Double posting is avoided by using timer and completion flags, as the following text discusses.

NetView schedules work on automated operator tasks on a first-in/first-out basis. Work elements resulting from the status change or timer modules become ready as NetView schedules them on the queue. Since NetView automated operator tasks process in a sequential manner, the queue can hold both the status change and the timer work elements at the same time. When this occurs, NetView must process only one work element, because handling both results in double posting to OPC, leading to errors in the OPC-defined application.

To avoid these errors, OPC Automation uses two flags in the automation status file entry. One of these flags is related to the status change module, the other to the timer module. When the modules are called, they examine the flags. If neither flag is on, the respective module turns on the flag associated with the active function and continues processing. If a flag is found on, the function exits, because the processing is completed by the other function while this work element was queued. This process depends on defining a single automated operator task for each NetView Tracker.

Operations Control

This section discusses the EVJESPIN module and obtaining information from OPC. "EVJESPIN — Initialization" on page 149 describes operator commands and their actions.

EVJESPIN Module

This EVJESPIN module, which is also used as a command, provides two separate capabilities:
- Creating an OPC Automation status file record

- Resynchronizing or unlocking a given subsystem

In certain situations, usually because of user definition or sequence errors, the timer and completion flags prevent OPC Automation from accepting any new requests for a subsystem on a target NetView. This flow ensures that errors are caught, actions of OPC Automation for the given subsystem are halted, and creation of additional problems is avoided before the original error is corrected. For example, if a subsystem startup request is in operation, then it is not prudent to process a shutdown request in the same interval. By not accepting any requests after an error is detected, the information in OPC Automation status file record then reflects the request that caused the problem. This should simplify problem determination.

Two correction methods are provided.

RESET This method resets the timer and completion flags to null. OPC Automation takes no other action since the flags are reset. OPC Automation then accepts new requests for this subsystem. Restarting the application at an appropriate point can control recovery from OPC/ESA.

SYNC The OPC Automation status file record contains the status of the subsystem that was the result of the last successful request; this status is either UP or DOWN. With the SYNCH option, EVJESPIN tries to synchronize the actual status of the subsystem with this information. When both are at variance, OPC Automation issues a SA z/OS startup or shutdown command to change the status of the subsystem to match that in the status file record. The SYNC function does not post to OPC on completion of the SA z/OS function. OPC's requests for the subsystem synchronize SA z/OS.

If you have defined a new subsystem to SA z/OS since last initializing NetView, you can create a single OPC Automation status file record. The (CREATE) option creates an OPC Automation status file record dynamically for the specified subsystem. OPC Automation initializes the record to a null condition, so that the subsystem can accept OPC Automation commands.

OPC Automation provides no automated function to remove an old OPC Automation status file record.

Obtaining Information from OPC

The application program interface of OPC/ESA allows OPC Automation to act directly on the OPC current plan. Using this interface, OPC Automation directly requests and updates OPC-based information.

Recovery operations provide one possible use of the OPC API by OPC Automation. For example, if a communications link to a target NetView is not available, the operator can use the OPCACMD command to manually post events that have occurred.

You can also use this interface when manual intervention is required, for example, when a user sequence error is detected. Use the OPCACMD operator command to manually access the information from OPC about operations in the current plan through the OPC API. This allows the determination and resolution of the sequence error to occur from a single NetView console.

The host with the OPC Controller task provides the only availability to the OPC API interface. Because of this, all recovery is done only in the NetView on the processor running the OPC Controller task. Sometimes, when a user-written module utilizes this function, another NetView requires the information. OPC Automation maintains an entry in the policy database to allow OPC Automation to determine the domain ID of the OPC NetView to which the request is sent (CONTROLLER DETAILS policy object, OCS entry type). This entry has the domain ID of the NetView on the processor running the OPC Controller. In this manner, all the SA z/OS applications can easily determine where to direct OPC Controller requests.

A user-written task or CLIST can also access the OPC Controller. Your own routines can list operations in the current plan with the OPCALIST command. You can modify the data in current plan operations with OPCAMOD. You can query the OPC calendar with OPCACAL. You can synchronize OPC with OPCACOMP, or OPCAPOST when you write your own automation routines, and you can update the status of special resources in OPC with OPCSRST or with the SRSTAT CLIST. This way you can trigger operations to run which have a special resource dependency in OPC. For more information, see

- "OPCALIST" on page 125,
- "OPCAMOD" on page 127,
- "OPCACAL" on page 121,
- "OPCACOMP" on page 124,
- "OPCAPOST" on page 130,
- "OPCSRST" on page 131.

Automated Recovery

OPC Automation provides an automated recovery function for requests that could not reach their destination because of connectivity problems, or because the NetView PPI was unavailable. Whenever a request fails because a connectivity problem exists, OPC Automation posts OPC with an error status and a return code of S999, S998 or UNTV.

Whenever the OPC Automation initiates a request and NetView or its program-to-program interface (PPI) is down or unavailable, OPC Automation posts OPC with an error status and a return code of UNTV.

When an OPC workstation (NV*xx*) is associated with the sysplex of the OPC controller (SYSPLEX keyword) but a search of all online systems in the local sysplex cannot find a definition for the supplied job name then it is assumed that the job runs on a sysplex member which is not up. OPC Automation posts OPC with a status of E (error) and a return code of S998.

When a required NNT connection is not available OPC Automation posts OPC with a status of E (error) and a return code of S999.

If the operator resets these operations, OPC Automation does not attempt any recovery. If the error code is not changed, OPC Automation invokes the operation again when connectivity is re-established, or when the OPC Controller is restarted.

OPC Automation calls the automated recovery function as part of the initialization of OPC Automation in a domain where the OPC Controller runs. OPC Automation

also invokes this function for S999 or S998 errors whenever an NNT link (automation gateway) is re-established to a domain where the OPC Controller runs.

When either of these two conditions occur, OPC Automation uses the OPC API function to obtain a list of all operations ended-in-error for the appropriate NV*xx* workstation or workstations. OPC Automation scans this list to find error codes starting with S. If any codes of this type are found, OPC Automation issues OPCAPOST for that operation with an X (reset) status. This resets the operation to the R (ready) status and re-invokes the EQQUX007 exit. OPC Automation attempts no other recovery.

Chapter 12. Automating Applications with OPC Automation

This chapter explains how to set up OPC Automation in OPC and in SA z/OS.

Defining Automated OPC Applications

This section describes what you need to do when you define an application in OPC that you can automate using OPC Automation.

Note: This section contains some specific details about how to define applications and other items to OPC. However, it does not explain basic OPC functions because it assumes that you have a prerequisite knowledge of OPC and will refer to the OPC documentation when necessary.

Defining Information for OPC Automation in OPC

The information that is passed to OPC Automation is entered in standard OPC description fields. This information is used to route requests where they are verified and executed. When the request is completed, a status change for the operation is sent back to OPC. The minimum information that needs transferring to accomplish this is listed in Table 7.

Table 7. OPC Automation Items Defined in OPC

Definition in OPC	Information item	Refer to
General reporting workstations that represent target NetView domains	OPC workstation ID representing the NetView domain ID	"Defining the Target NetView Domains" on page 89
OPC-defined application making requests to OPC Automation	Application name	Application field in Figure 46 on page 91
Target subsystem, such as RMF, for which this function is executed	Job name	Job name field in Figure 46 on page 91
Operations executed within a job	Operation number or numbers	No. field in Figure 46 on page 91
Request and request parameters to be performed for the specific operation	A request, such as STOP or START, and optional parameters	Operation text field in Figure 47 on page 91 and Figure 48 on page 92

Defining the Target NetView Domains

To define an OPC request that OPC Automation can use, a workstation representing the target NetView domain is required. This workstation, which is defined with OPC using the standard OPC dialogs, should be a general, automatic reporting workstation. Its name *must* have the format

`NV`*`xx`*

Note: Reserve NV*xx* workstations for OPC Automation. Unpredictable and undesirable results may occur if these workstation names are used for other workstations.

Figure 45 shows a typical definition.

```
--------------------- BROWSING A WORK STATION DESCRIPTION ---------------------
 Command ===>

 Enter the command R for resources or A for availability above.

 Work station          : NV00
 Description           : NetView AOFS6 for test

 Work station type     : General
 Reporting attribute   : Automatic
 Printout routing      : SYSPRINT
 Control on servers    : No

 Splittable            : No
 Job setup             : No
 Sub/Rel data set      :

 Transport time        :  0:00
 Duration              :

 Last updated by       : COLEY    on 05/08/95 at 09:54
```

Figure 45. Sample NVxx Workstation Definition in OPC

Entries in the SA z/OS policy database (WORKSTATION DOMAINS policy object, entry type ODM) translate NV*xx* to an actual NetView domain ID. The automation programmer defines these entries. In this manner, the scheduler defining the OPC applications does not need to know the NetView domain ID names, but rather works with the workstation representation of those names. This allows changes to the relationship of workstations to NetView domain IDs without modifying the OPC definitions.

Note: You may use OPC database management dialogs or batch loader jobs to define the NV*xx* workstations.

Defining Applications for OPC Automation in OPC

Standard OPC application description panels are used to define applications that put requests to OPC Automation. The following items are defined:

- Application making the request
- Function requested

Application Making the Request: The application making the request is defined to OPC with operations specified on the NV*xx* workstation, as shown in Figure 46 on page 91.

```
--------------------------------- OPERATIONS ---------------------  ROW 1 OF 2
 Command ===>                                                  Scroll ===> PAGE

 Enter/Change data in the rows, and/or enter any of the following
 row commands:
 I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
 S - Select operation details
 Enter the PRED command above to include predecessors in this list, or,
 enter the GRAPH command to view the list graphically.

 Application            : MAINT            Test for maint appl
 Row  Oper      Duration Job name  Operation text
 cmd  ws   no.  HH.MM
 '''' NV04 005   0.01    RMF_____  STOP____________________
      |    |                |
      |    |                |
      |    |                |
      |    |                +--------> The subsystem
      |    +-------------------------> The operation
      +------------------------------> The NetView Domain ID workstation
```

Figure 46. Defining the MAINT Application in OPC

Function Requested: The function requested and the request parameters, if any, are not standard OPC definitions. Enter these fields in the **Operation text** field using blanks as delimiters.

Figure 47 shows an example of how to define a request to stop and start RMF.

```
--------------------------------- OPERATIONS ---------------------  ROW 1 OF 2
 Command ===>                                                  Scroll ===> PAGE

 Enter/Change data in the rows, and/or enter any of the following
 row commands:
 I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
 S - Select operation details
 Enter the PRED command above to include predecessors in this list, or,
 enter the GRAPH command to view the list graphically.

 Application            : MAINT            Test for maint appl

 Row  Oper      Duration Job name  Operation text
 cmd  ws   no.  HH.MM
 '''' NV04 005   0.01    RMF_____  STOP____________________
 '''' NV04 010   0.01    RMF_____  CANCEL__________________
```

Figure 47. 'Operations' OPC Panel Showing OPC Automation Requests

OPC Automation permits the inclusion of two optional parameters in the request buffer. The target system builds the required command with these parameters, using information contained in the SA z/OS policy database. Alternatively, the parameters pass control information to optional user-written modules. Figure 48 on page 92 shows an example of the request using these optional parameters.

```
--------------------------------- OPERATIONS ---------------------  ROW 1 OF 2
Command ===>                                                    Scroll ===> PAGE

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details
Enter the PRED command above to include predecessors in this list, or,
enter the GRAPH command to view the list graphically.

Application            : FORCE            Test for maint appl

Row  Oper      Duration Job name  Operation text
cmd  ws   no.  HH.MM
'''' NV04 005   0.01    RMF_____  STOP FORCE IMM__________
```

Figure 48. Request Using Optional Parameters

Displaying OPC Automation Requests in OPC

Since OPC Automation requests are stored as operation text, you can view them in OPC, as shown in Figure 49.

```
----------------------------- BROWSING OPERATIONS ----------------  ROW 1 OF 2
Command ===>                                                    Scroll ===> PAGE

Enter the PRED command above to include predecessors in this list, or,
enter the GRAPH command above to view operations graphically.
Enter the row command S to select the details of an operation.

Application            : RMFBKUP           RMF Backup Processing

Row  Oper      Duration Job name  Operation text
cmd  ws   no.  time
'    NV04 005   0:01    RMF       STOP
'    NV04 010   0:01    RMF       START
****************************** BOTTOM OF DATA ********************************
```

Figure 49. Browsing Operations Including OPC Automation Requests

The following list defines several of the fields that are shown on the panel in Figure 49.

NV04 — Represents the target NetView domain or the sysplex the request is sent to.

RMFBKUP — The application submitting the request to OPC Automation.

RMF — Target subsystem. This looks like a job to OPC, but is actually a subsystem.

005 and 010 — Standard operation sequence numbers used by OPC.

STOP and START — Requested function. There are no parameters in this example.

Example of an Application Making a Request

This section provides an example of how an application that puts a request to OPC Automation is defined in OPC.

The application name is MAINT. This application consists of three operations:

- Stop RMF on the target system
- Schedule a batch job
- Restart RMF on successful completion of the batch job

Figure 50 shows the initial panel in the application creation process.

```
--------------------------- CREATING AN APPLICATION --------------------------
 Command ===> oper

 Enter/Change data below:
 Enter the RUN command above to select run cycles or enter the OPER command
 to select operations.

 Application id         : MAINT
 Valid from - to        : 95/04/17 - 99/12/31

 APPLICATION TEXT   ===> RMF maintenance_________
                                  Descriptive text of application
 Owner:
  ID                ===> NSC02___________
  TEXT              ===> ____________________
                                  Descriptive text of application owner
 PRIORITY           ===> 5           A digit 1 to 9 , 1=low, 8=high, 9=urgent
 VALID FROM         ===> 95/04/17    Date in the format YY/MM/DD
 STATUS             ===> A           A - Active, P - Pending
 AUTHORITY GROUP ID ===> ________    Authorization group ID
 CALENDAR ID        ===> ________________
                                  For calculation of work and free day
```

Figure 50. RMF Maintenance Application Primary Panel in OPC

In Figure 50, certain fields, such as calendar ID, are not used. However, OPC Automation does not preclude the use of normal application and operation functions.

Selecting OPER as a primary command allows the entry of the individual operations for this application.

In Figure 51, three operations are defined. The first and third send requests to OPC Automation in the NetView domain that is associated with the NV00 workstation. The second is a batch job named RMFMAINT that performs the batch maintenance tasks.

```
-------------------------------- OPERATIONS --------------------  ROW 1 OF 2
 Command ===> text                                            Scroll ===> PAGE

 Enter/Change data in the rows, and/or enter any of the following
 row commands:
 I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
 S - Select operation details
 Enter the TEXT command above to include operation text in this list, or,
 enter the GRAPH command to view the list graphically.

 Application            : MAINT            RMF maintenance

 Row  Oper      Duration Job name  Internal predecessors           More preds
 cmd  ws   no.  HH.MM                                             -Int-Ext-
 '''' NV00 005   0.01    RMF_____  ___ ___ ___ ___ ___ ___ ___ ___   0   1
 '''' CPU1 010   0.10    RMFMAINT  005 ___ ___ ___ ___ ___ ___ ___   0   0
 '''' NV00 015   0.01    RMF_____  010 ___ ___ ___ ___ ___ ___ ___   0   0
 ****************************** BOTTOM OF DATA ********************************
```

Figure 51. Operations in the MAINT Application

Selecting TEXT as a primary command allows entry of the operation text. OPC Automation uses the **Operation text** field to contain the request and up to two optional parameters for operations with the workstation defined for OPC Automation. Figure 52 shows the resulting operations text detail panel.

```
--------------------------------- OPERATIONS --------------------  ROW 1 OF 2
Command ===>                                                    Scroll ===> PAGE

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details
Enter the PRED command above to include predecessors in this list, or
enter the GRAPH command to view the list graphically.

Application            : MAINT            RMF maintenance

Row  Oper      Duration Job name  Operation text
cmd  ws   no.  HH.MM
'''' NV00 005   0.01    RMF_____  STOP____________________
'''' CPU1 010   0.11    RMFMAINT  ____________________
'''' NV00 015   0.01    RMF_____  START___________________
****************************** BOTTOM OF DATA ********************************
```

Figure 52. Operations Text Detail Panel

In the applications, OPC Automation defines OPC requests in a generic manner. Figure 52 shows the MAINT application with the first and last operations defined for the NetView workstation NV00. The requests that are forwarded to the NetView workstation NV00 are STOP and START. These requests are expanded by definitions in the policy database into commands; see "Executing OPC Requests with OPC Automation" on page 96.

Handling Time Dependencies

If you require a time dependency, do not place the time consideration on the NV*xx* defined operation because the status change drives the OPC user exit EQQUX007, regardless of the timer status. For a general workstation, such as those defined for OPC Automation, this occurs when all dependencies are fulfilled except the time consideration.

