System Automation for OS/390

AOC/MVS OPC Automation Programmer's Reference and Installation Guide

Version 1 Release 4

SC23-3820-02

System Automation for OS/390

AOC/MVS OPC Automation Programmer's Reference and Installation Guide

Version 1 Release 4

SC23-3820-02

Note!

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

Third Edition (June 1999)

This edition applies to Version 1 Release 4 of the AOC/MVS OPC Automation Feature (5685-151), and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

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

If you prefer to send comments electronically, use one of the following methods:

FAX (Germany): 07031 + 16-3456
FAX (Other Countries): (+49)+7031-16-3456
IBM Mail Exchange: DEIBMBM9 at IBMMAIL
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:

CICS	MVS/XA	System/370
DB2	MVS/ESA	System/390
IBM	NetView	VTAM
IMS	RMF	3090

About This Book

This book provides programming information for the *IBM Automated Operations Control/MVS Operations Planning and Control (AOC/MVS) OPC Automation Feature, Version 1 Release 4*. Hereafter, this book refers to AOC/MVS OPC Automation as SA OS/390 OPC Automation. This is due to the fact that AOC/MVS 1.4 has been withdrawn and replaced by System Automation for OS/390 (SA OS/390).

OPC Automation is a feature of System Automation for OS/390 (SA OS/390) that brings together batch and online console automation into a common focal point. This automation feature automates, simplifies, and standardizes console operations and the management of component, application, and production related tasks.

Who Should Use This Book

Programmers

This manual was written primarily for programmers, the people who install and configure OPC Automation.

Programmers should also have a copy of the *AOC/MVS OPC Automation Operator and Scheduler Reference*.

OPC Automation Operators and OPC Schedulers

Although the *AOC/MVS OPC Automation Operator and Scheduler Reference* was written for operators and schedulers, they can refer to this programmer's reference and installation guide if they require additional information.

Evaluators

Although the *AOC/MVS OPC Automation General Information* was written especially for evaluators, they can read this programmer's reference and installation guide to obtain a better understanding of OPC Automation.

Prerequisite Knowledge

Familiarity with the functions and components of SA OS/390 as well as OPC would prove beneficial to understanding the concepts discussed in this manual. These IBM offerings work closely together to perform the automated console operations that are described.

What's in This Book?

This book contains the following parts, chapters, and appendixes:

Part 1, Introduction to OPC Automation
Provides an overview of the OPC Automation in the following chapters:

Chapter 1, OPC Automation Solution
Explains OPC Automation's approach to automation and the components of OPC Automation.

Chapter 2, Hardware and Software Requirements
Discusses OPC Automation's hardward and software environment.

Part 2, Concepts
Discusses several functions in the following chapters:

Chapter 3, Flow Overview
Discusses the components that provide the facilities which make up the interface between NetView and OPC and describes the interaction of the components.

Chapter 4, Automated Operator Tasks
Discusses automated operator tasks for the OPC Controller and the OPC Tracker.

Chapter 5, Initialization
Discusses the two phases of the automation platform.

Chapter 6, Request Handling in the OPC-PCS/Controller System
Discusses generating requests, handling time dependencies, and changes to the status of the operation.

Chapter 7, Request Handling in the OPC-EMS/Tracker System
Discusses the flow of OPC Automation for a request invoking a base function of the automation platform.

Chapter 8, Operations Control
Discusses the EVJESPIN module and obtaining information from OPC.

Chapter 9, Automated Recovery
Discusses the automated recovery function for requests that could not reach their destinations due to connectivity problems.

Part 3, Coding Formats and Data Areas
Describes how to specify OPC Automation functions, the control file entries, and data areas in the following chapters:

Chapter 10, Specifying OPC Automation Functions
Discusses defining the automation platform to OPC and defining OPC to the automation platform.

Chapter 11, Control File Entries Used by OPC Automation
Discusses the OPCA CODE, OPCACMD, OPCA DOMAINID, OPCAPARM, and OPCA PCS.

Chapter 12, Data Areas
Shows the subsystem status file OPC Automation entry (EVJSTS) and discusses the requestor id block (EHKVAR9) and the request buffer.

Chapter 13, Guidelines for User-Written Operations
Discusses two types of user-supplied extensions for implementation of functions beyond those provided by OPC Automation.

Part 4, Planning and Installation
Discusses installation and implementation procedures for the programmer. This section tells the programmer how to install and implement OPC Automation.

Chapter 14, Installation
Discusses the steps involved in the installation process.

Chapter 15, Merge NetView Related Members
Describes how to build OPC Automation parameter data sets and assemble the code that enables OPC Automation to operate in the NetView environment.

Chapter 16, OPC Automation Initial Customization
Describes the definitions that take place in NetView.

The Appendixes
Provide the following information:

Appendix A, Status Display Facility Enhancements
Shows the CLISTs used to implement the supplied enhancements.

Appendix B, OPC Automation Worksheets
Provides worksheets to define the workstations, operations, and the OPC environment.

Appendix C, Sample OPC Automation Control File
Lists the parameters used with OPC Automation.

A glossary of related terms and an index are also included.

What's New in This Book?

This edition contains major changes concerning the following::

- Introduction of SA OS/390 as a supported environment
- Modifications to support TME 10 OPC Version 2 (APAR number 35607)
- Modifications supporting Release 4 of AOC/MVS OPC Automation
- Technical changes reflecting service updates

A vertical bar (|) in the left margin indicates changes to the text and illustrations.

Related Publications

The SA OS/390 Library

The following table shows the information units in the SA OS/390 library:

Table 1. SA OS/390 Library

Title	Order Number
SA OS/390 General Information	GC28-1541
SA OS/390 Licensed Program Specifications	GC28-1540
SA OS/390 Planning and Installation	GC28-1549
SA OS/390 Customization	GC28-1566
SA OS/390 Operations	GC28-1550
SA OS/390 Messages and Codes	GC28-1569
SA OS/390 Technical Reference	GC28-1593
AOC/MVS CICS Automation General Information	GC23-3813
AOC/MVS CICS Automation Operator's Guide	SC23-3815
AOC/MVS CICS Automation Programmer's Reference and Installation Guide	SC23-3814
AOC/MVS IMS Automation General Information	GC23-3816
AOC/MVS IMS Automation Operator's Guide	SC23-3818
AOC/MVS IMS Automation Programmer's Reference and Installation Guide	SC23-3817
AOC/MVS OPC Automation General Information	GC23-3819
AOC/MVS OPC Automation Operator's Guide and Scheduler's Reference	SC23-3821
AOC/MVS OPC Automation Programmer's Reference and Installation Guide	SC23-3820

The System Automation for OS/390 books (except Licensed Program Specifications) are also available on CD-ROM as part of the following collection kits:

- IBM Online Library OS/390 Collection (SK2T-6700)
- IBM Online Library Networking Collection (SK2T-6012)

These softcopy collections include the IBM Library Reader, a program that enables you to view online documentation.