To avoid this problem, define a dummy, non-reporting workstation. Place the timer dependency on this dummy workstation. Define any dependencies on the dummy workstation, which is the predecessor to the NV*xx* workstation. Once you satisfy all other dependencies and complete the time dependency, the dummy timer workstation completes immediately and starts the operation on the NV*nn* workstation.

As an example, redefine the MAINT application shown in Figure 50 on page 93, and Figures 51 and 52 on page 94, with a timer dummy workstation (TIMR) as the first operation of the application. The panel in Figure 53 on page 95 shows this new definition.

```
--------------------------------- OPERATIONS ---------------------  ROW 1 OF 2
Command ===>                                                  Scroll ===> PAGE

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details
Enter the TEXT command above to include operation text in this list, or,
enter the GRAPH command to view the list graphically.

Application            : MAINT            RMF maintenance

Row  Oper      Duration Job name  Internal predecessors           More preds
cmd  ws   no.  HH.MM                                              -Int-Ext-
'''' TIMR 005   0.01    ________  ___ ___ ___ ___ ___ ___ ___ ___   0   1
'''' NV00 010   0.01    RMF_____  005 ___ ___ ___ ___ ___ ___ ___   0   0
'''' CPU1 015   0.10    RMFMAINT  010 ___ ___ ___ ___ ___ ___ ___   0   0
'''' NV00 020   0.01    RMF_____  015 ___ ___ ___ ___ ___ ___ ___   0   0
****************************** BOTTOM OF DATA ********************************
```

Figure 53. Using Time as a Dependency

With this type of structure, OPC Automation can schedule the NV*xx* operation, rather than the timer-dependent dummy workstation, if the application needs scheduling on demand or restarting. This manually initiated procedure is independent of the time consideration, if appropriate.

Changes to the Status of the Operation

The operation with the NV*xx* workstation goes through several status changes as the request defined in the operator text is processed. The initial trigger is one of three status changes. When the operation moves to the A (arrival), R (ready), or * (ready with non-reporting predecessor) status, the OPC Automation function in the EQQUX007 exit is triggered. The exit then examines the request. If the request is valid, the exit transfers it to the target NetView.

If any definition problems are determined, OPC Automation updates the status to E with an error code of U*xxx*. See *System Automation for z/OS Messages and Codes*. OPC Automation takes no further action. The user is then responsible for correcting the error and restarting the application at the failed operation.

If OPC Automation encounters a connectivity problem, it marks the operation with a status of E (error) and an error code of S*xxx*. If this happens, OPC Automation automatically restarts the operation once the connectivity problem is resolved. However, if the operation status is changed manually, the automatic restart is suppressed.

After OPC Automation resolves the request and verifies it, the status is updated to S (started) before OPC Automation submits it. Once the request is submitted and action is requested, OPC Automation updates the status to C (completed).

If the desired result did not occur within the time period specified, the operation ends with an E status and a U*xxx* code, indicating that user intervention is required.

Extending the Daily Plan

OPC Automation does not call EQQUX007 for time-delay operations added at daily planning. To provide time-delay operations added at daily planning, you need to define an operation on a dummy workstation as a predecessor to the NV*xx* workstation. The operation on that workstation completes immediately after the daily plan is extended, and the operation on the NetView workstation is READY

when all of its other dependencies are satisfied. See "Handling Time Dependencies" on page 94 for more information and an example of this technique.

Defining an operation on a dummy workstation is required because the operation-status-change exit is called whenever an operation in the current plan changes status. That exit is also called when a new operation has been added to the current plan by a function other than daily planning jobs, for example, by PIF or by the MCP dialog. The exit is called when the operation is added either to an existing occurrence or as a result of a new occurrence being added to the current plan.

Sending a Request to Optional Installation-Provided Functions

OPC Automation allows installation-specific extensions through optional user-provided modules. Two types of functions are supported:

- Issuing a non-SA z/OS command or request
- Sending a request through to OPC Automation to an installation extension

Because these user-provided modules are unique to your installation, you need to obtain information from your systems programmer/analyst on the use and syntax of these functions.

Executing OPC Requests with OPC Automation

Generally, you must use the OPC-specific OPCA, OPCACMD, and (optionally) OPCAPARM keywords to put an application defined to SA z/OS under the control of OPC Automation. You must define these entries under the MESSAGES/USER DATA policy item of the respective application in the SA z/OS policy database. See *System Automation for z/OS Defining Automation Policy* for more information on the MESSAGES/USER DATA policy item, and Chapter 13, "MESSAGES/USER DATA Entries and USER E-T Pairs for OPC Automation," on page 103 for the OPC-specific keywords.

You can vary the amount of fine-tuning considerably. If you only want to start and stop subsystems known to SA z/OS through OPC in a standard way, you need not even code any of these keywords. On the other hand, you can call user-written modules with OPCACMD that perform tasks not related to any SA z/OS subsystem and that must inform OPC about the success of their execution independently of OPC Automation.

The following sections explain the connection between the OPC request and the OPC-specific MESSAGES/USER DATA keywords by a fairly typical example, describe the use of request parameters, and give an overview of the different request types.

OPC Requests and MESSAGES/USER DATA Keywords

You put a request to OPC Automation by specifying the requested function and (optionally) one or two parameters in the **Operation text** field of the **Operations** panel for an OPC application (for details, see "Defining Applications for OPC Automation in OPC" on page 90). For example:

```
--------------------------------- OPERATIONS ---------------------  ROW 1 OF 2
Command ===>                                                  Scroll ===> PAGE

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details
Enter the PRED command above to include predecessors in this list, or,
enter the GRAPH command to view the list graphically.

Application            : MAINT           Test for maint appl

Row  Oper      Duration Job name  Operation text
cmd  ws   no.  HH.MM
'''' NV04 005   0.01    RMF_____  START___________________
'''' NV04 010   0.01    CICS1H__  START COLD______________
```

Figure 54. OPC/ESA Operations Panel

The request is START in both entries. The second entry has one parameter, namely COLD.

OPCACMD Keyword

OPC Automation uses the **Job name** and the **Operation text** fields of the OPC operation to translate requests into commands. After identifying the subsystem through the **Job name** entry, it consults the OPCACMD entry in the MESSAGES/USER DATA policy item of this subsystem. The CMD attributes of this entry contain the commands that are to be issued in response to various requests, or combinations of request and parameter(s), that can be specified in the **Operation text** field. The following panel gives an example entry for RMF:

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                CMD Processing              Row 1 to 2 of 20
Command ===>                                                 SCROLL===> PAGE

Entry Type : Application          PolicyDB Name   : SCENARIO
Entry Name : RMF                  Enterprise Name : TEST

Subsystem  : RMF
Message ID : OPCACMD

Enter commands to be executed when resource issues the selected message.

Pass/Selection Automated Function/'*'
Command Text
START_____    ________
INGREQ RMF REQ=START,TYPE=NORM,SOURCE=EXTERNAL_____________________________
_______________________________________________________________________________

OPMSG_____    ________
MSG ALL RMF MSG_____________________________________________________________
_______________________________________________________________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 55. Specifying the Command for a Request

These entries specify in the **Command Text** field the commands that are to be issued for RMF when START, respectively, OPMSG requests are put to OPC Automation. Thus, when the 005 request of Figure 54 has been put to OPC Automation, OPC Automation will issue the command

```
INGREQ RMF REQ=START,TYPE=NORM,SOURCE=EXTERNAL
```

OPCA Keyword

Besides specifying the command to be issued in response to the request, you must also tell OPC Automation

- which status you expect the subsystem to assume as a result of this command, and
- the time interval within which the subsystem must assume this status.

This is done through the OPCA keyword. The following panel continues the example of Figure 55 on page 97.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing              Row 1 to 6 of 21
Command ===>                                                  SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : SCENARIO
Entry Name : RMF                   Enterprise Name : TEST

Subsystem  : RMF
Message ID : OPCA

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

    Code 1          Code 2          Code 3           Value Returned
START_________ _______________ _______________ UP,2,RMFUTMER_________________
OPMSG_________ _______________ _______________ UP,1,_____________________________
STOP__________ _______________ _______________ DOWN,2,____________________________
_____________ _______________ _______________ _____________________________
_____________ _______________ _______________ _____________________________
_____________ _______________ _______________ _____________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 56. Specifying Expected Status and Time Interval

Here, the request is specified in the **Code 1** column. The **Value Returned** column contains the expected status, the time interval (in minutes), and optionally a timer name. The **Code 2** and **Code 3** columns are intended for eventual request parameters and consequently left blank in this example.

Flow of Control

By the entries of Figure 55 on page 97 and Figure 56, OPC Automation will execute the start request of Figure 54 on page 97 for RMF as follows:

1. It sets the RMFUTMER timer to two minutes.
2. It issues the command `INGREQ RMF REQ=START,TYPE=NORM,SOURCE=EXTERNAL`.
3. If RMF has assumed the UP state before two minutes have passed, OPC Automation cancels the timer and posts the completion code C (for COMPLETED) back to OPC.
4. After the timer has expired, OPC Automation checks the status of RMF. If RMF is in the UP status, OPC Automation posts C to OPC; otherwise, it posts E (for ERROR) and an additional error code.

Request Parameters and the &EHKVAR*i* Variables

Besides the request itself, the OPC request can contain one or two parameters. You can use this additional information to associate different commands with different variants of the same request; as an example, consider the different startup types for CICS. You can pass the parameters to your command through the task global &EHKVAR1 and &EHKVAR2 variables.

As an example, suppose you want to specify the startup type for a CICS application in an OPC start request. Then you enter it as a request parameter in the **Operation text** field (see the 010 request in Figure 54 on page 97) and pass the startup type to the command specified in the OPCACMD entry by incorporating the &EHKVAR1 variable in the command text. In the following example, this command is not an SA z/OS command, but a user-written CLIST named MYCLIST that uses the startup information to apply the desired startup method.

```
  COMMANDS  HELP
 ------------------------------------------------------------------------------
                                 CMD Processing               Row 1 to 2 of 20
 Command ===>                                                  SCROLL===> PAGE

 Entry Type : Application           PolicyDB Name   : SCENARIO
 Entry Name : CICS1H                 Enterprise Name : TEST

 Subsystem  : CICS1H
 Message ID : OPCACMD

 Enter commands to be executed when resource issues the selected message.

 Pass/Selection Automated Function/'*'
 Command Text
 START_____    ________
 MYCLIST CICS1H &EHKVAR1_____________________________________________________
_____________________________________________________________________________

 _________     ________
 _____________________________________________________________________________
_____________________________________________________________________________
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 57. Specifying a Command that Requires Parameter Information

Then, when the request of the 010 operation in Figure 54 on page 97 is put to OPC Automation, MYCLIST will be called with the arguments CICS1H and COLD. A very simple version of MYCLIST could be as follows:

```
/* MYCLIST SAMPLE */
PARSE UPPER ARG CICSNAME STARTTYPE

IF STARTTYPE='COLD' THEN
   "INGREQ "||CICSNAME||" REQ=START,TYPE=COLD,OUTMODE=LINE"
ELSE
   "INGREQ "||CICSNAME||" REQ=START,TYPE=AUTO,OUTMODE=LINE"
EXIT 0
```

If you use parameters, you must code an OPCA entry for every parameter (combination). For the example of Figure 57, the OPCA entry could look like Figure 58 on page 100.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing              Row 1 to 6 of 21
Command ===>                                                  SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : SCENARIO
Entry Name : CICS1H                Enterprise Name : TEST

Subsystem  : CICS1H
Message ID : OPCA

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

    Code 1          Code 2          Code 3            Value Returned
START_____________ COLD_____________ _______________ UP,10,CICS1TMR_______________
START_____________ AUTO_____________ _______________ UP,5,CICS1TMR________________
_________________ _________________ _______________ ___________________________
_________________ _________________ _______________ ___________________________
_________________ _________________ _______________ ___________________________
_________________ _________________ _______________ ___________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 58. Specifying Expected Status and Time Interval for Different Request Parameters

Request Types

OPC requests can be classified into four types depending on the existence or non-existence of OPC-specific keywords and on the type of command associated with the OPCACMD keyword. The following sections give an overview of these types.

Starting and Stopping Subsystems without OPC-Related Keywords

For START and STOP requests, you need not code any OPC-specific MESSAGES/USER DATA keyword in the policy database. OPC Automation will issue a default command when it detects that no OPCA entry exists for the START or STOP request in the MESSAGES/USER DATA policy item of the subsystem to be started or stopped. In these cases you can specify a valid startup, respectively, shutdown type. The valid startup types are NORM and any startup type that has been defined in the STARTUP policy item of the subsystem to be started. The valid shutdown types are NORM, IMMED, or FORCE.

The start command issued by OPC Automation will be as follows:

```
INGREQ subsystem_name REQ=START,OUTMODE=LINE,SOURCE=EXTERNAL
```

If you specify a startup type, this type will be added to the command.

The format of the stop command issued by OPC Automation is:

```
INGREQ subsystem_name REQ=STOP,OUTMODE=LINE,SOURCE=EXTERNAL
```

If you specify a shutdown type, this type will be added to the command.

Note that if you want to add or change any parameter other than the TYPE parameter, you must specify your INGREQ command in an OPCACMD entry and code the associated OPCA entry. For more information on INGREQ, see *System Automation for z/OS Operator's Commands*.

When the startup or shutdown type is invalid for the subsystem in question, or when the command encounters any problem, the operation will fail with a user abend code of U003.

Canceling a Start or Stop Request

In many cases after a subsystem has been started or stopped, the previous request to SA z/OS needs to be removed to allow normal automation actions to continue. This can be achieved by issuing a CANCEL request type without parameters. Internally an INGSET CANCEL command will be issued to remove the previous START or STOP request for the resource.

Requests Using SA z/OS Automation Functions

These are requests for which an OPCACMD entry is coded, and where the command specified in that entry changes the automation status of the subsystem to UP, RUNNING or AUTODOWN. This can be an SA z/OS base command (for example, INGREQ), but also a user-written command that calls INGREQ. The name of the request must not begin with 'UX'. When the request contains parameters, these are stored in the &EHKVAR1 and &EHKVAR2 variables, respectively, and you can pass them to the command by incorporating these variables into the command text.

For every OPCACMD entry you must code an associated OPCA entry. An OPCAPARM entry is not required.

For START and STOP requests, you may want to use this type instead of coding no OPCACMD entry at all, if you want to override the default values for certain INGREQ parameters.

Subsystem Related Requests not Using SA z/OS Automation Functions

These are requests for which an OPCACMD entry is coded, and where the command specified does not trigger a status change of the subsystem to which it relates. The name of the request must not begin with 'UX'. When the request contains parameters, these are stored in the &EHKVAR1 and &EHKVAR2 variables, respectively, and you can pass them to the command via these variables.

For every OPCACMD entry with such a command you must code an associated OPCA entry. You must also define a corresponding OPCAPARM entry. This entry must specify a user-written timer module that informs OPC whether or not the request was successful (by calling OPCACOMP); this module is called by OPC Automation after the timer defined in the OPCA entry has expired.

For more information on this request type, see "User Functions Related to an SA z/OS-Defined Subsystem" on page 135.

Non-Subsystem Requests

These are requests for which an OPCACMD entry is coded, and where the command specified in that entry does not relate to a subsystem known to SA z/OS. The name of these requests must begin with 'UX'. The OPCACMD entry for a non-subsystem request must be coded as a USER E-T pair. With requests of this type, the complete request buffer will be stored in &EHKVAR1, and it is up to the user —written command to analyze this information. For the request buffer in general, see "Request Buffers and OPC Automation Log Entries" on page 81; for the format of the request buffer for 'UX' requests, see Table 10 on page 133.

No OPCA or OPCAPARM entry is needed for non-subsystem requests. The responsibility for informing OPC about the success of the operation lies entirely with the user-written command.

For more information on this request type, see “Non-Subsystem Operations” on page 139.

Chapter 13. MESSAGES/USER DATA Entries and USER E-T Pairs for OPC Automation

As OPC Automation is integrated into SA z/OS, you must enter any information for OPC Automation in the policy database via the customization dialogs. In most cases the customization dialogs precisely determine the format in which this information must be entered. There are, however, some OPC application-specific automation parameters that must or can only be specified as entries in the MESSAGES/USER DATA policy items of the respective application, or as USER E-T pairs; for general information on the MESSAGES/USER DATA policy item and USER E-T pairs, see *System Automation for z/OS Defining Automation Policy*. In these cases, the customization panels provide no information about the keywords and the format of their parameters.

The following chapter contains detailed descriptions of these automation entries. Note, however, that a general understanding of the MESSAGES/USER DATA policy item will be assumed.

Translating Format Descriptions

The following two examples show how to convert the formal descriptions of the keyword parameters into entries in the MESSAGES/USER DATA and USER E-T PAIRS panels of the customization dialogs.

The first example is the OPCACMD keyword. With this entry, you specify the command that is executed in response to a certain OPC request (see "OPCACMD" on page 112 for more details). The format description of OPCACMD is as follows:

Format