SA OS/390 Homepage

For the latest news on SA OS/390, visit the SA OS/390 homepage at http://www.s390.ibm.com/products/sa/

Related Product Information for the Base Program

The following table shows the books in the related product libraries that you may find useful for support of the SA OS/390 base program.

Table 2 (Page 1 of 2). Related Product Books

Title	Order Number
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
MVS/ESA SP V5 System Commands	GC28-1442
MVS/ESA SPL Application Development Macro Reference	GC28-1857
NetView for MVS V3R1 Administration and Security Reference	SC31-8045
NetView for MVS V3R1 Automation Implementation	SC31-8050
NetView for MVS V3R1 Automation Planning	SC31-8051
NetView for MVS V3R1 Command Reference	SC31-8047
NetView for MVS V3R1 Customization Guide	SC31-8052
NetView for MVS V3R1 Customization: Writing Command Lists	SC31-8055
NetView for MVS V3R1 Installation and Administration Guide	SC31-8043
NetView for MVS V3R1 RODM and GMFHS Programming Guide	SC31-8049
NetView for MVS V3R1 User's Guide	SC31-8056
NetView for MVS V3R1 Tuning Guide	SC31-8048
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 V1R2.0 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

Table 2 (Page 2 of 2). Related Product Books	
Title	**Order Number**
S/390 Managing Your Processors	GC38-0452
TSO/E REXX/MVS Users Guide	SC28-1882
TSO/E REXX/MVS Reference	SC28-1883
VSE/SP Unattended Node Support	SC33-6412
VSE/ESA 1.1.0 Unattended Node Support	SC33-6512
VTAM Version 3 Release 3 Network Implementation Guide	SC31-6404
VTAM Version 3 Release 4 Network Implementation Guide	SC31-6434

Related Product Information for Workstation Operations

The following are the books in the related product libraries that you may find useful for support of SA OS/390 workstation operations.

Table 3. Related Product Books	
Title	**Order Number**
APPC System Definitions in MVS/ESA and OS/2	GG66-3224
APPC Programming Considerations	GG24-3818
APPC Application Examples	GG24-3819
Distributed Console Access Facility User's Guide	GE13-0061
IBM Communications Manager/2 Version 1.1	G221-3630
IBM Communications Manager/2 Version 1.1 Information and Planning Guide	SC31-7007
IBM Communications Manager/2 Version 1.1 Workstation Installation Guide	SC31-6169
IBM Communications Manager/2 Version 1.1 Configuration Guide	SC31-6171
IBM Communications Manager/2 Version 1.1 User's Guide	SC31-6108
IBM Operating System/2 Version 2.1 Using the Operating System	S61G-0905
IBM Operating System/2 Warp	SR28-5668
NetView for MVS V3R1 Graphic Monitor Facility User's Guide	SC31-8095
Official Guide to Using OS/2 Warp	SR28-5659
Personal Communications Programmer's Guide	SC31-8660
Personal Communications Reference	SC31-8259
Personal Communications Tell Me About OS/2 Access Feature	SC31-8257
Personal Communications Up and Running	SC31-8258

Part 1. Introduction to OPC Automation

Chapter 1. OPC Automation Solution

TME 10 Operations Planning and Control (TME 10 OPC), Operations Planning and Control/Enterprise Systems Architecture (OPC/ESA), or Operations Planning and Control/Advanced (OPC/A) is a scheduling system that submits, tracks, and recovers the execution of batch work through a job entry system (JES) interface. NetView implements SA OS/390, and serves as the basis for automated console operations. OPC Automation is a program offering that capitalizes on the strengths of NetView, SA OS/390, and OPC by providing the ability to greatly expand job execution, scheduling, monitoring, and alert notification capabilities.

Note: For consistency and clarity, this document uses the term OPC to refer to OPC/A, OPC/ESA and TME 10 OPC. Similarly the term OPC Controller refers to OPC/A PCS, OPC/ESA Controller or TME 10 OPC Controller. AOC OPC Automation supports all products.

OPC Automation's Approach to Automation

With OPC Automation, NetView can use OPC calendar information to achieve a single-calendar definition that handles multiple systems and sites. A change in the OPC calendar can affect all the systems, ensuring consistency throughout the systems complex.

System Automation for OS/390 (SA OS/390) automates MVS console operations and provides the base for further automation when used with the NetView Solutions family of program offerings. SA OS/390 is a powerful application, designed to greatly reduce the time and effort required to meet automation objectives.

This approach to automation combines NetView and OPC with an SA OS/390 and OPC Automation. It provides a function that does not exist in any of these applications alone. Thus, the end result of combining these applications in an automated environment far exceeds the capabilities of these products when used individually. These applications complement the other so that the total of their capabilities is greater than the sum of their parts.

Basic Concept

Large, complex systems frequently require comprehensive schedules. There are regular workdays, other workdays (weekends), and complicated business cycles that take into consideration events such as:

- Holidays
- Financial quarter-end processing
- Sales promotions
- Maintenance
- Product development phases
- Testing

NetView does not easily lend itself to implementing these types of calendars. However, OPC has excellent calendar-management capabilities.

OPC Automation's basic concept consists of moving the management of functions that require calendar control from NetView and SA OS/390 to OPC, even if no batch component exist. This ensures a single point of control and eliminates

problems resulting from a loss of synchronization in calendars between OPC and NetView.

Examples of Placing Calendar Control

You can control startup and shutdown with SA OS/390 or with the OPC calendar functions. The following are examples of where to place this control:

- In the first situation, TSO is scheduled for availability at all times, regardless of dates or time of day. Here TSO is defined using the SA OS/390 control file, since calendar-specific control is not required.
- In the second case, TSO is required for specific hours on business days and different hours on weekends and holidays. Since SA OS/390 alone does not easily encompass calendar-specific events, you should define this in OPC and tie into SA OS/390 with OPC Automation. This approach offers a single point of control for all automated events.

SA OS/390

SA OS/390 provides automated console operation functions that are implemented through NetView CLISTs, command processors, message tables, and panels. These automation capabilities address the majority of subsystem and component automation requirements. They are an integral part of NetView and the MVS operating system. The automation of local resources in the operating system provides the primary focus of this approach.

When multiple MVS systems are interconnected and require consolidated operations at one focal-point system, you can configure SA OS/390 to communicate automation-related status and commands to and from that focal-point system. This enables you to view status from multiple systems on a single system which acts as a focal point.

SA OS/390 provides the capability to automatically start, monitor, and terminate MVS subsystems, components, and applications, such as JES2 or JES3, VTAM, TSO/E, IMS, CICS, DB2, RMF, NetView, and many others.

OPC defines a workstation as a unit or place that performs a specific data processing function. Examples of workstations include JCL preparation, data entry, CPUs, and printers. Activities that occur on workstations are referred to as operations. OPC Automation extends the idea of workstations to include NetView. Each NetView with AOC/MVS in your enterprise is represented by an OPC workstation. An OPC workstation may also represent all NetViews running SA OS/390 within the same sysplex where the OPC Controller (or OPC/A PCS) is running. These NetView workstations then schedule and perform operations on behalf of batch applications.

SA OS/390 Control File

A control file defines the scope of the automation that is performed. The control file supplied with SA OS/390 contains a basic sample set of subsystem and component definitions. This control file is designed to be easily expanded for specification of additional subsystems and components.

OPC Automation is an extension of SA OS/390. To implement OPC Automation, the following control file entries were added:

- OPCA CODE
- OPCACMD
- OPCA DOMAINID
- OPCAPARM
- OPCA PCS
- ENVIRON OPCAO

OPCA CODE

The OPCA CODE entry defines the parameters used for various requests. This entry is coded for each subsystem. For example:

```
RMF  OPCA,CODE=(START,,,'UP,3,RMFUTMER')
```

OPCACMD

The OPCACMD entry specifies the actual automation command that is issued for a request. This entry is coded for each subsystem. For example:

```
RMF  OPCACMD,CMD=(START,,'SETSTATE RMF,RESTART,START=YES')
```

OPCA DOMAINID

| The OPCA DOMAINID entry relates to an OPC/A automatic workstation to either a specific NetView domain ID or collectively to all NetView domains on the local sysplex. For example:

```
OPCA   DOMAINID,
           CODE=(NV06,,,AOF06),
           CODE=(NV00,,,AOF01),
           CODE=(NV08,,,SYSPLEX)
           CODE=(NV01,,,XBAOF)
```

OPCAPARM

The OPCAPARM entry defines the parameters used for various requests. This entry is coded for each subsystem. For example:

```
RMF  OPCAPARM,CODE=(START,,,',,')
```

OPCA PCS

| The OPCA PCS entry specifies either the NetView domain on which the OPC Controller resides or that the local sysplex is to be searched for the active controller when required. It must also specify the MVS subsystem name for the OPC controller.

| Example 1:

```
OPCA PCS,
      DOMAIN=AOF01,
      SUBSYS=OPCA
```

Example 2:

```
OPCA PCS,
      DOMAIN=SYSPLEX,
      SUBSYS=OPCC
```

ENVIRON OPCAO

The ENVIRON OPCAO entry specifies certain system-wide defaults, for:

- Retention of critical messages (MSGKEEP)
- Determining if operations can be reset after NetView has been unavailable (OPCRESET)
- Checking subsystem status before allowing requests to proceed (REQSTAT)

For example:

```
ENVIRON OPCAO,REQSTAT=YES,
              MSGKEEP=04:00,
              OPRESET=00:30
```

For a complete description of OPC Automation control file entries, see Chapter 11, “Control File Entries Used by OPC Automation” on page 65

For a complete description of the base SA OS/390 control file entries, to the *System Automation for OS/390 Customization*.

System Initialization with OPC Automation

JES starts OPC, which is usually operational at all times, as a task without SA OS/390. OPC Automation then transfers the responsibility of starting OPC from JES to SA OS/390, as described in the following scenario:

- The OPC Tracker has JES as a parent.
- During the IPL process, as soon as JES is running, AOC issues a start command for the Tracker subsystem.
- Once the Tracker has started, SA OS/390 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 Automation Table

SA OS/390 monitors messages received and compares them with those in the NetView automation table, formerly called the *message table*. When a message occurs that ordinarily requires manual operator intervention, such as responding to an outstanding WTOR, the control file directs a predefined response to the MVS console without operator intervention.

NetView Interface to OPC Automation

The program-to-program interface (PPI), a high-performance interface, provides synchronization and bidirectional 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, outstanding tape mounts, and OPC errors
- Implementation of a two-way interface between OPC and NetView with SA OS/390:
  - OPC defines and controls interactive applications. Support is provided to start and stop subsystems that are defined to the SA OS/390 application.
  - Database tasks can run in both interactive and batch systems with full synchronization between the activities.
  - SA OS/390 can access OPC calendars and other information.
  - NetView operators can access and update OPC-defined applications without the need to log on to OPC.
- Two user extensions:
  - OPCACOMP allows the startup and shutdown of subsystems not controlled by SA OS/390.
  - UX*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.

For example, if you wanted to display OPC detail information from a NetView console, enter the following command from any NetView command line:

OPCACMD

After you press ENTER, OPC Automation displays the AOC OPC: Display or Modify OPC Data panel (EVJKAC01), as shown in Figure 1.

```
EVJKAC01         AOC OPC: Display or Modify OPC Data
                                                          Date: 05/17/95
                                                          Time: 14:36:36

Specify search criteria and press ENTER

  Subsystem    :  OPCC                From ACF-file

  Application  :  rmf*_____________   Can be generic
  Opno         :  ____                Numeric
  Jobname      :  _________           Can be generic
  Wsname       :  ____                Can be generic
  Group        :  _________           Can be generic
  Owner        :  _________           Can be generic
  Priority     :  _                   1-9 (1=low, 9=high)
  Errcode      :  ____                Can be generic
  Status       :  _                   A/W/S/R/C/I/E/U



Action===>
F1= Help     F2= End      F3= Return                              F6=Roll
```

Figure 1. AOC OPC: Display or Modify OPC Data Panel

Note: This presentation shows how OPC Automation panels interface with OPC. For further details, refer to *AOC/MVS OPC Automation Operator and Scheduler Reference*.

To list all of the applications defined to OPC on this system, type an asterisk (*****) in the application field, as shown in Figure 2.

```
Application  :  *_____________    Can be generic
```

Figure 2. Generic Search Function

Although this illustration shows the generic search function, refer to *AOC/MVS OPC Automation Operator and Scheduler Reference* for more details in selecting your list.