```
OPCACMD CMD=(request,,command)
        .
        .
        .
        [CMD=(request,,command)]
```

The `NAME=value` pairs are called the *attributes* of the entry. OPCACMD has one attribute, CMD, which must occur at least once and may occur more than once. For general information on the format descriptions, see "Notation for Format Descriptions" on page xiii.

The translation of the format description for OPCACMD depends on whether or not the specified command is related to a subsystem that is known to SA z/OS. In the first case (which will also be treated first), the entry is defined in the MESSAGES/USER DATA item of the respective subsystem; for the second case, in which it must be entered within a USER E-T pair, see the end of this section on page 107.

Accordingly, suppose first that a START request is specified in an OPC operation for the subsystem CICS1H, and that you want to start CICS1H with the INGREQ command (for the connection between the OPC request and the command, see Chapter 12, "Automating Applications with OPC Automation," on page 89). To

specify an OPCACMD entry for CICS1H in the customization dialogs, you must call the **Message Processing** panel for that subsystem:

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
                              Message Processing            Row 1 to 4 of 20
Command ===>                                                 SCROLL===> PAGE

Entry Type  : Application           PolicyDB Name   : SCENARIO
Entry Name  : CICS1H                Enterprise Name : TEST

Subsystem   : CICS1H

Enter messages issued by this resource that will result in automated actions.
Actions:  CMD = Command  REP = Reply  CODE = CODE  USER = User defined values

Action    Message ID                                  Cmd  Rep  Code  User
          Description
CMD____   OPCACMD_____________________________
          Commands for OPC requests_______________
_______   ____________________________________
          ____________________________________________
_______   ____________________________________
          ____________________________________________
_______   ____________________________________
          ____________________________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 59. Message Processing Panel of the Customization Dialogs 1

In this panel you specify the keyword of the entry (OPCACMD) in the **Message ID** field. The attributes are specified through the **Action** field. Here two cases must be distinguished according to the following rule:

Rule

- The attribute names CMD, CODE, and REP must be entered in the **Action** field; the values for these attributes are specified in a follow-on panel.
- For all other attributes, you must enter USER in the **Action** field; in this case, both name and value are entered in a follow-on panel.

The fields on the right side of the panel specify how many actions of the respective type are associated with the message ID of the respective line.

To specify the value for the CMD attribute enter `CMD` in the **Action** field and press ENTER. This invokes the **CMD Processing** panel:

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                CMD Processing              Row 1 to 2 of 20
Command ===>                                                 SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : SCENARIO
Entry Name : CICS1H                Enterprise Name : TEST

Subsystem  : CICS1H
Message ID : OPCACMD

Enter commands to be executed when resource issues the selected message.

Pass/Selection Automated Function/'*'
Command Text
START_____    ________
INGREQ CICS1H REQ=START,TYPE=AUTO_____________________________________________
______________________________________________________________________________

_________     ________
______________________________________________________________________________
______________________________________________________________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 60. CMD Processing Panel of the Customization Dialogs

Every entry in this panel consists of three fields that correspond to the three items of the value list for the CMD attribute. Thus, the general format for the CMD attribute is

CMD=([*Pass/Selection*],[*Automated_Function*],*Command_Text*)

The format description of the CMD attribute for a certain keyword specifies what type of information you must enter in the three fields. For the OPCACMD keyword, the omission of the second value signifies that the **Automated Function** field is to be left blank. In the first field, you must specify the request; this is the first value in the **Operation text** field of the OPC request. A command text must be specified in the third field.

For the CMD, REP, and CODE attributes, the following notational conventions apply:

Notational Conventions for CMD, CODE, REP

- The '=' sign, the parentheses enclosing the value list, and the commas separating the individual values must not be entered in the panels.

  They just serve to make the format description more readable and to identify uniquely the panel field with which a value specification is associated.
- When the format of any value is specified in more detail, and this specification itself contains a comma (or blank), the value is enclosed in single quotes; these quotes must also not be entered in the respective panel field.

For more information on the panel fields, see *System Automation for z/OS Defining Automation Policy*.

The OPCA entry (see "OPCA" on page 109) supplies the second example. This entry defines the state that is the expected result of the command specified in the

OPCACMD entry, and the time interval within which this state must have been reached. The format of the OPCA entry is as follows:

Format

```
OPCA CODE=(request,[parm1],[parm2],'expstatus,timerint,timerid')
     .
     .
     .
     [OPCA CODE=(request,[parm1],[parm2],'expstatus,timerint,timerid')]
```

If the expected status is UP, and this state must have been assumed within three minutes, you must enter the OPCA entry for CICS1H as follows.

First specify the keyword in the **Message Processing** panel:

```
 COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
                              Message Processing            Row 1 to 4 of 20
Command ===>                                                 SCROLL===> PAGE

Entry Type  : Application          PolicyDB Name   : SCENARIO
Entry Name  : CICS1H               Enterprise Name : TEST

Subsystem   : CICS1H

Enter messages issued by this resource that will result in automated actions.
Actions:  CMD = Command  REP = Reply  CODE = CODE  USER = User defined values

Action    Message ID                                 Cmd  Rep  Code  User
          Description
_______   OPCACMD_____________________________
          Commands for OPC requests_______________
CODE___   OPCA_______________________________
          Expected status and time interval______
_______   _____________________________________
          _____________________________________
_______   _____________________________________
          _____________________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 61. Message Processing Panel of the Customization Dialogs 2

Now you must enter CODE in the **Action** column according to the rule stated on page 104. When you press ENTER, the **Code Processing** panel is called:

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing              Row 1 to 6 of 20
Command ===>                                                  SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : SCENARIO
Entry Name : CICS1H                Enterprise Name : TEST

Subsystem  : CICS1H
Message ID : OPCA

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

    Code 1          Code 2          Code 3            Value Returned
START_________ _______________ _______________ UP,3,RMFUTMER_______________
_____________ _______________ _______________ ____________________________
_____________ _______________ _______________ ____________________________
_____________ _______________ _______________ ____________________________
_____________ _______________ _______________ ____________________________
_____________ _______________ _______________ ____________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 62. Code Processing Panel of the Customization Dialogs

Here you must specify the values of the CODE attribute as displayed in Figure 62. The general format of the CODE attribute is:

CODE=([*Code_1*],[*Code_2*],[*Code_3*],*Value_Returned*)

For the OPCA keyword, **Code 1** must be the request. In the **Code 2** and **Code 3** fields, you can specify eventual parameters of the request. The **Value Returned** must specify the expected status, the time interval, and (optionally) a timer name, separated by commas.

Note: For other keywords, the fields can have a completely different function from those discussed above.

An asterisk (*) is admitted as a *trailing* wildcard character for the three **Code** fields; that is, you can specify simply * and ABC*, but not *ABC.

The preceding two examples illustrate how format descriptions are translated into MESSAGES/USER DATA entries. However, as already indicated, the OPCACMD entry must be coded in a USER E-T pair if the command to be specified is not related to a subsystem known to SA z/OS. The following figure provides an example:

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
                        UET Keyword-Data Specification          Row 3 from 3
Command ===>                                                  SCROLL===> PAGE

Entry Type : User E-T Pairs        PolicyDB Name   : SCENARIO
Entry Name : NONSUBS               Enterprise Name : TEST
UET Entry  : DUMMY                 UET Type        : OPCACMD

Action   Keyword/Data(partial)
________ CMD
         (UXCKSPL,,'EVJERUX1 &EHKVAR1')
****************************** Bottom of data ********************************









 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 63. OPCACMD in a USER E-T Pair

As you can see from Figure 63, you must enter CMD in the **Keyword** field and the value list in the **Data** field.

For the USER E-T pairs, the following translation conventions apply:

Notational Conventions for UET Keyword/Data Pairs

- The '=' sign must not be entered in the panels.
- Everything to the right of the '=' sign, including parentheses, commas, and single quotes, *must* be entered in the **Data** field.

OPC-Specific MESSAGES/USER DATA Keywords

The following keywords are specific for OPC Automation.

Entry	Description
"OPCA" on page 109.	Use this ID to define the expected state of the subsystem and the time interval within which this state must have been reached.
"OPCACMD" on page 112.	Use this ID to specify the command to be issued in response to an OPC request.
"OPCAPARM" on page 116.	Use this ID to specify modifications of eventual request parameters and a timer module for user-written commands.

OPCA

Purpose

With the OPCA entry, you define the state that is the expected result of a request (with or without parameters), and the time interval within which this state must have been reached. The OPCA entry is defined in the MESSAGES/USER DATA policy item of the subsystem which is to be put under control of OPC.

Format

Format

```
OPCA CODE=(request,[parm1],[parm2],'expstatus,timerint,timerid')
     .
     .
     .
     [OPCA CODE=(request,[parm1],[parm2],'expstatus,timerint,timerid')]
```

Parameters

request
Request specified in the OPC operation text.

parm1
Parameter 1 as specified in the OPC operation text.

parm2
Parameter 2 as specified in the OPC operation text.

expstatus
Expected status of the subsystem at the completion of the request. *expstatus* stands for one of the following values: UP, RUNNING or DOWN.

Note: For backward compatibility CTLDOWN and AUTODOWN are also allowed for this entry and will be treated the same as DOWN.

timerint
Timer interval in minutes. The maximum value permitted is 1439 (23 hours and 59 minutes).

Set a timer interval that is long enough for the operation to complete reasonably. If the operation does not complete in the interval specified, then an error is posted to OPC.

timerid
Timer ID — from 1 to 8 characters.

This must be a valid NetView timer ID with a value not equal to `ALL` or beginning with `SYS`, `ING`, or `AOF`. This field is optional.

Usage Notes

For every OPCA entry, there must be a corresponding OPCACMD entry.

Example 1

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing              Row 1 to 6 of 21
Command ===>                                                  SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : SCENARIO
Entry Name : RMF                   Enterprise Name : TEST

Subsystem  : RMF
Message ID : OPCA

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

    Code 1          Code 2          Code 3            Value Returned
START_____________ _______________ _______________ UP,3,RMFUTMER_______________
STOP______________ _______________ _______________ DOWN,2,RMFDTMER_____________
_____________________ _____________ _____________ ___________________________
_____________________ _____________ _____________ ___________________________
_____________________ _____________ _____________ ___________________________
_____________________ _____________ _____________ ___________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

This panel shows the entries for two requests without parameters.

Example 2

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing              Row 1 to 6 of 21
Command ===>                                                  SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : SCENARIO
Entry Name : CICS1                 Enterprise Name : TEST

Subsystem  : CICS1
Message ID : OPCA

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

    Code 1          Code 2          Code 3            Value Returned
START_____________ AUTO____________ _______________ UP,5,CICS1TMR_______________
START_____________ COLD____________ _______________ UP,10,CICS1TMR______________
_____________________ _____________ _____________ ___________________________
_____________________ _____________ _____________ ___________________________
_____________________ _____________ _____________ ___________________________
_____________________ _____________ _____________ ___________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

This example shows two entries, one for an automatic start, and one for a cold start, of a subsystem called CICS1. The AUTO or COLD parameter is added to the START request in OPC to indicate which one of several startup procedures is to be used. Presumably the two operations will take differing amounts of time, so the timer intervals are different.

This OPCA code entry is used in conjunction with a user-written CLIST, specified in the OPCACMD entry; see “Example 2” on page 113 for details of that entry and the sample CLIST.

Example 3

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing              Row 1 to 6 of 21
Command ===>                                                  SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : SCENARIO
Entry Name : TESTAPPL              Enterprise Name : TEST

Subsystem  : TESTAPPL
Message ID : OPCA

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

    Code 1          Code 2          Code 3            Value Returned
RECYCLE________ _______________ _______________ UP,5,TESTTIMER_______________
_______________ _______________ _______________ ____________________________
_______________ _______________ _______________ ____________________________
_______________ _______________ _______________ ____________________________
_______________ _______________ _______________ ____________________________
_______________ _______________ _______________ ____________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

This example shows a RECYCLE type of operation (bring the subsystem down and restart it immediately) being defined. See also “Example 4” on page 115 for the corresponding OPCACMD definition.

OPCACMD

Purpose

With the OPCACMD entry, you define the command that is executed in response to a request (with or without parameters). Except for non-subsystem commands, there must be a corresponding OPCA entry for every OPCACMD entry.

Format

Format

```
OPCACMD CMD=(request,,command)
        .
        .
        .
        [CMD=(request,,command)]
```

Parameters

request
Request specified in the OPC operation text.

command
Actual command to be built.

Usage Notes

This entry is necessary for all request types except START, STOP, and CANCEL requests. If no entries are supplied for START, STOP and CANCEL, the actions taken are detailed in “Starting and Stopping Subsystems without OPC-Related Keywords” on page 100 and “Canceling a Start or Stop Request” on page 101. The place where the OPCACMD entry is defined depends on the request type. When the request is related to a subsystem, it is defined in the MESSAGES/USER DATA policy item of the respective application. When the request is a non-subsystem request (see “Non-Subsystem Operations” on page 139), the OPCACMD entry must be entered in a USER E-T PAIRS entry.

Use SA z/OS commands to shut down and start up subsystems. This avoids the problem of having to determine the specific commands required for each subsystem.

Example 1

```
  COMMANDS  HELP
-------------------------------------------------------------------------------
                                CMD Processing              Row 1 to 2 of 20
Command ===>                                                 SCROLL===> PAGE

Entry Type : Application          PolicyDB Name   : SCENARIO
Entry Name : RMF                  Enterprise Name : TEST

Subsystem  : RMF
Message ID : OPCACMD

Enter commands to be executed when resource issues the selected message.

Pass/Selection Automated Function/'*'
Command Text
START_____   ________
INGREQ RMF REQ=START,VERIFY=NO,SOURCE=EXTERNAL,TYPE=NORM_____________________
_______________________________________________________________________________

STOP______   ________
INGREQ RMF REQ=STOP,VERIFY=NO,SOURCE=EXTERNAL,TYPE=NORM_____________________
_______________________________________________________________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Example 2

```
  COMMANDS  HELP
-------------------------------------------------------------------------------
                                CMD Processing              Row 1 to 2 of 20
Command ===>                                                 SCROLL===> PAGE

Entry Type : Application          PolicyDB Name   : SCENARIO
Entry Name : CICS1                Enterprise Name : TEST

Subsystem  : CICS1
Message ID : OPCACMD

Enter commands to be executed when resource issues the selected message.

Pass/Selection Automated Function/'*'
Command Text
START_____   ________
MYCLIST CICS1 &EHKVAR1________________________________________________________
_______________________________________________________________________________

__________   ________
_______________________________________________________________________________
_______________________________________________________________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

This example assumes that AUTO or COLD is added as a parameter to the START request in OPC to indicate which one of several startup procedures is to be used. The command specified in the **Command Text** field is a user-written CLIST. This CLIST is passed the parameter value (AUTO or COLD) in the &EHKVAR1 variable.

A very simple version of the CLIST, which could be expanded in your environment to include other parameters and additional function, could be as follows:

```
/* MYCLIST SAMPLE */
PARSE UPPER ARG CICSNAME STARTTYPE

IF STARTTYPE='COLD' THEN
```

```
   "INGREQ "||CICSNAME||" REQ=START,TYPE=COLD,OUTMODE=LINE"
ELSE
   "INGREQ "||CICSNAME||" REQ=START,TYPE=AUTO,OUTMODE=LINE"
EXIT 0
```

The OPCACMD entry of this example must be supplemented by an OPCA entry as in "Example 2" on page 110.

Example 3

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
                          UET Keyword-Data Specification           Row 3 from 3
Command ===>                                                     SCROLL===> PAGE

Entry Type : User E-T Pairs        PolicyDB Name   : SCENARIO
Entry Name : NONSUBS               Enterprise Name : TEST
UET Entry  : DUMMY                 UET Type        : OPCACMD

Action   Keyword/Data(partial)
________ CMD
         (UXCINITS,'MVS $TI20-30,C=P')
****************************** Bottom of data ********************************









 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

This example shows a command that is not related to a subsystem known to SA z/OS (see "Non-Subsystem Operations" on page 139), and which, accordingly, must be defined as a USER E-T pair (see "Non-Subsystem Operations" on page 139). The jobname of the OPC request would be `DUMMY`.

OPC Automation recognizes such requests by the fact that the request name begins with 'UX'. In the example, OPC Automation simply issues the MVS command `$TI20-30,C=P`, which tells JES to change initiators 20 to 30 so that they process jobs of class P.