After you press ENTER, OPC Automation displays the Display or Modify OPC Data panel (EVJKAC03), as shown in Figure 3.

```
EVJKAC03            Display or Modify OPC Data              Page:   1  of   1
                                                            Date: 05/17/95
CMD: C change  B browse                                     Time: 14:36:42
     H hold  R release  N no-op  U unno-op
CMD  Application      Jobname   Ws  Opno St   Inp. Arr   Description  H    N
---  ---------------- -------- ---- ---- -    ---------- -----------  -    -
 _   RMFDLY           RMF      NV11 0001 C    9505262230 STOP         N    N
 b   RMFDLY           RMFMAINT CPU1 0002 E    9505262230              N    N
 _   RMFDLY           RMF      NV11 0003 W    9505262230 START        N    N
 _   RMFDLY           RMF      NV11 0001 A    9505302200 STOP         N    N
 _   RMFDLY           RMFMAINT CPU1 0002 W    9505302200              N    N
 _   RMFDLY           RMF      NV11 0003 W    9505302200 START        N    N
 _   RMFDLY           RMF      NV11 0001 A    9506062200 STOP         N    N
 _   RMFDLY           RMFMAINT CPU1 0002 W    9506062200              N    N
 _   RMFDLY           RMF      NV11 0003 W    9506062200 START        N    N




Action===>
F1= Help     F2= End      F3= Return                 F5= Refresh   F6= Roll
```

Figure 3. Display or Modify OPC Data Panel

To display detail information about application MAINT, type **B** in the CMD field. After you press ENTER, OPC Automation displays the OPC Occurrence Data panel (EVJKAC02), as shown in Figure 4.

```
EVJKAC02              OPC Occurrence Data
                                                        Date: 05/17/95
                                                        Time: 14:38:30

Subsystem    : OPCC               Deadline  : 95/05/26 23:30
Application  : RMFDLY             OPIA date :   /  /     OPIA time :   :
Opno         : 0002               Edur      : 0005       PS reqd   : 0001
Inp arrival  : 95/05/26 22:30     Job Class : P          R1 reqd   : 0000
Jobname      : RMFMAINT           Auto Sub  : Y          R2 reqd   : 0000
Wsname       : CPU1               AJR       : Y          A E C     : Y    Y/N
                                  Clate     : N          Timedep   : N
                                  Form no   : ____       Hi RC     : 0000
Priority     : 5                  Reroute   : N          Restart   : Y
Error Code   : SB37               DeadWTO   : N          Cat mgt   : N
Status       : E                Manual hold : N          NOP       : N
                                  Description : ________________________
                                  User data   : ________________


Action===>
F1= Help     F2= End      F3= Return                          F6=Roll
```

Figure 4. OPC Occurrence Data Panel

Status Display Facility

SA OS/390 uses the Status Display Facility to provide a central focus of information. The Status Display Facility describes the status of all automated systems and applications of a NetView complex, such as:

- A consolidated, hierarchical view of an entire operating environment with detailed information where required.
- Dynamically updated status panels that use color, representing system and application status to enhance usability and to expedite comprehension of priority information.
- A central repository for the status of automated resources and components.
- A facility for viewing the status of multiple target systems by a focal-point system operator.
- Simplified techniques for presenting and maintaining resource status for multiple systems, subsystems, and applications. For example:
  - Multiple programs can asynchronously update the Status Display Facility with subsystem status without concern for sequence or priorities posted by other programs.
  - The Status Display Facility resolves the priority of conflicting statuses and displays the most severe status.
  - When a problem is resolved, the program resolving the problem can update or clear the condition previously set without regard for any other status posted by other programs. The status of the application is automatically updated to the current status or most serious problem as appropriate.

OPC Automation provides the following additional fields and detail panels for the Status Display Facility:

CRITMSG Critical messages

TAPES Outstanding tape mounts

TSOUSERS TSO users logged on

BATCH Batch jobs being executed

OPCERR OPC-detected errors

To use the Status Display Facility, refer to *AOC/MVS OPC Automation Operator and Scheduler Reference*.

Request and Confirmation Transaction Flow

Figure 5 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 5, the jobname is specified as RMF and the operation text is START.

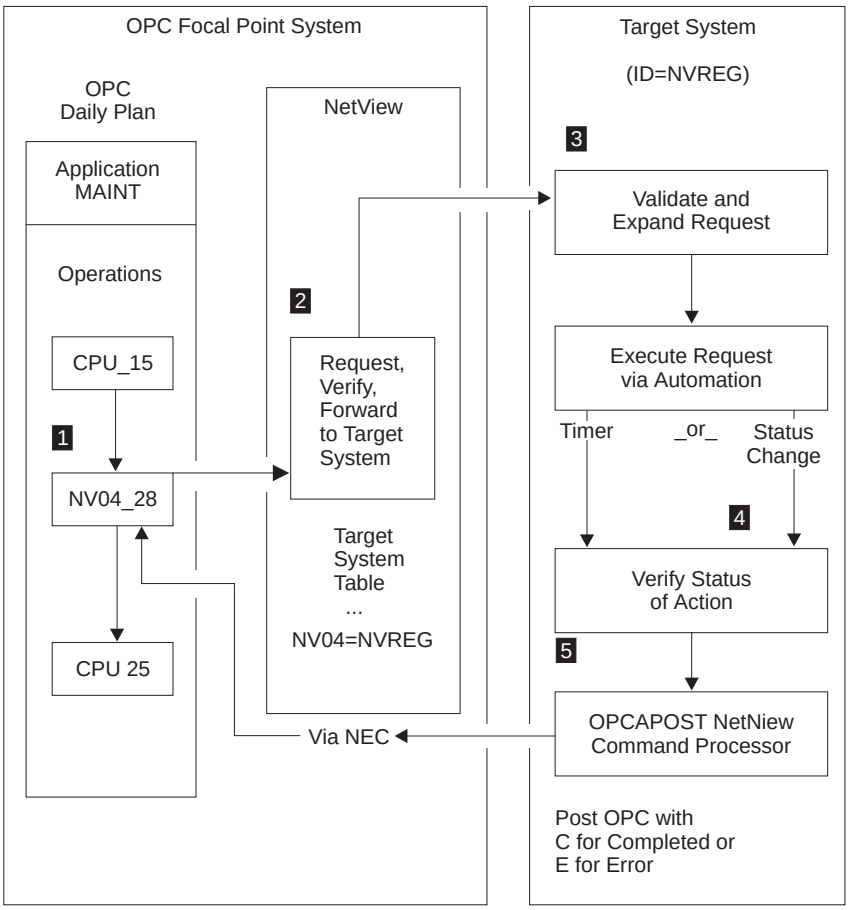


Figure 5. NetView–OPC Interface Flow. Syntax and definition errors, target system availability, recovery, and resynchronization via OPC API and NetView PPI are not shown in this example.

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 SA OS/390 OPC Automation parameter definitions.

Figure 5 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 request to start RMF which is then forwarded to NetView Domain NVREG.

OPC Automation uses the NetView PPI to transfer the request from OPC to NetView. This transfer of the NetView 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 OS/390. 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 OS/390 control file.

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 automation control file. In Figure 5 on page 12, the request function translates the request buffer to the SA OS/390 function, which then starts RMF, but not an MVS START command. The automation control file entry to start RMF is:

```
RMF OPCACMD,CMD=(START,,'SETSTATE,RESTART,START=YES')
```

Functions other than START and STOP of systems based on SA OS/390 may require user programming.

The command is dynamically generated using definitions in the SA OS/390 control file. 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 OS/390 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/DRKUSINT 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, the OPCACOMP function is called. 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 5 on page 12.

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.

OPC Automation Log Entries

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 6 shows a request buffer log entry. In this example, OPC is processing the MAINT application. An OPC controller running on domain NVDOM has made a request to OPC Automation to perform START for the RMF subsystem. When this action is completed, OPC Automation changes the status of the NV00_10 operation in the MAINT application to C (completed).

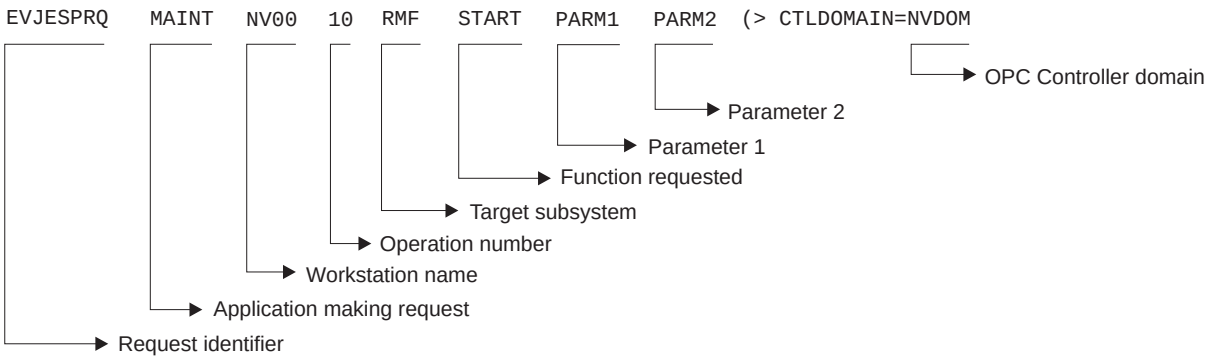


Figure 6. NetView Log Entry of an OPC Generated Request

Chapter 2. Hardware and Software Requirements

OPC Automation is supported on any hardware and software environment supported by SA OS/390.

OPC Automation supports the following program products:

- TME 10 Operations Planning and Control (TME 10 OPC) Version 2 Release 1 or Version 2 Release 2, program number 5687-OPC.
- Operations Planning and Control/Enterprise Systems Architecture (OPC/ESA) Version 1 Release 2 or higher, program number 5696-007
- Operations Planning and Control/Advanced (OPC/A) Version 1 Release 2, program numbers 5665-371, 5665-372, and 5665-373

For OPC Automation sysplex related functions (e.g., control of ARM enabled applications, support for a standby OPC controller) the following additional minimum requirements are imposed:

- TME 10 OPC Version 2 Release 1 or higher
- OPC Automation APAR number OW35607
- SA OS/390 Version 1 Release 3
- The OPC Controller must be running in a sysplex environment

When running OPC Automation in conjunction with OPC/A, function is limited to the automation functions described in this book, and to the fields in the OPCACMD interface which are available in OPC/A. If the operator attempts to alter fields which OPC/A does not support, error messages will result. Also, the SRSTAT command will not function in an OPC/A environment, as OPC/A does not support this environment.

Part 2. Concepts

Chapter 3. Flow Overview

OPC Automation is an interface between NetView, OPC, and SA OS/390. These components provide the facilities which make up the interface. This chapter 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 32 describes this. OPC Automation initialization includes the automated recovery sequences described in Chapter 9, “Automated Recovery” on page 41. Also refer to *AOC/MVS OPC Automation Operator and Scheduler Reference*.

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 12 summarizes this request.

Figure 7 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.

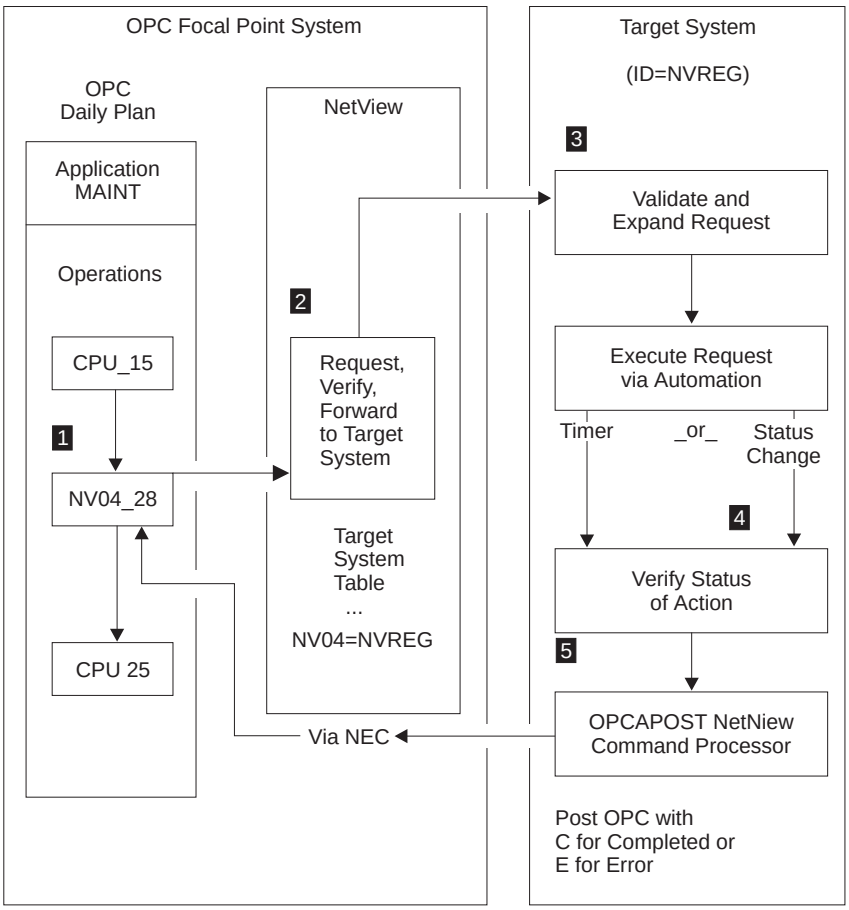


Figure 7. NetView-OPC Interface Flow. Syntax and definition errors, target system availability, recovery, and resynchronization via OPC API and NetView program-to-program interface are not shown in this example.

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*nn* specifies a NetView automatic general workstation with a NetView domain index of *nn*, which is resolved in the PCS-NetView into the target NetView domain ID through the definitions in the automation control file. OPC can define the NV*nn* workstation with all regular specifications, such as parallel servers and special resources.