Example 4

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                CMD Processing               Row 1 to 2 of 20
Command ===>                                                  SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : SCENARIO
Entry Name : TESTAPPL              Enterprise Name : TEST

Subsystem  : TESTAPPL
Message ID : OPCACMD

Enter commands to be executed when resource issues the selected message.

Pass/Selection Automated Function/'*'
Command Text
RECYCLE___     ________
EVJESHUT TESTAPPL ALL_____________________________________________________
_____________________________________________________________________________

__________     ________
_____________________________________________________________________________
_____________________________________________________________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

This example shows a RECYCLE type of operation (bring the subsystem down and restart it immediately) being defined. Note that the command to be issued is EVJESHUT, which will issue the

```
INGREQ TESTAPPL REQ=STOP,RESTART=YES,SCOPE=ALL,SOURCE=EXTERNAL
```

command in linemode and verify that it is accepted or else post the operation in error. For more information on EVJESHUT, see “EVJESHUT” on page 120; for the corresponding OPCA definition, see “Example 3” on page 111.

OPCAPARM

Purpose

The OPCAPARM entry supplies replacements for eventual request parameters and the name of a user-written timer module. The OPCAPARM entry is defined in the MESSAGES/USER DATA policy item of the subsystem which is to be put under control of OPC.

OPCAPARM is optional except for requests that relate to a subsystem defined to SA z/OS, but where the command specified in the OPCACMD entry is not an SA z/OS command; in this case, you must specify a timer module.

Format

Format

```
OPCAPARM CODE=(request,parm1,parm2,'parm1value,parm2value,timermod')
         .
         .
         .
         [CODE=(request,parm1,parm2,'parm1value,parm2value,timermod')]
```

Parameters

request
Request specified in the OPC-operation definition.

parm1
Parameter 1 as specified in the OPC-operation text.

parm2
Parameter 2 as specified in the OPC-operation text.

parm1value
Substitution value used in the actual command.

parm2value
Substitution value used in the actual command.

timermod
Module called at the timer interval specified in the OPCA CODE entry for this subsystem. You must specify a timer module when the command specified in the OPCACMD entry relates to a subsystem defined to SA z/OS, but does not trigger a status change; see "User Functions Related to an SA z/OS-Defined Subsystem" on page 135.

Usage Notes

OPCAPARM is optional except for requests that relate to a subsystem defined to SA z/OS, but where the command specified in the OPCACMD entry is a user-supplied module that does not trigger a status change of the subsystem. In this case, a you must specify a timer module. See "Implementing Completion of a Request" on page 136 for more details.

Example 1

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing              Row 1 to 6 of 21
Command ===>                                                  SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : SCENARIO
Entry Name : RMF                   Enterprise Name : TEST

Subsystem  : RMF
Message ID : OPCAPARM

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

    Code 1          Code 2          Code 3             Value Returned
START_________ _______________ _______________ ,,_____________________________
_____________ _______________ _______________ _____________________________
_____________ _______________ _______________ _____________________________
_____________ _______________ _______________ _____________________________
_____________ _______________ _______________ _____________________________
_____________ _______________ _______________ _____________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

In this example, no additional parameters are needed for the request. An entry of this sort is entirely optional, and need not be coded at all.

Example 2

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing              Row 1 to 6 of 21
Command ===>                                                  SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : SCENARIO
Entry Name : CICS1                 Enterprise Name : TEST

Subsystem  : CICS1
Message ID : OPCAPARM

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

    Code 1          Code 2          Code 3             Value Returned
START_________ AUTO__________ _______________ ,,_____________________________
START_________ COLD__________ _______________ ,,_____________________________
_____________ _______________ _______________ _____________________________
_____________ _______________ _______________ _____________________________
_____________ _______________ _______________ _____________________________
_____________ _______________ _______________ _____________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

This example shows two entries, one for an automatic start, and one for a cold start, of a subsystem called CICS1. The AUTO or COLD parameter is added to the START request in OPC to indicate which one of several startup procedures is to be used.

Chapter 14. OPC Automation Common Routines and Data Areas

This chapter contains the common routines that are supplied by OPC Automation. Furthermore, it describes the data areas that are used to transfer requests from OPC to SA z/OS.

OPC Automation Common Routines

This chapter describes OPC Automation common routines which request information or perform tasks associated with OPC Automation. You can use these common routines in automation procedures you create. Examples, sample routines, and data area information are given to show how this might be done.

OPC Automation provides new routines to retrieve and update OPC Automation-unique information. These routines can also be used in user-written extensions of OPC Automation. The following routines are arranged alphabetically for easy reference.

Note

When using any of the OPC Automation commands that use the EQQYCOM (also called the PIF) interface, you should be careful to ensure serialization. This can be achieved by choosing the same auto operator for the command to run on. If you fail to do this you may receive an EQQZ038E message indicating that the OPC message log is not available, because the log data set (EQQMLOG) is required by EQQYCOM exclusively for each invocation.

EQQYCOM is used by the following commands: OPCACAL, OPCALIST, OPCAMOD.

EVJESHUT

Purpose

Use the EVJESHUT routine in a RECYCLE operation defined in the policy data base. A RECYCLE operation is one in which a subsystem which is currently UP is brought down and immediately restarted. EVJESHUT will issue the appropriate INGREQ command and verify that it is accepted by the automation platform. If it is not accepted, then EVJESHUT will post the operation in error to OPC. See "Example 4" on page 115 to see how to include EVJESHUT in an OPCACMD entry.

Format

Syntax

```
EVJESHUT  subsys scope
```

Parameters

subsys
The name of a valid subsystem defined to SA z/OS.

scope
The scope of the INGREQ request. The valid values for *scope* are ALL, CHILDREN, and ONLY. See *System Automation for z/OS Operator's Commands*.

Usage Notes

The **Status check on requests** field in the ENVIRON OPCAO item of the OPC SYSTEM DETAILS policy object determines for a system what OPC Automation is to do when a subsystem of this system is already in the status requested by OPC. This field affects EVJESHUT as follows: When the field is set to NO, and the subsystem is already in AUTODOWN, CTLDOWN, DOWN, or ENDED status, then a RECYCLE operation with EVJESHUT will be allowed to proceed and EVJESHUT will issue an INGREQ command to stop the requested subsystem. If the field is set to YES, then the subsystem must be in UP or RUNNING status for EVJESHUT to be issued; all other statuses will result in the operation terminating and an error posted to OPC.

OPCACAL

Purpose

The OPCACAL command retrieves OPC calendar status information. Use this command for your automation CLISTs. OPCACAL uses the EQQYCOM (also called the PIF) interface in OPC. For more information about this interface, see the *OPC/ESA Interfaces Guide*.

To show the use of this command, OPC Automation provides a REXX sample named EVJERCAL.

Format

Syntax

```
OPCACAL [SUBSYS=subsystem,CALENDAR=calname]
```

Parameters

SUBSYS=*subsystem*
OPC/ESA subsystem ID — 4 characters.

OPCA is the default subsystem name.

CALENDAR=*calname*
Calendar ID — 16 characters.

DEFAULT is the default calendar name, used if this parameter is not coded.

Usage Notes

OPCACAL returns data in the following format:

- EVJ440I date_format day_of_week Work, or
- EVJ440I date_format day_of week Free

The date format is set by the NetView DEFAULTS LONGDATE value. For example:

```
EVJ440I 05/03/04 MONDAY Work
```

or

```
EVJ440I 20040502 SUNDAY Free
```

The EVJ440I message is PIPEd to the CONSOLE ONLY and does not appear in the netlog. The message can be processed by calling OPCACAL in a PIPE.

OPCACMD

Purpose

The OPCACMD command is primarily used to retrieve OPC current plan data so that it can be viewed or modified by operators. The operator simply types in OPCACMD and fills in the panel (EVJKAC01) which is returned. But OPCACMD will also accept optional parameters on input. It could thus be assigned to a PF key, making it more user-friendly.

Format

Syntax

```
OPCACMD APPLID=application_id,OPNO=operation_number,
        WSNAME=ws_name,STATUS=status,ERRCODE=errorcode,
        PRIORITY=priority,OWNER=owner_name,GROUP=group_name,
        JOBNAME=job_name
```

Parameters

application_id
Application name — 1 to 16 characters. May be generic.

operation_number
Operation number — 2 digits. Must be numeric or left unspecified.

ws_name
Workstation name — 4 characters. May be generic.

status
Occurrence status — 1 character. Must be valid or left unspecified.

Valid status:

A Arriving
C Completed
E Error
I Interrupted
R Ready
S Started
U Undecided
W Waiting
X Reset

errorcode
Error code — 4 characters. May be generic.

priority
Priority — 1 digit. Must be numeric or left unspecified. Acceptable values are from 1 (low) to 9 (high).

owner_name
OPC application owner name — up to 16 characters. May be generic.

group_name
OPC application group name — up to 8 characters. May be generic.

job_name
Job name — up to 8 characters. May be generic.

Usage Notes

Only as many parameters are required as necessary to identify the application(s) requested. Some parameters may be left unspecified (they will default). Some may

be generic; that is, they may be only partial and end with an asterisk (*) to indicate a partial match. You may also use a percent sign (%) to substitute for a single character.

OPCACOMP

Purpose

The OPCACOMP command completes execution of a subsystem-related request by updating the OPC Automation status file and calling OPCAPOST (see "OPCAPOST" on page 130).

Format

Syntax

```
OPCACOMP  subsys,sequence_number,status [,error_code]
```

Parameters

subsys
Subsystem or pseudo-subsystem to identify the request.

sequence_number
Sequence number assigned to this request by the EVJESPVY module.

status
Operation status reflected to OPC Valid statuses:
C Complete
E Error

error_code
Error code — 4-character value
Takes the form *annn*, where *a* is alphabetic and *nnn* are numerics. Do not specify the values U*xxx* and S*xxx*; reserve them for OPC Automation.
If the status is error, the error code is returned to OPC.

Usage Notes

Call this routine in user-written functions that are related to a subsystem defined to SA z/OS whenever the standard modules for completing a request (EVJESPSC and EVJESPTE) cannot be used. See "Implementing Completion of a Request" on page 136.

Example

```
OPCACOMP  RMF,842,E,R028
```

This example shows setting the operation requested for RMF in an error status with an error code of R028.

OPCALIST

Purpose

The OPCALIST command retrieves OPC data. Use this command in your own automation CLISTS. The module creates a CPOPCOM call to OPC to retrieve the data. OPCALIST uses the EQQYCOM (also called the PIF) interface in OPC. For more information about this interface, see the *OPC/ESA Interfaces Guide*.

Format

Syntax

```
OPCALIST SUBSYS=subsystem,ADID=id,IA=yymmddhhmm,
         PRIORITY=nnnn,ERRCODE=cccc,STATUS=s,OPNO=nnnn,
         JOBNAME=name,WSNAME=name,
         GROUP=groupname,OWNER=name
```

Parameters

SUBSYS=*subsystem*
OPC subsystem ID — 4 characters.

ADID=*id*
Application description ID — up to 16 characters.

IA=*yymmddhhmm*
Input arrival date *yymmdd* and time *hhmm*.

PRIORITY=*nnnn*
Priority — 4 digits.

ERRCODE=*cccc*
Error code — 4 characters.

STATUS=*s*
Occurrence status. Valid statuses:

R	Ready
S	Started
C	Completed
E	Error
I	Interrupted

Refer to the OPC documentation for more information.

OPNO=*nnnn*
Operation number — 4 digits.

JOBNAME=*name*
Job name — up to 8 characters.

WSNAME=*name*
Workstation name — 4 characters.

GROUP=*groupname*
OPC/ESA application group name — up to 8 characters.

OWNER=*name*
OPC/ESA application owner name — up to 16 characters.

Usage Notes

Only as many parameters are required as necessary to identify the application(s) requested. Some parameters may be left unspecified (they will default). Some may

be generic; that is they may be only partial and end with an asterisk (*) to indicate a partial match. You may also use a percent sign (%) to substitute for a single character.

The response to the OPCALIST is made up of three messages. The following example below shows a typical response where:

EVJ410I
This is the message header, showing row titles. This is always present.

EVJ411I
This is the detail message. If there are no entries matching the selection criteria this message is not produced.

EVJ412I
This is the end of request message.

Response details are:

ADID
Application Description Id — up to 16 characters

JOBNAME
Job Name — up to 8 characters

WS
Workstation Name — up to 4 characters

OPNO
Operation Number — up to 4 numbers

S Status (See “Parameters” on page 125 for valid statuses)

ERRC
Error Code (set to none for no error)

IA Input Arrival — date (yymmdd) and time (hhmm)

OPTEXT
Descriptive Text — up to 24 characters

Example

```
EVJ410I   ADID   JOBNAME   WS  OPNO S ERRC IA          OPTEXT
EVJ411I   MAINT2 RMF      NV05 0010 C NONE 9707190615  STOP
EVJ411I   MAINT2 RMF      NV05 0015 C NONE 9707190615  START
EVJ411I   MAINT2 RMF      NV05 0010 C NONE 9707200615  STOP
EVJ411I   MAINT2 RMF      NV05 0015 C NONE 9707200615  START
EVJ412I   END OF REQUEST
```

OPCAMOD

Purpose

The OPCAMOD command modifies OPC data. This command is used in the OPCACMD CLIST and could be used in your own automation CLISTS. The module creates a CPOPCOM or CPOCCOM call to OPC to perform occurrence or operation changes. OPCAMOD uses the EQQYCOM (also called the PIF) interface in OPC. For more information about this interface, see the *OPC/ESA Interfaces Guide*.

Format

Syntax

```
OPCAMOD SUBSYS=subsystem,ADID=id,IA=yymmddhhmm,
        IANEW=yymmddhhmm,DEADLINE=yymmddhhmm,PRIORITY=nnnn,
        ERRCODE=cccc,OPNO=nnnn,STATUS=s,
        JOBNAME=name,WSNAME=name,DESC=text,
        EDUR=hhmm,PSUSE=nnnn,R1USE=nnnn,R2USE= nnnn,
        JCLASS=c,AEC=Y|N,ASUB=Y|N,AJR=Y|N,TIMEDEP=Y|N,
        CLATE=Y|N,HRC=value,FORM=value,OPIA=yymmddhhmm,
        OPDL=yymmddhhmm,RERUT=Y|N,USERDATA=userdata,
        RESTA=Y|N,DEADWTO=Y|N
```

Parameters

SUBSYS=*subsystem*
OPC/A subsystem ID — 4 characters (default OPCA).

ADID=*id*
Application description ID — up to 16 characters (required).

IA=*yymmddhhmm*
Input arrival date *yymmdd* and time *hhmm* (required).

IANEW=*yymmddhhmm*
New input arrival date and time.

DEADLINE=*yymmddhhmm*
Deadline date and time.

PRIORITY=*nnnn*
Priority — 4 digits.

ERRCODE=*cccc*
Error code — 4 characters.

STATUS=*s*
Occurrence status. Valid statuses:

R	Ready
S	Started
C	Complete
E	Error
I	Interrupted

Refer to the OPC documentation for more information.

JOBNAME=*name*
Job name — up to 8 characters.

WSNAME=*name*
Workstation name — 4 characters.

DESC=*text*
Descriptive text — 24 characters.

EDUR=*hhmm*
Estimated duration (hours and minutes).

PSUSE=*nnnn*
Number of parallel servers required — 4 digits.

R1USE=*nnnn*
Amount of resource 1 required — 4 digits.

R2USE=*nnnn*
Amount of resource 2 required — 4 digits.

JCLASS=*c*
MVS job class — 1 character.

AEC=Y/N
Y — Perform automatic error completion. N — Do not perform automatic error completion.

ASUB=Y/N
Y — Perform automatic job submission. N — Do not perform automatic job submission.

AJR=Y/N
Y — Perform automatic job hold and release. N — Do not perform automatic job hold and release.

TIMEDEP=Y/N
Y — Time dependent job. N — Not a time dependent job.

CLATE=Y/N
Y — Cancel if time job and late. N — Do not cancel if time job and late.

HRC
Highest successful return code.

FORM=*value*
Form number — 8 characters.

OPIA=*yymmddhhmm*
Operation input arrival date and time.

OPDL=*yymmddhhmm*
Operation deadline date and time.

RERUT=Y/N
Y — Reroutable operation. N — Not a reroutable operation.

USERDATA=*userdata*
User data — up to 16 characters.

RESTA=Y/N
Y — Restartable operation. N — Not a restartable operation.

DEADWTO=Y/N
Y — Issue WTO if deadline missed. N — Do not issue WTO if deadline missed.

Usage Notes

OPCAMOD may be used to set the status of operations occurring on general workstations which are not automatically reporting (such as CPUs). OPCAPOST will set the status of operations which occur on automatically reporting general workstations (a NetView, for example).

Format

Syntax

```
OPCAMOD SUBSYS=subsystem,ADID=id,IA=yymmddhhmm,
        IANEW=yymmddhhmm,DEADLINE=yymmddhhmm,PRIORITY=nnnn,
        ERRCODE=cccc,STATUS=s
```

Parameters

SUBSYS=*subsystem*
OPC/ESA subsystem ID — 4 characters (default OPCA).

ADID=*id*
Application description ID — up to 16 characters (required).

IA=*yymmddhhmm*
Input arrival date *yymmdd* and time *hhmm* (required).

IANEW=*yymmddhhmm*
New input arrival date and time.

DEADLINE=*yymmddhhmm*
Deadline date and time.

PRIORITY=*nnnn*
Priority — 4 digits.

ERRCODE=*cccc*
Error code — 4 characters.

STATUS=*s*
Occurrence status. Valid statuses:

W	Waiting
C	Completed

Example

```
OPCAMOD SUBSYS=OPCA,ADID=TEST,IA=9403100900,OPNO=0010,
  STATUS=W
```

This example will set the status of operation number 0010 in application test, to waiting.

Example

```
OPCAMOD SUBSYS=OPCA,ADID=TEST,IA=9403100900,STATUS=W
```

This example will set the occurrence status of application TEST to WAITING. All operations in application TEST will be set to WAITING.

OPCAPOST

Purpose

OPCAPOST posts the status of an OPC Automation operation back to OPC. Because OPCAPOST uses the EQQUSINT interface, it can only change the status of operations on automatic reporting workstations. For more information on this interface, see *OPC/ESA Installation and Customization*.

Format

Syntax

```
OPCAPOST ADNAME=adname,WSNAME=wwww,OPNUM=nn,
         TYPE={S|C|I|E|X},ERRCODE=xxxx
```

Parameters

ADNAME=*adname*
Application name — 1 to 16 characters.

WSNAME=*wwww*
Workstation name — 1 to 4 characters.

OPNUM=*nn*
Operations number — 2 digits.

TYPE={S | C | I | E | X}
Type of call — 1 character. Acceptable event types:

S Started
C Complete
I Interrupted
E Error
X Reset

ERRCODE=*xxxx*
Error code — 4 characters.

Note: This parameter is only valid with TYPE=E.

Usage Notes

OPCAPOST can be used to set the status of an operation which occurs on an automatically reporting general workstation (a NetView, for example) only. Due to restrictions in the OPC interface, OPCAPOST cannot set the status of operations on general workstations which are not automatically reporting (such as CPUs). OPCAMOD is available to set the status of such operations if this is required.

OPC Automation sets a return code on completion of the execution of the OPCAPOST command processor, as follows:

0 Successful command

4 Parameter error

8 OPCAPOST failed.

OPCSRST

Purpose

The OPCSRST command lets you manipulate the availability of OPC special resources. It is similar to the SRSTAT operator command, as presented in Chapter 16, "OPC Automation Operator Commands," on page 145.

Format

Syntax

```
OPCSRST SUBSYS=subsys,SRNAME=srname,AVAIL=Y|N
```

Parameters

subsys
Subsystem ID — 4 characters.

srname
Special resource name — up to 44 characters.

AVAIL=Y/N
Availability indicator.

Usage Notes

OPCSRST uses the following return codes:

0 Accepted by OPC — Special Resource-event issued

4 Request failed — Parameter error detected by this program

8 Request failed — Rejected by OPC — No Special Resource-event issued.

Internal failures:

1011 SRNAME keyword not supplied
1010 AVAIL keyword not supplied
1020 DSILOD failed — Unable to load EQQUSINS
1030 DSILCS failed — Unable to obtain SWB
1031 DSIPRS failed — Unable to determine size of PDB
1032 DSIGET failed — Unable to obtain storage
1033 DSIPRS failed — Unable to do parse

Example

```
OPCSRST SUBSYS=OPCT,SRNAME='IMSCNTL.IMS01A.RUNNING',AVAIL=Y
```

In this example, the OPCSRST command is run when the IMS control region IMS01A becomes available. The special resource becoming available makes it possible for work that depends on this control region's execution to run. When IMS01A becomes available, a number of applications are added to the current plan.

The variable used for *subsys*, OPCT, is the name of the tracker subsystem. Only in this command is the tracker subsystem name required.

Data Areas

This section shows the following:
- Requestor ID block (&EHKVAR9)
- Request buffer

Requestor ID Block (&EHKVAR9)

OPC Automation sets the task global variable (&EHKVAR9) in the request module and passes it to the user module, as follows:

```
Name of Subsystem, Sequence #, Module Name, Domain ID
```

Table 8 shows the lengths and values of the variables.

Table 8. Lengths and Values of Task Global Variable (EHKVAR9)

Variable	Name of Subsystem	Sequence #	Module Name	Domain ID
Length (characters)	8	4	8	5
Values	SA z/OS Subsystem Name (Standard)	OPC Sequence Number (Numeric)	Check Module Name	NetView Domain ID (Standard)

The following example shows values substituted for each variable shown in Table 8.

```
RMF,7842,OPCACOMP,NETVT
```

The values are defined as follows:

RMF	This request is for subsystem RMF.
7842	The OPC sequence number is 7842.
OPCACOMP	When the requested function has been completed, invoke the OPCACOMP module.
NETVT	This function was executed in NetView domain NETVT.

Note: Other options can also use this block. Although OPCACOMP is shipped with the OPC Automation option, you can also use a user-supplied module.

Request Buffer

Table 9 shows the request buffer layout for standard subsystem operations:

Table 9. Request Buffer Layout for Standard Subsystem Operations. Length represents the maximum length if format is variable.

Field	Length	Format Fixed/Variable	Value	Obtained From
Request ID	8	F	EVJESPRQ	constant
delimiter	1	F	blank	constant
Application name	16	V	variable	ADNAME
delimiter	1	F	blank	constant
Workstation name	4	F	NV*nn*	WSNAME
delimiter	1	F	blank	constant
Operation no	3	V	1 – 255	OPNO
delimiter	1	F	blank	constant
Subsystem name	8	V	variable	JOBNAME
delimiter	1	F	blank	constant
Request	8	V	variable	first field in TXTOP
delimiter	1	F	blank	constant
Parameter 1 (optional)	*	V	variable	second field in TXTOP
delimiter	1	F	blank	constant
Parameter 2 (optional)	*	V	variable	third field in TXTOP

Note: The length of Parameter 1 or 2 is from 1 to 8 characters.

Table 10 shows the request buffer layout for non-subsystem, user extension (UXaaaaaa) operations:

Table 10. Request Buffer Layout for Non-Subsystem, User Extension (UXaaaaaaa) Operations. Length represents the maximum length if format is variable.

Field	Length	Format Fixed/Variable	Value	Obtained From
Request ID	8	F	EVJESPRQ	constant
delimiter	1	F	blank	constant
Application name	16	V	variable	ADNAME
delimiter	1	F	blank	constant
Workstation name	4	F	NV*nn*	WSNAME
delimiter	1	F	blank	constant
Operation no	3	V	1 – 255	OPNO
delimiter	1	F	blank	constant
Subsystem name	8	V	variable	JOBNAME
delimiter	1	F	blank	constant
Request	24	V	variable	TXTOP

Any parameter with a variable length is left-adjusted and all trailing blanks are ignored.

Chapter 15. Guidelines for User-Written Operations

OPC Automation allows two types of user-supplied extensions for implementation of functions beyond those provided by OPC Automation. These facilities provide support for the following types of user-supplied modules:

- A non-SA z/OS command or function that performs an action for a subsystem known to SA z/OS.
- An independent user-supplied function which is scheduled for the user. This type of function uses OPC Automation as a communications vehicle between OPC and the user-supplied module. A relationship is not required with any SA z/OS-defined subsystems.

The following sections describe an overview of each of these types of user-supplied modules and provide examples of each module's possible use.

User Functions Related to an SA z/OS-Defined Subsystem

OPC Automation provides support for stopping and starting SA z/OS-defined subsystems. Certain environments require you to issue a command or to perform a function outside the scope of SA z/OS. This may include a situation where a system command needs issuing or where a user-written function needs to perform a logical decision.

For example, you may need to issue a system command before taking action on a subsystem. If you always issue this command, specify it as part of the startup sequence in the SA z/OS policy database. However, since you may not need to use this command under certain conditions, OPC can initiate a user-supplied module to perform the command. You can split the startup sequence with the system commands, so OPC executes them separately from the subsystem startup commands. If this is the case, define each command sequence to OPC as an operation. Using scheduling parameters, such as specific types of days, you can include or exclude certain operations.

For example, consider a subsystem which normally runs on a specific processor. On weekends, you use this processor for testing purposes and move the application to another, perhaps smaller, processor within the same complex. On the days that the application needs moving, you need several VTAM® VARY commands to start the VTAM application statements.

In OPC, you can define an extra operation or application which runs on the first free day of each period, and another which runs on the first working day of each period. OPC calls the CLIST containing the VTAM commands. This allows the issuing of the appropriate VARY commands when needed before you start the application subsystem on the correct processor.

Triggering a user-written CLIST provides another example. This determines if all users of a specific application are logged off before issuing the commands to take down the subsystem.

Flow of Control

In a situation where a non-SA z/OS command needs issuing, specify the user CLIST or command processor in the OPCACMD entry of the subsystem. In

response to the request, instead of issuing an SA z/OS command, OPC Automation passes control to this user CLIST or command processor. All information available is made accessible to the user-supplied module. If the user-supplied module does not trigger a status change of the subsystem and returns control to OPC Automation synchronously, you are responsible for completing the operation. This should be done by calling OPCACOMP once the results of these commands are analyzed. The OPCACOMP module ensures that actions are accomplished in the correct sequence, does some housekeeping, updates the SA z/OS status file, and calls the OPCAPOST command processor to return the specified completion code to OPC; for more details see "Implementing Completion of a Request."

Figure 64 shows the flow including the user responsibilities.

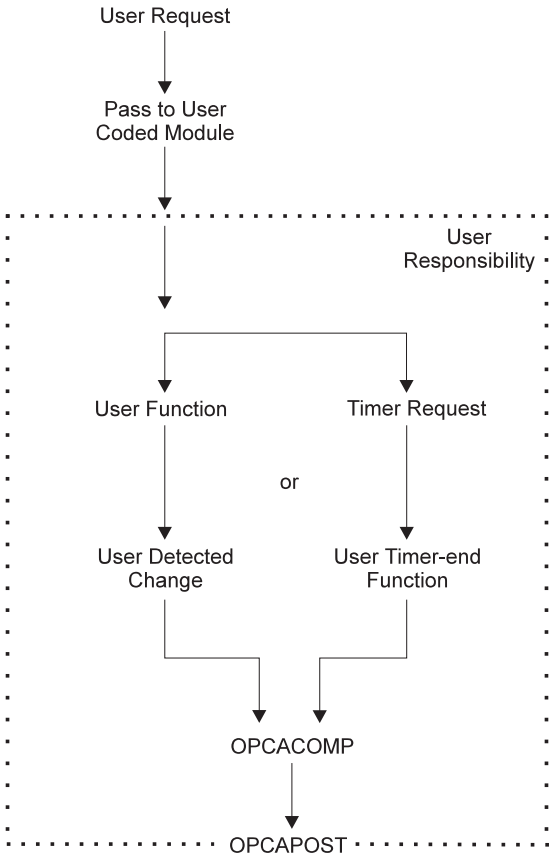


Figure 64. Request Flow for a Subsystem-Related User Function

To simplify implementation, you may plan to only use the timer function or only the detection of the completion of the command. If you use only the event-driven method, then consider what happens if the anticipated event fails to occur.

Implementing Completion of a Request

The general mechanism for executing requests for subsystems that have an OPCA entry coded in their MESSAGES/USER DATA policy item is as follows:

1. The request (with one or two optional parameters) and the job name of the OPC operation are passed to OPC Automation.
2. OPC Automation identifies the SA z/OS definition of the subsystem through the job name of the OPC operation.

3. OPC Automation retrieves the expected result, the timer interval, and possibly a timer name, for the respective request/parameter combination from the OPCA entry of the subsystem.
4. OPC Automation then checks if an OPCAPARM entry is present and if that entry contains a timer module name.
5. If a timer module is specified, OPC Automation sets the timer with this timer module. Otherwise, it uses a standard timer module (EVJESPTE).
6. OPC Automation issues the command that is specified in the OPCACMD entry.

After the command has been issued, two things remain to be done:

- If the command was executed successfully, the status file record for this subsystem must be updated.
- The (positive or negative) result of the request must be posted back to OPC.

How this is done, depends on the type of command specified in the OPCACMD entry.

Using OPC Automation Standard Modules

If the command specified in the OPCACMD entry triggers a status change of the subsystem to RUNNING, UP or AUTODOWN (no matter whether it is a user-written command or an SA z/OS standard command), you can leave it to OPC Automation to perform these two tasks. The modules responsible for this are the status change module (EVJESPSC) and the standard timer module (EVJESPTE), already mentioned in step 5 above; two flags, a completion flag and a timer flag, ensure that both modules are not active at the same time. The two standard modules operate as follows:

1. The *status change module* is called whenever the status of the subsystem changes. It first checks the timer flag. If that flag is set, EVJESPSC terminates at once. If the flag is not set, EVJESPSC checks if the status change is the result of an OPC request, and if the new status is identical to the expected result as specified in the OPCA entry. When both conditions are satisfied, the status change module assumes that the request was successfully executed and
   a. purges the timer defined in the OPCA entry,
   b. sets the completion flag in the status file record for this subsystem,
   c. actualizes further fields of the status file record, and
   d. posts the result of the request back to OPC.
2. The *timer module* (EVJESPTE) is called after the timer set in step 5 has expired (except when you have specified your own timer module in the OPCAPARM entry). It first checks the completion flag. If that flag is set, EVJESPTE will terminate at once. If the completion flag is not set, EVJESPTE sets the timer flag and compares the current state of the subsystem with the expected result of the OPCA entry. If both are compliant, the timer module assumes that the request was executed successfully; it updates the status file accordingly and posts a positive result back to OPC. If they are not compliant, it only posts the failure of the request back to OPC.

Programming your own Completion Routines

If you specify a command in the OPCACMD entry that does not change the status of the subsystem to UP, RUNNING or AUTODOWN, then you cannot use the standard modules for completing the request. In this case, you must perform the update of the status file and the posting of the result to OPC. You can do that in the command module (specified in the OPCACMD entry) or in a user-written

timer module (specified in the OPCAPARM entry) or in both. The user-written timer module is called by OPC Automation in the following format:

```
MODULE_NAME subsystem_name expected_result OPC_application_ID
            OPC_workstation_ID request_sequence_number
```

To simplify completion of a user-written module, OPC Automation provides the following facilities.

The OPCACOMP Command: This command, which is described in more detail in "OPCACOMP" on page 124, updates the status file and posts the result of the request back to OPC.

In particular, OPCACOMP first checks if the timer flag is set. If so, it will terminate at once. If not, it will

1. set the completion flag in the status file record of the subsystem for which it is called,
2. actualize further fields of the status file record, and
3. post the result of the request back to OPC.

The main difference between OPCACOMP and the standard modules (EVJESPSC and EVJESTPE) is that OPCACOMP does not check if the current status of the subsystem is in agreement with the expected result. Rather, it requires the (positive or negative) result of the request as one of its input parameters, and usually simply forwards this result to OPC. Thus, a user-written module must itself decide whether or not the request was executed successfully. It can then pass that information to OPCACOMP in order that the request be completed in an orderly manner.

One of the input parameters for OPCACOMP is the sequence number of the current request (see "OPCACOMP" on page 124). OPC Automation provides this and other information in some task global variables. Note, however, *that it will do this only when you have specified a timer module in the OPCAPARM entry*. You can specify a user-written timer module or the EVJESPTE standard module in the OPCAPARM entry. The following section describes the information contained in the global variables.

The &EHKVAR7, &EHKVAR8, and &EHKVAR9 Variables: When you supply a timer check module in the OPCAPARM entry (third value of the **Value Returned** field) OPC Automation sets some task global variables as follows:

&EHKVAR7 This variable contains the expected status, the timer interval, and the timer id as specified in the OPCA entry. The values are separated by commas.

&EHKVAR8 This variable contains the string 'OPC'.

&EHKVAR9 This value contains the subsystem name, the sequence number, the name of the timer check module and the domain, separated by commas. This is also known as the Requestor ID block; see "Requestor ID Block (&EHKVAR9)" on page 132.

Do not modify the information in the task global variables. OPC uses information in &EHKVAR7 if the timer is purged. The SA z/OS problem determination uses information in &EHKVAR8. In order to call OPCACOMP, the sequence number of the current request must be known; this number is stored in &EHKVAR9.

Non-Subsystem Operations

Operations of this type, containing requests named UX*xxxxxx*, allow you to perform commands that are independent of a specific subsystem. Figure 65 shows the flow for these types of operations.