In the MAINT example in Figure 7, 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 DRKUX007 exit in OPC/A-PCS. In OPC/ESA systems, the EQQUX007 exit in the OPC/ESA controller carries out this function.

EQQUX007 (DRKUX007) Exit

Each change of status on any workstation calls the EQQUX007 (DRKUX007 in OPC/ESA) exit, which checks for a NV*nn* workstation proceeding to the R (ready) status. The EQQUX007/DRKUX007 exit ignores all other conditions (operations going to A or * are counted as ready). Figure 8 shows the flow of this process.

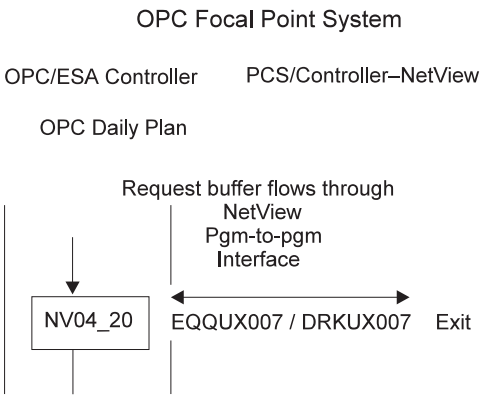


Figure 8. EQQUX007/DRKUX007 Exit

When an NV*nn* workstation moves to the R status, the workstation generates a request buffer. Fields pointed to by registers in the EQQUX007/DRKUX007 exit provide all of the data for the request buffer. For the layout of the fields in the request buffer, see Table 6 on page 79 and Table 7 on page 80.

OPC Automation supplied EQQUX007/DRKUX007 exit logic verifies that all fields exist except the optional parameter fields. If this exit logic determines if 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 DRKUX007 exit contains no capability to directly change the status of an OPC operation when an error code is posted to OPC, the EQQUX007/DRKUX007 exit uses the EQQUSINT/DRKUSINT 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 PCS/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/DRKUSINT 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/DRKUX007 logic. OPC Automation then restores registers and returns control to OPC.

If the OPC Automation EQQUX007/DRKUX007 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 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 OS/390 application. The PPI dispatcher task (EVJTOPPI), a NetView subtask, receives the SA OS/390 action requests from the buffers from the EQQUX007/DRKUX007 exit. Figure 9 shows this flow.

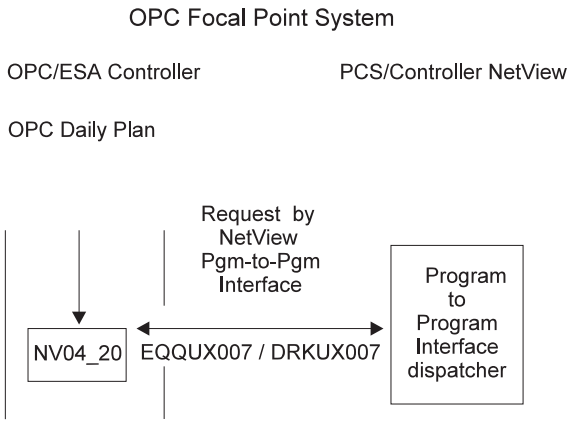


Figure 9. PPI Dispatcher

On the basis of the sending task identifier, the PPI dispatcher determines the function in SA OS/390 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/PCS-NetView. This module receives the action request buffer from the PPI dispatcher task. Figure 10 shows this process.

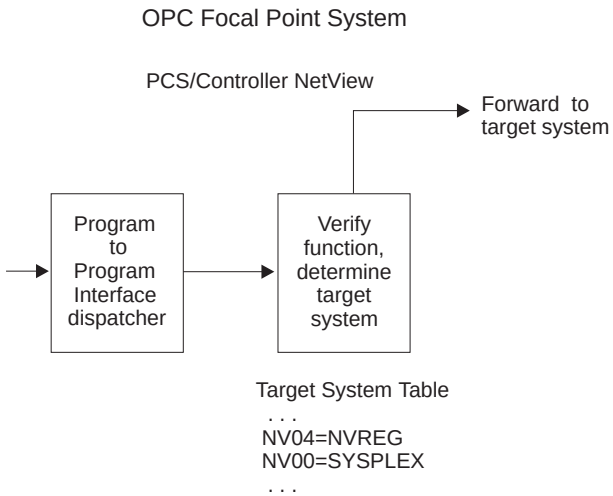


Figure 10. Verify Module

The verify module uses the NV*nn* index to obtain the destination NetView domain ID from the automation control file. If the relevant NV*nn* index specifies SYSPLEX then all AOCs in the local sysplex are queried for the status of the job named in 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*nn* 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 automation control file needs to define this workstation using the OPCA DOMAINID entry. You must manually reset operations that are posted-in-error since OPC Automation carries out no automated recovery for definition errors.

If NV*nn*=SYSPLEX has been specified and OPC Automation does not find the job defined to any online AOC 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 AOC 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/DRKUX007 exit.

Request Module (EVJESPRQ)

The arrival of a request from the verify module drives the request module in Tracker/EMS-NetView. OPC Automation installs the request module on each system running an OPC/ESA Tracker. Figure 11 shows the flow of this process.

The main functions of the request module are to:

- Translate the OPC-generated request to one which is understood by SA OS/390
- Schedule a user-defined function

For more information on a user-defined alternative to the request module, see “Relating to the SA OS/390 Defined Subsystem” on page 81.