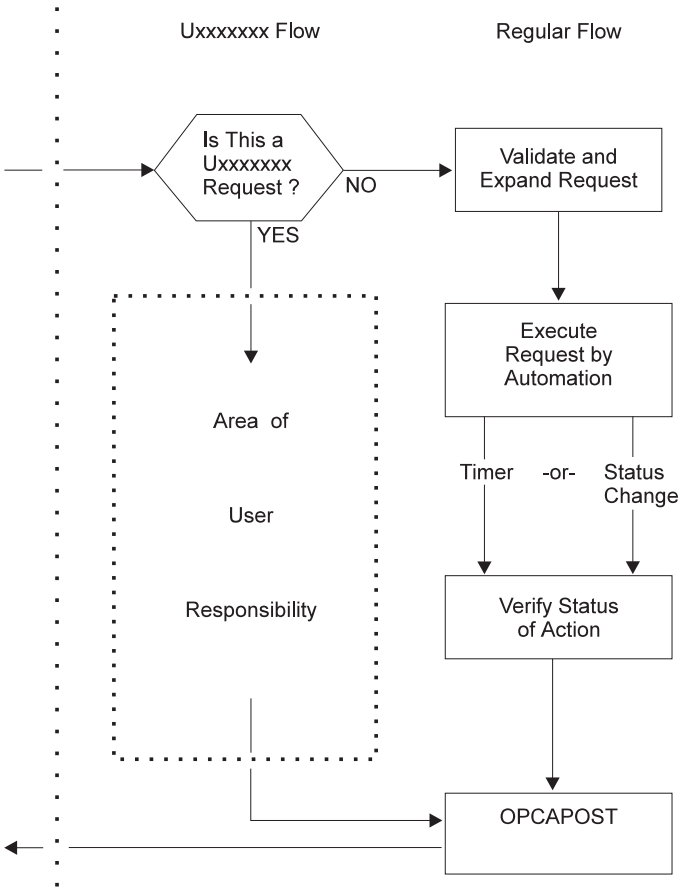


Figure 65. User Exit UXxxxxxx Flow

OPC Automation uses this type of exit for several purposes. At any point in the production cycle, OPC Automation allows you to invoke a user CLIST or procedure that can interact with system resources, such as the storage management subsystem.

Let's consider an example. Suppose, in a specific application flow within OPC, return codes show action that is taken by operations. When a specific job in this application completes, one of several user completion codes can result.

- A completion code of 0 indicates that application processing is to continue to the next operation.
- A user completion code of 50 indicates that the next two operations are skipped.
- A user condition code of 70 indicates that the application is completed at this operation.

Any other completion codes are treated as errors. Figure 66 on page 140 shows the subject operations in this application, and the desired flow of control on the basis of the condition codes of the job that runs as part of the CPU_20 operation.

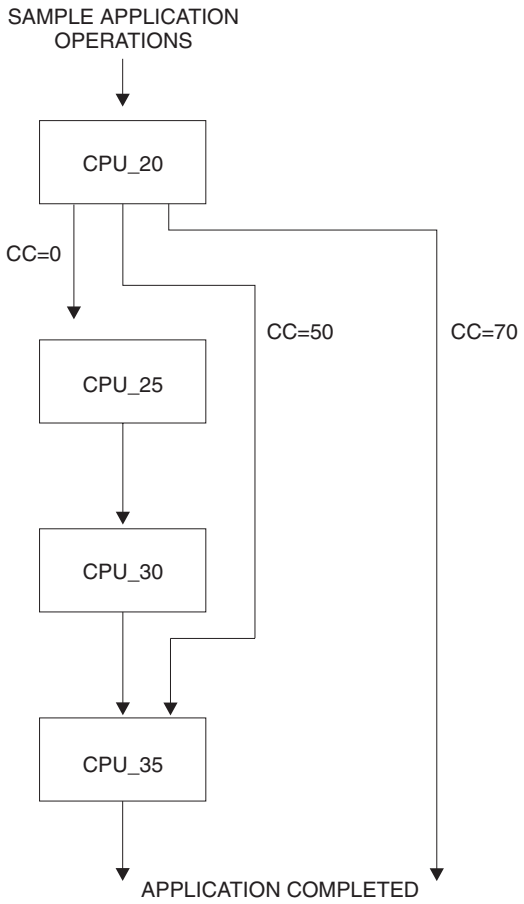


Figure 66. Condition Code Driven Application Flow

In the preceding example, OPC handles all condition code situations, except 50 and 70, which it intercepts. OPC accomplishes this interception in several fashions, such as user code in a JJC error exit. This code could then drive OPC Automation with a user exit (UX*xxxxxx*) request. This request would pass to the specified NetView to a user-written task. This task could then use the OPCAMOD command to do a modify current plan to OPC for the application in question on the basis of the condition code received as part of the user exit request.

Flow of Control

When the name of a request starts with `UX`, OPC Automation assumes that the request is not related to a subsystem known to SA z/OS. As before, it expects to find an OPCACMD entry within a policy object that is identified through the **Job name** field of the OPC operation. However, if no match is found for USER E-T pair 'OPCACMD jobname', then OPC Automation will check for USER E-T pair 'OPCACMD OPCA', and if again no match is found, OPC Automation will check for USER E-T pair 'OPCACMD subsystem'. Although user exit processing is designed to be non-subsystem related, this approach provides flexibility for users who have jobnames that do not match subsystems names but still prefer subsystem-related processing.But there are two differences as compared to subsystem-related requests:

- The only keyword that is needed is OPCACMD. OPCA and OPCAPARM are ignored.The CMD attributes of the OPCACMD entry should have the following format:

  ```
  CMD=(UXxxxxxx,,'userfunc &EHKVAR1')
  ```

For &EHKVAR1, see "Parameters Passed to a User Exit."

- The policy object identified through the **Job name** field of the OPC operation should be a USER E-T pair (see "OPCACMD" on page 112).

For non-subsystem requests, OPC Automation immediately tries to issue the command specified in the OPCACMD entry. After issuing the command, the request module of OPC Automation terminates. It is up to the user function to determine whether or not the request was executed successfully. The user function should then call OPCAPOST (see "OPCAPOST" on page 130) with the corresponding completion code. This returns the control of the application processing to OPC. The samples contain a code template for a non-subsystem command (EVJERUX1).

Parameters Passed to a User Exit

When the request name begins with `UX` OPC Automation stores the complete request buffer in the &EHKVAR1 task global variable. This variable must be forwarded to the command as an input parameter, as indicated in the format description above.

In contrast to a subsystem operation, the request buffer for a non-subsystem operation contains the entire request in one field. For the request buffer in general, see "Request Buffers and OPC Automation Log Entries" on page 81; the format of the request buffer for 'UX' requests is described in Table 10 on page 133.

Interaction with CICS Automation

The following example shows how to use the CEMTPPI command of CICS Automation to open and close CICS files. The CEMTPPI command allows you to perform CEMT commands on any CICS subsystem. If CICS Automation is not installed, then you can perform a similar function using the MVS MODIFY command from a NetView CLIST. First, you need these requests:

UXCICSOP Requests CICS to open a file.

UXCICSCL Requests CICS to close a file.

The example selects the CLIST names of CICSOPEN and CICSCLOS. Using these names, the format of the CMD attributes of the OPCACMD entry (see "OPCACMD" on page 112) is as follows:

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
                        UET Keyword-Data Specification          Row 3 from 3
Command ===>                                                  SCROLL===> PAGE

Entry Type : User E-T Pairs        PolicyDB Name   : SCENARIO
Entry Name : NONSUBS               Enterprise Name : TEST
UET Entry  : DUMMY                 UET Type        : OPCACMD

Action   Keyword/Data(partial)
________ CMD
         (UXCICSOP,,'CICSOPEN &EHKVAR1')

________ CMD
         (UXCICSCL,,'CICSCLOS &EHKVAR1')
****************************** Bottom of data ********************************




 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 67. OPCACMD Entry for Interaction with CICS

Figure 68 shows the definition of the operation text and other fields in OPC.

```
--------------------------------- OPERATIONS ---------------------  ROW 1 OF 1
Command ===>                                                  Scroll ===> PAGE

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details
Enter the PRED command above to include predecessors in this list, or,
enter the GRAPH command to view the list graphically.

Application            : PAYMAINT         Payroll Master Update

Row  Oper      Duration Job name  Operation text
cmd  ws   no.  HH.MM
'''' NV04 015   0.01    DUMMY___  UXCICSCL CICS01 PAYROLL____
```

Figure 68. Defining Sample CICS Application in OPC

The example uses the CICS subsystem name and the file name as parameters to the request. These parameters are optional and flexible. Thus, the CICS name could also be passed through the **Job name** field. The REXX code for CICSOPEN and CICSCLOS is supplied in the samples as EVJRUX2 and EVJERUX3.

Interaction with IMS Automation

The following example shows how to use the IMSCMD of IMS Automation to start and stop databases in IMS. The IMSCMD command allows you to perform IMS MTO commands on any IMS in the system. Other IMS commands could be imbedded into IMSCMD and incorporated in NetView CLISTs you write yourself, using similar logic to that shown in the EVJERUX4 and EVJERUX5 CLISTs supplied with the samples. If IMS Automation is not installed, then you can perform similar function by replying to the outstanding reply ID of the IMS you wish to communicate with from a NetView CLIST you write yourself. First, you will need these requests:

UXIMSSDB Requests to start a database.

UXIMSPDB Requests to stop a database.

The example selects the CLIST names EVJERUX4 and EVJERUX5. Using these names, the format of the CMD attributes of the OPCACMD entry (see "OPCACMD" on page 112) is as follows::

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
                        UET Keyword-Data Specification           Row 3 from 3
Command ===>                                                   SCROLL===> PAGE

Entry Type : User E-T Pairs        PolicyDB Name   : SCENARIO
Entry Name : NONSUBS               Enterprise Name : TEST
UET Entry  : DUMMY                 UET Type        : OPCACMD

Action   Keyword/Data(partial)
________ CMD
         (UXIMSSDB,,'EVJERUX4 &EHKVAR1')

________ CMD
         (UXIMSPDB,,'EVJERUX5 &EHKVAR1')
****************************** Bottom of data ********************************




 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 69. OPCACMD Entry for Interaction with IMS

Figure 70 shows the OPC definition of the operation text and other fields.

```
--------------------------------- OPERATIONS ----------------------  ROW 1 OF 1
Command ===>                                                 Scroll ===> PAGE

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details
Enter the PRED command above to include predecessors in this lis
enter the GRAPH command to view the list graphically.

Application            : CUSTMAINT        Customer DB update

Row  Oper      Duration Job name  Operation text
cmd  ws   no.  HH.MM
'''' NV01 020   0.02    DUMMY___  UXIMSSDB IMS05Z_________
```

Figure 70. Defining Sample IMS Application in OPC

The parameters of the request are the IMS subsystem name and the database name. The REXX code of EVJRUX4 and EVJERUX5 is supplied in the samples.

Chapter 16. OPC Automation Operator Commands

The following commands are used with OPC Automation:

Table 11. OPC Automation Commands

Command	Description
DFCRIT	Sets a critical SDF message. See "DFCRIT" on page 147.
DFUPDT	Sets or resets an SDF message alert based on message suffix. See "DFUPDT" on page 148.
EVJESPIN	Normally used only during initialization by SA z/OS. The operator can use this command manually to create a new status file record or resynchronize an existing one. The actions available with this command are INIT, RESET, SYNC, and CREATE. See "EVJESPIN — Initialization" on page 149.
OPCA	Displays the OPC Automation Main Menu (EVJT0000) that lists the most commonly used commands and provides access to the tutorials. See "OPC Automation Main Menu and Tutorials" on page 146.
OPCACMD	Displays the OPC Automation: Display or Modify OPC data (EVJKAC01) panel that is used to interact dynamically with OPC. See "OPCACMD — Interacting Dynamically with OPC" on page 149.
OPCAQRY	Lists pending NetView operations. See "OPCAQRY — Display Status of Operations" on page 151.
OPCAPOST	Posts an operation in OPC from SA z/OS. See "OPCAPOST — Posting an OPC Operation from SA z/OS" on page 154.
SRSTAT	Determining status of OPC special resources. See "SRSTAT — Determining OPC Special Resource Status" on page 155.

OPC Automation Main Menu and Tutorials

Type **OPCA** on the command line. After you press ENTER, OPC Automation displays the **SA/OPC – Main Menu**, as shown in Figure 71.

```
EVJT0000                  SA/OPC - MAIN  MENU

   TYPE    COMMAND             DESCRIPTION                     TUTORIAL

                               SA/OPC Automation                  1

    P      SDF                 Display facility additions         2

    P      OPCAQRY             Display status of operations       3
    L      OPCAPOST            Manually post status to OPC        4
    P      OPCACMD             Display and modify OPC data        5
    L      SRSTAT              Update special resource status     6




Note: Commands of type L (linemode) do not have an input panel
      and must be invoked with all required parameters specified.


Enter a Tutorial Number or Command ==>
          PF1= Help   PF2= End   PF3= Return
          PF6= Roll
```

Figure 71. SA/OPC – Main Menu

To obtain information on using this panel, enter the tutorial number adjacent to the command.

DFCRIT

Purpose

Use the DFCRIT command to add critical messages to the Status Display Facility. These messages are normally selected through the automation table, although you could invoke the CLIST from other places, such as user-written automation CLISTS.

Format

Syntax

DFCRIT *message text* **DFCRIT TYPE=***t*, *message text*

Parameters

t A 1-character value corresponding to an SDF CRITMSG type entry in the control file. A, E, I, and W are supplied with OPC Automation. Other values may be specified, provided an SDF CRITMSG*t* corresponds to that message.

message text
Message text added to the Status Display Facility critical message display panel.

```
IF MSGID='IOS001I' & TEXT=MESSAGE
  THEN EXEC(CMD('DFCRIT 'MESSAGE) ROUTE(ALL *));
```

Usage Notes

If the **TYPE=** parameter is not specified, *t* is set to the last character of the message ID, and the search is made. If no CRITMSG*t* entry is found, the CRITMSG value will be used.

Example

If you wish to see certain application messages in blue reverse video, add:

```
SDF CRITMSGU,CO=B,PR=500,HL=R
```

to your control file and call DFCRIT from the message table as follows:

```
IF MSGID='NORMAL' &amp. TEXT=MESSAGE
  THEN EXEC(CMD('DFCRIT TYPE=U,'MESSAGE) ROUTE(ALL *));
```

DFUPDT

Purpose

Use the DFUPDT command to insert display data for extensions to the Status Display Facility. The normal automation commands are issued to route this data to a focal point host in a distributed environment.

Format

Syntax

```
DFUPDT type,resource,component,ref_value,info,text
```

Parameters

type
Type used to get Status Display Facility parameters from the control file.

resource
Status Display Facility resource name.

component
Status Display Facility component name.

ref_value
Reference value used for the Status Display Facility entry. Used as a way to group related or duplicate entries. If not supplied, then use *resource*.

info
Information text that appears on the panel for this entry. If not supplied, then use *resource*.

text
Message or user text that appears in the detail panel for this entry. If not specified, then use *resource*.

EVJESPIN — Initialization

OPC Automation uses this command during initialization when SA z/OS is started. An operator can also use it to create a new OPC Automation status file record or to resynchronize an existing one.

Four command actions are available. The first, INIT, acts on all OPC Automation controlled subsystems, while the other three are available for the specified subsystem only.

EVJESPIN CMD=*action***,SUBSYSTEM=***subsystem*

CMD=*action* The command to execute where *action* is one of the following:

INIT Forces an equivalent action to that taken during normal initialization. Do not specify a subsystem parameter for this parameter. To properly understand the action of this command parameter and for a complete overview of the initialization process, refer to the initialization topic in "Initialization Module (EVJESPIN)" on page 79.

RESET
Resets the timer and comp flags to a null value, and unlocks a specific subsystem after a user error is detected and corrected. By resetting the timer and comp flags, OPC Automation again accepts requests from OPC.

SYNC Checks the last completed status field in OPC Automation and ensures that the specified subsystem status agrees whether the last completed status is UP or DOWN. OPC Automation resets the timer and comp flags to null. This marks the action as completed. Use SYNC when manual action is necessary to stop or start a subsystem.

CREATE
Creates a new OPC Automation status file record from the control file definition for a specified subsystem. Use this function to refresh the control file dynamically with new OPC Automation definitions, when NetView is not recycled.

subsystem The subsystem initialized. Do not specify a subsystem with INIT.

OPCACMD — Interacting Dynamically with OPC

OPCACMD will invoke INGOPC. For details of the INGOPC command, see *System Automation for z/OS Operator's Commands*.

DFTSOU

Purpose

Use the DFTSOU command to update the status of TSO Users in SDF or NMC.

Format

Parameters

UPDATE

A single TSO user status is updated according to the message that triggered this command.

This is the default option if none are supplied.

SCAN A Display TSO users command is issued and the results are used to set the status of TSO users in SDF and NMC.

Usage Notes

This command can be used in two ways.

Firstly the UPDATE parameter can be used when the command is placed in an Automation Table to track the status of TSO users.

Secondly the SCAN parameter can be used in either the UP Message command policy item, or the ACORESTART message command policy for TSO. This would cause a refresh of the status of TSO users should the SA z/OS NetView be restarted.

Example

Used in an Automation Table:

```
IF MSGID = 'IEF125I'
 THEN EXEC(CMD('DFTSOU ') ROUTE(ONE *));
IF MSGID = 'IEF126I'
 THEN EXEC(CMD('DFTSOU ') ROUTE(ONE *));
IF MSGID = 'IEF450I'
 THEN EXEC(CMD('DFTSOU ') ROUTE(ONE *));
```

Used in the TSO ACORESTART command:

```
DFTSOU SCAN
```

OPCAQRY — Display Status of Operations

The OPCAQRY command displays the status of OPC Automation operations.

To use this command, type the following on any NetView command line:

OPCAQRY

After you press ENTER, OPC Automation displays the **SA/OPC Automation Operation Status Display** panel, as shown in Figure 72.

```
EVJKCGAA      SA/OPC - Automation Operation Status Display
                                                              Date: 08/17/00
Valid Actions: B Browse  D Delete  R Reset                    Time: 11:43:00

Act  Job     Application    Request     Date     Time     Status
 _   CX06AA                           05/24/00   08:44    No request
 _   DBSYS1  TEST2          STOP       05/29/00   14:16    Complete
 _   RMF     RMFMAINT       START      05/30/00   19:27    Complete
 _   SUBSYS1                          ********   *****    No request









Command ==>
F1= Help      F2= End      F3= Return               F5= Refresh  F6=Roll
```

Figure 72. SA/OPC - Operation Status Display Panel

The panel in Figure 72 shows the status of requests from OPC Automation in NetView. It also provides a convenient place to delete unused records or to reset an operation in the event of problems.

The fields shown on the panel in Figure 72 are defined as follows:

Act Action field, used for browsing, deleting, or resetting the status file record. See "Selecting Actions" on page 152 for more discussion on this topic.

Job Name Job name from OPC, typically used to represent a subsystem.

Application OPC application requesting the operation.

Last requested action
Action specified in the OPC operation description text.

Date Date the request was received.

Time Time the request was received.

Status Status of the request in NetView, either complete, incomplete, timeout, or no request. A status of timeout indicates that the operation is marked in error because it did not complete within the time limit set by the system programmer in the OPCA code entry. A status of incomplete indicates that the operation did not achieve

the expected status set by the system programmer in the same entry. Complete and no request are considered normal statuses.

Selecting Actions

Select one of the following valid actions for any operation listed on the OPC Automation Operation Status Display panel, as shown in Figure 72 on page 151.

B Browse. Selecting the browse action allows the user to examine a record on OPC Automation Operation Status panel as shown in Figure 73 on page 153.

D Delete. This action deletes the record from OPC Automation status file.

A confirmation pop-up will appear, as follows:

```
..... DELETE CONFIRMATION .....
:                               :
:                               :
:       RECORD ID... TESTSYS    :
:                               :
:     _ ENTER 1 TO DELETE RECORD,:
:       PRESS F3 OR F12 TO KEEP RECORD.:
:.......................................:
```

R Reset. This clears flags in the record for reuse by performing an EVJESPIN CMD=RESET. This is useful for recovering operations that did not process normally. See “EVJESPIN — Initialization” on page 149 for more details.

After you type **B** (browse) in the Act field of the OPC Automation Operation Status Display panel shown in Figure 72 on page 151 and press ENTER, OPC Automation displays the OPC Automation Operation Status Detail panel, as shown in Figure 73.

```
EVJKCGAA            SA/OPC - Operation Status Display
                                                                Date: 08/17/00
 Status file record display for CX06AA                          Time: 14:44:00


 EHK170I OPCA RECORD DISPLAY FOR: CX06AA
 EHK171I ID= CX06AA          , TYPE= OPCA, OPID= RICK
 EHK172I LAST COMPLETED STATUS=  , LAST SEQUENCE NUMBER= 0000
 EHK173I TIMER FLAG=  , COMPLETION FLAG=
 EHK174I CURRENT SEQUENCE NUMBER= 0000, CHECK MODULE=
 EHK175I EXPECTED STATUS=         , TIMER INTERVAL=         ,TIMER ID=
 EHK176I ADNAME=                 , WSNAME=
 EHK177I OPNUM=   , JOBNAME=          , DATE= 08/17/00 ,TIME= 14:33
 EHK178I REQUEST=          , PARM1=          , PARM2=
 EHK002I END




Command ==>
F1= Help      F2= End      F3= Return               F5= Refresh  F6=Roll
```

Figure 73. OPC Automation: Operation Status Detail Panel

OPCAPOST — Posting an OPC Operation from SA z/OS

This command is used by SA z/OS to inform OPC of status changes. This is accomplished by the OPCAPOST command processor, which is normally used internally in OPC Automation. Although you can issue OPCAPOST as an operator command, operators should use OPCACMD, if possible. OPCACMD provides a full-screen interface to OPC and dynamically acknowledges the action, rather than OPCAPOST.

If you determine that you must use the OPCAPOST command, refer to "OPCAPOST" on page 130.

SRSTAT — Determining OPC Special Resource Status

This command lets you manipulate the status of OPC special resources. Status is returned via messages. The format of this command is:

SRSTAT *srname*,**SUBSYS=***subsys*,**AVAIL=Y|N**

srname
Special resource name — up to 44 characters.

Note: The special resource name must be enclosed in single quotes if it contains any spaces or commas.

subsys
Subsystem ID — 4 characters.

AVAIL=Y/N
Availability indicator.

Example:

```
SRSTAT EOD.CICSPRD1.TRANS,SUBSYS=OPCT,AVAIL=Y
```

In the above example, end-of-day transactions are required to complete before production work can begin. SRSTAT is executed when the transactions are complete. The special resource name EOD.CICSPRD1.TRANS is used to trigger OPC/ESA applications that are able to run when the transactions are finished. A number of applications are added to the current plan.

The variable used for *subsys*, OPCT, is the name of the tracker subsystem. Only in this command is the tracker subsystem name required.

Chapter 17. Resynchronization and Recovery Considerations

OPC Automation combines the capabilities of two different subsystems, OPC and SA z/OS, which may reside over several systems. As a result, a failure or a scheduled interruption of services with one of the subsystems, processors, or telecommunications facilities may occur and prevent OPC Automation from processing operations. Further complications arise by shifting work load across multiple system images, either for scheduled workload balancing or as part of a recovery situation.

In such a case, a loss of synchronization can occur between the OPC schedule and the OPC Automation components in NetView. When this happens, you may need a manual process to examine the OPC schedule, to ensure that OPC Automation in NetView is performing actions as required and, in some cases, to resynchronize OPC and OPC Automation.

During a loss of contact or a failure with either OPC or NetView, OPC Automation's facilities invoke and restore the environment as it was before the failure so that event scheduling can pick up where it left off. This results in a satisfactory resolution and manual resynchronization is not required. See "Automated Recovery Functions" on page 161 for additional information.

Generally, the longer the outage, the more likely that resynchronization is required. This depends on the number of scheduled events that are not processed. A long outage during the day may have a smaller synchronization impact than a shorter outage during a period when many online facilities are started or shut down.

Examples and Scenarios

This section describes possible scenarios for resynchronization and recovery.

Loss of Contact Between OPC and OPC Automation

Under most situations, once OPC Automation re-establishes connectivity, its automatic recovery schedules requests for execution that it could not execute prior to connectivity. In some situations you may need to intervene manually, such as when the request is no longer valid. The following sections discuss several reasons for manual intervention.

Taking Action Manually on the Target System

The scheduler acts on the ended-in-error notification, requesting that an operator with access to the target system perform the required operation manually. The operation requested by OPC completes, and the scheduler manually updates it to a completed status, enabling OPC to continue processing.

You should issue EVJESPIN CMD=SYNC.

Taking Action Too Late

The remaining processing window is too small to allow an operation to occur. For example, an online system may require a certain amount of time to initialize. If this amount of time is close to the scheduled shutdown time, you probably should override the request and complete the operation manually.

You should issue EVJESPIN CMD=RESET.

Queuing Several Actions for a Specific Target Subsystem

Rather than not having enough time for a system initialization as in the previous example, the outage may have lasted long enough for a specific subsystem to receive several queued operations that frequently conflict. For example, an online task may have a start request with an ended-in-error status because of connectivity problems. This same task may also have a stop request already due for scheduling. If you allow automatic recovery for this application, the subsystem would start, but an immediate shutdown would follow.

Complete these operations manually. You should issue EVJESPIN CMD=RESET.

Backup on a Different Processor

If you perform a backup using a different processor, pay special attention to ensure that you properly restore the work load on the new system. Depending on the backup structure, you need to follow one of several different procedures discussed in the next sections.

Full Takeover onto a Standby System at the Same Site

This is the simplest type of backup. It becomes the same as a single-system recovery if the data is also available. OPC Automation uses the information in the status file to restore the various subsystems to the pre-backup status.

Full Takeover onto a Standby System at a Different Site

If the status file is not available, restructure the environment manually. Examine pertinent applications that control this specific NV*nn* workstation. The last completed NV*nn* operation requires manual triggering. You can achieve this with the OPCACMD function, allowing the NetView operator direct access to the relevant applications. Although, at times, you may find it necessary to perform specific operations manually, in most cases, resetting an operation with EVJESPIN or restarting an application produces the desired effect.

Takeover onto a Working System

In most situations, the takeover is onto a working system and is restricted to certain critical applications. The previously discussed considerations as to whether the system is at the same site or at another site apply to this situation. Since not all applications are restarted on the backup system, several new considerations become important. Certain applications need cancelling before you can achieve a restoration of services. Some situations can result in duplications, such as with subsystems like TSO, which are frequently found on every MVS system. Although normally this is controlled by SA z/OS, OPC Automation can control this. Here, duplicate applications need cancelling for the backup period.

During the backup period, use one of these two methods to run OPC Automation:

- Add the new work load to the resident NetView by changing the NV*nn* workstation entry in the platform control file to point to the same NetView domain ID.
- Start another copy of NetView with the NetView domain ID used in the failing system.

The consideration of which method to use becomes important once you restore the original configuration.

With the single NetView solution method, you need to resolve the subsystems manually since their original identity is lost. With the extra NetView solution method, you can stop the subsystems controlled by the extra NetView by shutting it down. This simplifies the restoration process, requiring almost no manual intervention.

In both cases, restoring the environment follows similar procedures used to backup a host onto a standby backup system.

Long Term Outage

You must manually intervene when the outage duration is more than a single scheduling cycle. This type of recovery is confusing since many applications are shown as late in the OPC plan. Carefully review these applications since some of the them still need scheduling, while others need to be cancelled. For applications that need scheduling, certain operations involving the online portion need cancelling or holding. To ensure success, this type of recovery needs precise planning and monitoring. Otherwise, you can use the scenarios previously outlined.

Example Using Doubly-Defined NetView Domain IDs

The example in Figure 74 shows the WORKSTATION DOMAINS entry for a 4-processor environment:

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing              Row 1 to 6 of 21
Command ===>                                                  SCROLL===> PAGE

Entry Type : Workstation domainID  PolicyDB Name   : SCENARIO
Entry Name : SAMPLE_01             Enterprise Name : TEST

Subsystem  : OPCA_DOMAINID
Message ID : DOMAINID

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

    Code 1          Code 2          Code 3           Value Returned
NV00_____________ _____________ _____________ NVTOR_____________________
NV01_____________ _____________ _____________ XBAOF_____________________
NV02_____________ _____________ _____________ AOFT5_____________________
NV06_____________ _____________ _____________ AOFS6_____________________
_____________ _____________ _____________ _____________________________
_____________ _____________ _____________ _____________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 74. Mapping of NVxx Workstations to Domain IDs

Here, NV00 maps to the NVTOR NetView domain ID, and the NV01 workstation maps to the XBAOF NetView domain.

Under normal circumstances, each NetView domain ID represents a processor with its MVS operating system and a unique NV*xx* general automatic reporting workstation. For situations such as testing, backup, or work load management, this relationship needs no maintenance.

In the previous example, both NV00 and NV06 OPC-defined workstations represent their own specific NetView domain, NVTOR and AOFS6, respectively.

Assume that the system represented by NVTOR has failed, and you make the decision to shift the work load to the AOFS6 system. You can accomplish this by changing the domain ID in the first CODE statement from NVTOR to AOFS6. This would imply that the AOFS6 domain is associated with two OPC workstations, namely NV00 and NV06.

If you accomplish this change without altering the OPC definitions, you must reload the SA z/OS control file. The scheduler or operator needs to ensure that the OPC-defined applications that are running in the failed system are restarted on the backup system. Because the SA z/OS status records are on the failed system, the scheduler manually recovers the failed environment. Once resynchronization completes, any new scheduled event originally intended for the NetView domain ID NVTOR automatically is scheduled for AOFS6. After you resolve the problem on the NVTOR system, perform the previous scenario in reverse order to restore the system to its original configuration.

When double definitions of this type are used, exercise caution to avoid creating conflicting requests for specific subsystems. For example, if RMF exists in the AOFS6 domain, OPC can then schedule a shutdown request on NV00 and a start request on NV06.

Automated Recovery Functions

Only a small portion of OPC Automation resides in the OPC address space in OPC user exits. These exits communicate to the rest of OPC Automation which resides in the NetView address space. A loss of contact results if a NetView address space becomes unavailable or if OPC Automation code in NetView is unavailable. Also, a communication failure can prevent a request from reaching its ultimate destination.

OPC Automation automated recovery determines which operations are affected by a specific loss of communications. It also determines when the connectivity and availability of a given target NetView is corrected and the NV*nn* operation is reset to the ready state. This redrives the EQQUX007 exit, allowing it to re-create the original request.

OPC Actions in a Loss of Contact Situation

The EQQUX007 exit or an intermediary NetView with an ended-in-error status and a return code of Sxxx reports a connectivity loss to OPC. OPC does not schedule any dependent operations and shows an error on the operations ended-in-error Status Display Facility panel. OPC takes no further action until the connectivity is restored and OPC Automation automatic recovery is invoked.

The operator or scheduler can manually override the ended-in-error status, thus allowing the application to continue or cancelling it.

OPC Automation Actions in a Loss of Contact Situation

If loss of connectivity to NetView is detected in the OPC Automation portion of the EQQUX007 exit (using the EQQUSINT function directly), then OPC Automation posts an ended-in-error status with a UNTV return code. OPC Automation uses this mechanism because the EQQUX007 exit cannot directly modify operation status.

If the request is received in NetView, but OPC Automation cannot propagate the request to the appropriate target system, OPC Automation uses the OPCAPOST function to post the operation as ended-in-error with an S999 return code.

Glossary of Terms

The intent of this glossary is to define terms as TME 10 OPC uses them. However, where applicable, terms are taken from the *IBM Dictionary of Computing*, New York; McGraw-Hill, 1994. These terms are marked by an asterisk (*). Unless otherwise noted, the definitions below apply equally well to OPC/ESA and TME 10 OPC.

A

actual duration. At a workstation, the actual time in hours and minutes it takes to process an operation from start to finish.

APAR. Authorized program analysis report. A report of a problem caused by a suspected defect in a current unaltered release of a program.

all workstations closed. A user defined interval during which *all* OPC's workstations are not available for running applications under OPC's control.

Note: All the workstations could be either shut down *or* simply not available to OPC.

application. (1) A group of related operations performed together to satisfy a specific end user task. (2) A measurable and controllable unit of work that completes a specific user task such as the running of payroll or financial statements. The smallest entity that an application can be broken down into is an operation. Generally, several related operations make up an application.

application description. A database description of an application.

application ID. The name of an application. Examples: Y1976, Payroll.

arrival (A). Status of an operation that indicates it is waiting for the input to arrive before processing.

authority. The ability to access a protected resource.

authority group. A name used to generate a RACF resource name for authority checking.

automatic events. Events recognized by or triggered by an executing program. Automatic events are usually generated by OPC job tracking programs but may also be created by a user-defined program.

automatic reporting workstation. A workstation that reports events (the starting and stopping of operations) in real time to OPC, such as a processor or printer.

automatic job recovery. An OPC function which allows you to specify, in advance, alternative recovery strategies for applications or operations ended in error.

availability. * The degree to which a system (and in OPC, an application) or resource is ready when needed to process data.