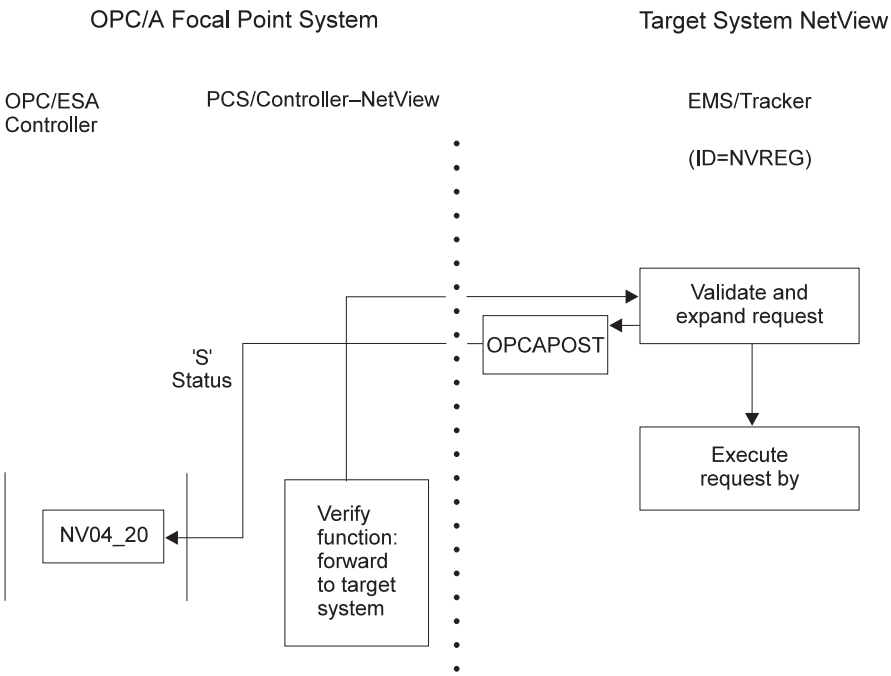


Figure 11. Request Module

If required, the request module uses definitions in the control file to translate the subsystem name to the job name and to create the SA OS/390 command that initiates the function requested. The control file contains request and parameter fields and uses these entries to obtain the command text and the parameter syntax for the actual request. 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. An error is posted to OPC by the OPCAPOST module, the error is logged, and a WTO is issued. 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 11, the request module translates the requested action in the buffer to the SA OS/390 command required to start RMF. The SA OS/390 command then starts RMF.

Except for starting, stopping, or recycling SA OS/390-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 “Relating to the SA OS/390 Defined Subsystem” on page 81.

For subsystem operations, the OPCAPOST command processor posts to OPC if the values are found in the automation control file. OPC then changes the status from R (ready) to S (started). OPC Automation then issues the SA OS/390 command, and checks the return code of the operation. If the command is properly executed, OPC Automation issues a timer request, on the basis of the delay specified in the control file, and the request module terminates.

A change of subsystem status calls the status-change exit module. If the status change does not occur, the time-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 automation control file and dynamic substitution of command fragments on the basis of the parameters.

Status Change Module (EVJESPSC)

SA OS/390 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 12 shows the flow of this process.

If an outstanding action for the changed subsystem exists, and the new results in 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.

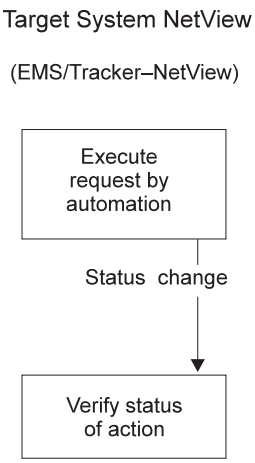


Figure 12. Status Change Module

Timer Module (EVJESPTE)

Under normal conditions, a request passed to SA OS/390 results in the desired status change before the timer expires, and OPC purges the timer. When this sequence does not occur, and the timer remains at the end of the timer interval, SA OS/390 drives the timer module.

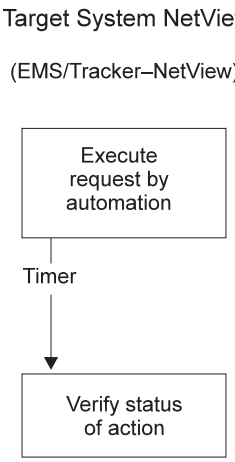


Figure 13. Timer Module

For normal SA OS/390 functions, the status file provides the current status and compares the results 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 13 shows the flow of this process.

OPCAPOST Command Processor

The OPCAPOST command processor calls EQQUSINT/DRKUSINT, which passes the completion code to the OPC Tracker in this system. The OPC Tracker (or EMS) forwards the completion code to the system running the Controller (or PCS) through shared DASD or where OPC/A-PCS is running at a different location through OPC/A-NEC. Figure 14 shows the flow of this process.

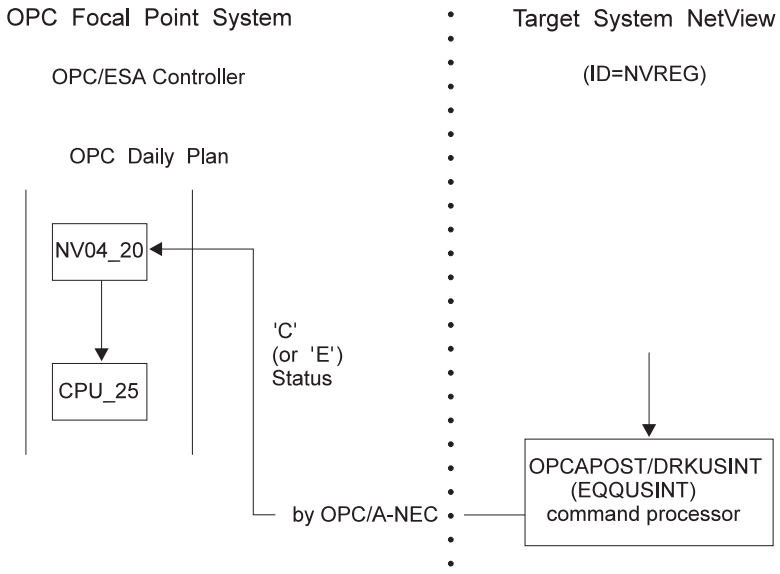


Figure 14. OPCAPOST Command Processor

Other functions use the OPCAPOST command. See “OPCAPOST” on page 63 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 14. 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.

Chapter 4. Automated Operator Tasks

Defining OPC to SA OS/390

OPC is an SA OS/390 controlled subsystem. Normal automation control file definitions can describe OPC. In addition, SA OS/390 defines an automated operator task 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 OS/390 application.

OPC Automation requires actions in a specific order. Changes in this order can result in unpredictable and undesirable results. To ensure that 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 though 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.

```
AUTOOPS     OPCAOPR1,ID=AUTOPCP,MSG=(CSY*,DRK*,CSZ*,EQQ*,EVJ*)
AUTOOPS     OPCAOPR2,ID=AUTOPCE,MSG=(CSY*,DRK*,CSZ*,EQQ*,EVJ*)
```

For the automated operator task OPCAOPR1, 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 OPCAOPR1 and OPCAOPR2.

Chapter 5. Initialization

SA OS/390 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 OS/390 application, OPC starts as a JES task. With OPC Automation, the responsibility of starting OPC transfers from JES to the SA OS/390 application. Figure 15 shows an example of the startup of OPC/ESA during an IPL process. OPC/A startup is similar.