B

batch loader. An OPC batch program you can use to create and update information in the application description and operator instruction databases.

bracketed DBCS. A MIXED format field consisting of a DBCS part only, that is, DBCS characters enclosed by a shift-out/shift-in control character pair.

browse. An ISPF/PDF dialog function that manages data for display only. This function lets the user view but not change data.

C

CP. Current plan.

calendar. The data that defines the operation department's processing schedule in days and periods.

capacity. The actual number of parallel servers and workstation resources available during a specified open time interval.

capacity ceiling. The maximum number of operations a workstation can handle simultaneously.

case code. A code in the automatic job recovery function that represents a group of abend codes or return codes. Any code in the JOBCODE and STEPCODE parameters is considered a potential case code if defined as such in the case code macro.

closed workstation. A workstation that is unavailable to process work for a specific time, day, or period.

command. * A request from a terminal for the performance of an operation or the execution of a particular program. A character string from a source external to a system that represents a request for system action.

complete. Status of an operation indicating that it has finished processing.

completion code. An OPC system code indicating how the processing of an operation ended at a workstation.

complex of processors. A JES2 multi-access spool system or a JES3 system with more than one processor.

computer workstation. A workstation that performs MVS processing and usually reports status to OPC automatically. A processor when used as a workstation. It can refer to single processors or multiprocessor complexes serving a single job queue (for example JES2 or JES3 systems).

controller. The portion of TME 10 OPC or OPC/ESA that runs on the controlling processor and contains the tasks that manage OPC databases and plans.

critical path. The route within a network with the least amount of slack time.

current plan. A minute by minute schedule of each operation of an application. It reflects the current state of the operating environment showing the status of work completed and work still to be done.

current schedule. The database that contains the current plan information.

cyclic interval. The number of days in a cyclic period.

cyclic period. A period with a specific origin date and set frequency. A cyclic period can be broken down into two types:

- Those that include work and free days
- Those that include only work days.

Cyclic periods must always represent a fixed time period in days. For example, week (7 days).

D

daily plan. A set of plans that shows work that the operations department does on a particular day or shift. A list by day and application of all operations to be performed within the operations department.

default calendar. (1) A calendar that you have defined for OPC to use when you do not specify a calendar in an application description. (2) A calendar that OPC uses if you have neither specified a calendar in an application description, nor defined your own default calendar.

deadline. See deadline date and deadline time.

deadline date. The latest date by which an occurrence must be complete.

deadline time. The latest time by which an occurrence must be complete.

defined. An open day status which indicates that specific open time intervals exist for a workstation on a particular day.

dependency. A relationship between two operations where the first operation must successfully finish before the second operation can begin.

dialog. The user's online interface with OPC.

displacement. A number specifying 'Number of Days from Period Start' or 'Number of Days from Period End'. Sometimes called offset. See offset.

duration. The time an operation is active at a workstation.

E

edit. An ISPF/PDF dialog function that is used for editing text, collecting data, and modifying data.

end user. A person who uses the services of the data processing center.

ended in error (E). The OPC reporting status for an operation that has ended in error at a workstation.

error code. The system completion code or program return code for automatic reporting workstations. The code entered by the workstation operator for manually reporting workstations.

exclusive. The state of a special resource indicating that it is fully used by one operation and cannot be used simultaneously by other operations.

exclusive resource. A workstation resource that is solely used by one operation and cannot be shared with other operations.

expected arrival time. The time when an operation is expected to arrive at a workstation. It may be calculated by daily planning or specified in the long-term plan.

extend current period. An OPC function that allows the user to extend the current plan up to a maximum of 504 hours (21 days) from the current end date.

external dependency. A relationship between two occurrences where an operation in the first occurrence must successfully finish before an operation in the second occurrence can begin processing. See dependency.

external predecessor. The name given to the operation in the first occurrence of an external dependency that must finish before its external successor can begin processing.

external successor. The name given to the operation, in the second occurrence of an external dependency, that cannot begin until its external predecessor completes.

F

free day. A nonworking day.

free day rule. A rule that determines how OPC will treat free days when the application run day falls on a free day. The rule is as follows:

Excluded: Free days excluded; only work days are taken into account.

Included: Free days included; all days are taken into account, as follows:

(1) Run before the free day.
(2) Run after the free day.
(3) Run on the free day.
(4) Do not run on the free day.

G

general workstation. A workstation where activities, usually manual, and other than printing and processing, are carried out. Manual activities might be data entry or job setup. A general workstation reporting to OPC is usually manual, but can be automatic.

generic search argument. A portion of a key containing a generic search character which in OPC is an asterisk (*) or percent sign (%). The asterisk represents any string of characters and the percent sign any single character. Use with any portion of a key to search the database for items to be displayed as part of a listing. Examples: %ABC, A*C, A*.

H

host processor. * A processor that controls all or part of a user application network. * In a network, the processing unit in which the access method for the network resides.

highest return code. A numeric value from 0 to 4095. If this return code is exceeded during a job's processing, the job will be reported as ended in error.

I

incident log. An optional function available under the job completion checker.

input arrival. The user-defined date and time an operation or an application becomes ready for processing.

internal dependency. A relationship between two operations within an occurrence where the first operation must successfully finish before the second operation can begin.

internal predecessor. The name given to the operation of an internal dependency that must finish before its internal successor can begin processing.

internal successor. The name given to the operation of an internal dependency that cannot begin until its internal predecessor completes processing.

ISPF. Interactive System Productivity Facility.

interrupted (I). An OPC reporting status for an operation indicating that the operation has been interrupted while processing.

J

job. * A set of data that completely defines a unit of work for a computer. A job usually includes all necessary computer programs, linkages, files, and instructions to the operating system. In OPC, an operation performed at a CPU workstation.

job completion checker (JCC). An optional function of OPC that provides an extended checking capability of the results from CPU operations.

job control language (JCL). * A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

JES. Job Entry Subsystem.

job entry subsystem (JES). * A system facility for spooling, job queuing, and managing I/O.

job setup. The preparation of a set of JCL statements for a job at an OPC workstation you defined for this purpose.

job submission. An OPC process that presents jobs to MVS for running on an OPC defined workstation at a time specified in the daily plan.

JS. The JCL repository data set.

K

keyword. * A symbol that identifies a parameter. * A part of a command operand that consists of a specific character string (such as DSNAME=).

keyword parameter. * A parameter that consists of a keyword, followed by one or more values.

L

LTP. Long-term plan.

last operation. (1) An operation in an occurrence that has no internal successor. (2) The terminating node in a network.

latest start. The latest start day and time (calculated by OPC) for an operation that will allow all occurrences to meet their deadline.

layout ID. A unique name that identifies a specific ready list layout.

limit for feedback. See feedback limit.

local. * Synonym for channel-attached.

local processor. * In a complex of processors under JES3, a processor that executes users' jobs and that can assume global functions in the event of failure of the global processor. In OPC, a processor in the same installation that communicates with the controlling OPC processor through shared DASD communication.

long-term plan. A high-level schedule of processing activities for the forthcoming weeks and months. The scope of a long-term plan can be from one day to four years.

The long-term planning function produces a list of application occurrences identified by name, date, and run time for a specified planning period.

M

manual reporting workstation. A type of workstation reporting where events, once they have taken place, are manually reported to OPC. This type of reporting requires that some action be taken by a workstation operator. Manual reporting is usually performed from a list of ready operations.

mass updating. A function of the application description dialog where a large update to the application database can be requested.

modify current plan. An OPC dialog function used to dynamically change the contents of the current schedule to respond to changes in the operation environment. Examples of special events that would cause alteration of the current schedule are: a rerun, a deadline change, or the arrival of an unplanned application.

most critical application occurrences. Those unfinished applications that have a latest start time that is less than or equal to the current time.

N

node. * In a network, a point where one or more functional units interconnect transmission lines.

noncyclic period. A period that has a varying frequency for which you must define each origin date. Examples: month, payroll period, and quarterly.

nonreporting. A reporting attribute of a workstation which indicates that information is not fed back to OPC.

O

OPC/ESA. Operations Planning and Control/Enterprise Systems Architecture

occurrence. Each instance of an application in the long-term plan and current plan is called an occurrence.

An application occurrence is one attempt to process that application. Occurrences are distinguished from one another by run date, input arrival time, and application ID. For example, one application that runs four times a day is said to have four occurrences a day.

offset. A maximum of 12 positive and 12 negative values in the ranges 1 to 999 and -1 to -999 that indicate on which days of a calendar period an application shall run. See displacement.

OPC host. The processor where OPC updates the current plan database.

OPC local processor. A processor that connects to the OPC host or remote processor through shared event data sets.

open time interval. The time interval during which a workstation is active and can process work.

operation. An operation is a unit of work that is part of an occurrence and is processed at a workstation.

operation waiting for arrival. The status of an operation that indicates that the necessary input has not arrived at a workstation so that the operation can begin processing. This status is applicable only for operations without predecessors.

operation status. The status of an operation at a workstation.

An operation's status can be one of the following:

A Waiting for input to arrive.

R Ready for processing. All predecessors are complete.

***** Ready for processing. There is a nonreporting predecessor. All predecessors are complete but

one or more predecessors were executed at a nonreporting workstation.

S Started.

I Interrupted operation.

C Complete.

E Operation ended in error.

W Waiting for predecessor to complete.

U Undecided. The status is not known.

operator. * (ISO) A symbol that represents the action to be performed in a mathematical operation. * In the description of a process, that which indicates the action to be performed on operands. * A person who operates a machine.

option. A selection item on a menu panel in the OPC dialog.

origin date. The date on which a period (cyclic or noncyclic) starts.

P

panel. * A particular arrangement of presentation windows used to show information to the user. OPC uses only fixed-format panels.

parallel operations. Operations at workstations that are not dependent on one another and therefore can be performed simultaneously.

parallel server. The function that processes operations at a workstation, especially when there is more than one such function. See server.

parameter. * (ISO) A variable that is given a constant value for a specified application and that may denote the application. * A name in a procedure that is used to refer to an argument passed to that procedure.

pending application description. An application description which is incomplete and not ready for use in planning or scheduling.

period. A business processing cycle. A time period defined in the OPC calendar. They are used to describe when, and how often, applications are to run.

period name. A name of a period. Examples are week, month, quarter and fiscal period end.

period type. Periods are of two types: cyclic or noncyclic.

PDF. program development facility.

predecessor. An operation of an internal or external dependency that must finish successfully before its successor operation can begin.

printout routing. The ddname of the daily planning printout data set.

print workstation. A workstation that prints output and usually reports status to OPC automatically.

priority. A digit from 1 to 9 (where 1 = low, 8 = high, and 9 = urgent) that determines how OPC schedules applications to run. A number from 1 (low priority) to 9 (high priority) which establishes the importance of an application relative to other applications.

processor. * (ISO) In a computer, a functional unit that interprets and executes instructions. * A functional unit or part of another unit (such as a terminal or a processing unit) that interprets and executes instructions.

program interface. An OPC interface that allows a user-written program to issue various types of requests to the OPC subsystem.

Q

QCP. Query current plan.

R

RACF. Resource Access Control Facility.

read authority. A type of access authority that allows a user to read the contents of a data set, file, or storage area, but not to change it.

ready (R). The status of an operation indicating that predecessor operations are complete and that the operation is ready for processing.

ready list. A display list of all the operations ready to be processed at a workstation. Ready lists are the means by which workstation operators manually report on the progress of work.

recovery. See automatic job recovery.

remote processor. A processor connected to the OPC host processor by a VTAM network.

remote job tracking. The function of tracking jobs on remote processors connected by VTAM links to an OPC controlling processor. This function enables a central site to control the submitting, scheduling, and tracking of jobs at remote sites.

replan current period. An OPC function that recalculates planned start times for all occurrences to reflect the actual situation.

reporting attribute. A code that specifies how a workstation will report events to OPC.

rerun. An OPC function where an application or part of an application that ended in error can be run again.

rescale factor. A value from 0 to 100 used to reduce the new duration value by a given percentage amount.

return code. An error code issued by OPC for automatic reporting workstations.

row command. A dialog command used to manipulate data in a table.

run cycle period. A time frame defining the effective period and run days of a calendar period.

run day. The date on which an application is to run. It is expressed as a number relative to the start or the end of a run cycle period.

S

SAF. System Authorization Facility.

search argument. A value that is used to search the database for an item that is to be part of a displayed listing.

selection criteria. Search arguments entered on a list criteria panel in the dialog that limit the contents of a listing.

server. A program or device set up for a workstation to perform a service for that particular type of workstation. For example, an initiator is a server for a computer workstation. A printer is a server for a print workstation.

service functions. Functions of OPC that let the user deal with exceptional conditions such as investigating problems, preparing APAR tapes, and testing OPC during implementation.

shared DASD. Direct access storage device that can be accessed from more than one processor.

shared resource. A special or workstation resource that can be used simultaneously by more than one operation while the operation is processed at a work station.

slack. Used to refer to 'spare' time. Can be calculated for the critical path by taking 'Deadline less the Input Arrival less the Sum of Operation Durations'.

smoothing factor. A value between 0 and 100 that controls the extent to which actual durations are fed back into the application description database.

SMP. System Modification Program.

special resource. Resources that are not associated with a particular workstation but are needed to process work there.

splittable. Refers to an operation that can be interrupted while processing at a workstation.

standard. User specified open time intervals for a typical day at a work station.

status. The current state of an operation or an occurrence.

started (S). An OPC reporting status of an operation or an application indicating that an operation or an occurrence is started.

submit/release data set. A data set shared between the OPC host and a local OPC processor that is used to send job stream data and job release commands from the host to the local processor.

subresources. A set of resource names and rules for the construction of resource names. OPC uses these names when checking a user's authority to access individual OPC records.

subsystem. * A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system.

successor. An operation in an internal or external dependency that cannot begin until its predecessor completes processing.

sysout class. * An indicator used in data definition statements to signify that a data set is to be written on a system output unit. It applies only to print workstations.

T

temporary operator instructions. Operator instructions that have a specific time limit during which they are valid. They will be displayed to the workstation operator only during that time period.

TME 10 OPC. TME 10 Operations Planning and Control

tracker. The portion of TME 10 OPC or OPC/ESA that runs on every system in your complex. It acts as the communication link between the MVS system that it runs on and the controller.

tracking event log. A log of job tracking events and updates to the current schedule.

transport time. The time allotted for transporting materials from the workstation where the preceding operation took place, to the workstation where the current operation is to occur.

TSO. Time Sharing Option.

time zone support. A feature of OPC that allows applications to be planned and run with respect to the

local time of the processor that runs the application. Some networks may have processors in different time zones. The controlling processor will make allowance for differences in time during planning activities, for example the input arrival time of predecessor applications, to make sure that interacting activities are correctly coordinated.

turnover. A subfunction of job tracking that is activated when job tracking creates an updated version of the current schedule.

U

undecided (U). An OPC reporting status for an operation or an application indicating that the status is not known.

update authority. Access authority given to a user by RACF to use the ISPF/PDF edit functions of the OPC dialog. Access authority to modify a master file or data set with the current information.

V

validity period. The time interval defined by an origin date and an end date within which a run cycle or an application description is valid.

versions. Applications with the same ID but different validity dates.

VSAM. Virtual Sequential Access Method.

VTAM. Virtual Telecommunication Access Method.

W

waiting (W). An OPC reporting status (for an application) indicating that it is waiting for a predecessor operation to complete.

waiting list. A list of submitted jobs that are waiting to be processed.

work day end time. The time at which OPC will consider a work day to have ended when that work day immediately precedes a free day. For example, if you specify Saturday to be a free day, you could specify 08.00 hours. Saturday morning as the end of Friday's work day. OPC can then plan work to be done from 00.00 to 08.00 Saturday morning, as if that time was actually part of Friday.

workstation. A unit, place, or group that performs a specific data processing function. A logical place where work occurs in an operations department.

OPC requires that you define the following characteristics for each workstation: the type of work it does, the quantity of work it can handle at any particular time, and the times it is active. The activity that occurs at each workstation is called an operation.

workstation description database. An OPC database containing descriptions of the workstations in the operations department.

workstation resources. Limited resources defined for each workstation that an operation requires a certain amount of to process work.

workstation type. Each workstation can be one of three types: computer, print, or general.

work day. A day on which applications can normally be scheduled to start.

Index

Special characters

A

B

C

D

E

F

P

R

S

T

U

V

W

Readers' Comments — We'd Like to Hear from You

System Automation for z/OS
OPC Automation
Programmer's Reference
and Operator's Guide
Version 2 Release 3

Publication No. SC33-7046-05

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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.

Name _____ Address _____

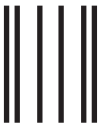
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC33-7046-05

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Entwicklung GmbH
Department 3248
Schönaicher Strasse 220
D-71032 Böblingen
Federal Republic of Germany

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5645-006

Printed in USA

SC33-7046-05