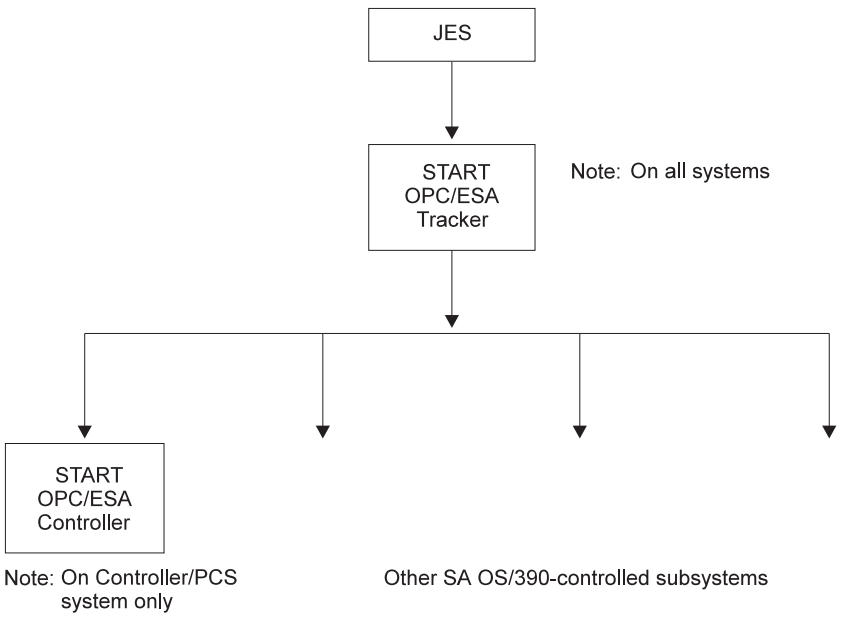


Figure 15. OPC/ESA Startup During IPL Process

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 OS/390 controlled flow.
- Once OPC/ESA Tracker starts, the SA OS/390 application issues a start command for the OPC/ESA controller on the control host only.
- The SA OS/390 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 OS/390 application completed initializing its defined tasks, the startup of the OPC-controlled subsystem phase 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 automation control file 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 for OPC. This allows coding entries into the control file before defining the subsystems in the rest of SA OS/390 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 *AOC/MVS OPC Automation Operator and Scheduler Reference*.

Chapter 6. Request Handling in the OPC-PCS/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*nn* workstation, where NV*nn* represents a NetView domain. When the daily planned execution of OPC makes this NV*nn* workstation ready, OPC Automation starts through the EQQUX007 OPC/ESA (DRKUX007 in OPC/A) user exit. See Figure 16 for an illustration of this flow.

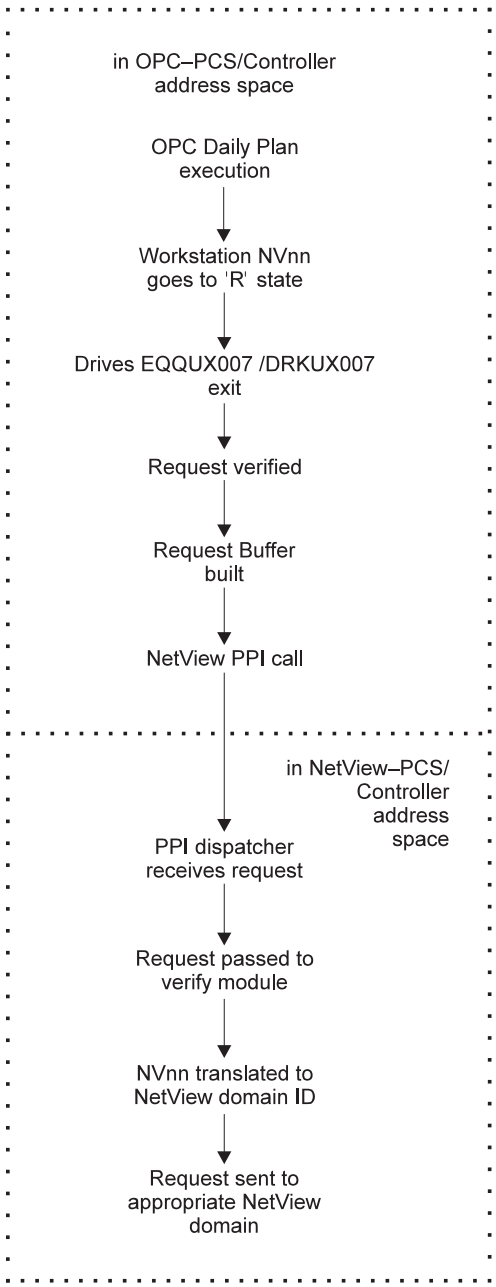


Figure 16. Request Handling in the OPC PCS/Controller Processor

Each OPC change of status drives the OPC/A-PCS DRKUX007 (OPC/ESA EQQUX007) exit. OPC Automation logic in the exit intercepts a change of status to R (ready) for an NV*nn* 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-PCS/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 OS/390 is restarted, provided these errors occurred less than a user-specified time interval. (See “ENVIRON OPCAO” on page 71.)

The verify module translates the NV*nn* to a real NetView domain ID, and sends the request buffer to the appropriate domain through forwarding functions of SA OS/390. If the request is destined for the same system as the one that OPC-PCS/Controller is on, OPC Automation transfers the request to the request module running under the NetView-EMS/Tracker automated operator task.

The request is one of three types:

A base request for an SA OS/390 defined subsystem
: The System Automation for OS/390 directly supports this request type. Chapter 7, “Request Handling in the OPC-EMS/Tracker System” on page 37 discusses this topic.

Nonbase SA OS/390 request
: Chapter 13, “Guidelines for User-Written Operations” on page 81 describes a request for an SA OS/390 defined subsystem that is not a base SA OS/390 request. Now, once the action completes, the user is responsible for posting to OPC Automation using OPCACOMP.

A request for a user-defined module
: This may involve an SA OS/390 defined subsystem. For this type of function, OPC Automation provides only the transport mechanism for the request buffer to the user-defined module. The user is responsible for posting to OPC. Chapter 13, “Guidelines for User-Written Operations” on page 81 further describes this task.

Handling Time Dependencies

If you require a time dependency, do not place the time consideration on the NV*nn* defined operation because the status change drives the OPC user exit EQQUX007/DRKUX007, 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*nn* 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.

Figure 17 is an example of using time as a dependency. In this illustration, the previous example of the MAINT application is redefined with a timer dummy workstation (TIMR) as the first operation of the application.

```
--------------------------------- 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 17. Using Time as a Dependency

If the application needs scheduling-on-demand or restarted, you may schedule the NV*nn* operation with this type of application. If necessary, this allows the manually initiated procedure to run independent of the time consideration.

Changes to the Status of the Operation

The operation with the NV*nn* workstation proceeds through several status changes, as the request defined in the operator text is processed. The initial trigger consist of one of three status changes. The progression of the operation to the A, R, or * status triggers the OPC Automation function in the EQQUX007/DRKUX007 exit. The exit logic examines the request. If the request is valid, it transfers the request to the PCS-NetView and through the NetView PPI. OPC Automation sends the request to the appropriate target NetView, as determined by the NetView domain ID represented by the NV*nn* workstation. If any definition problems exist, OPC Automation updates the status to E with an error code of U*xxx* and takes no further action. The user is then responsible for correcting the error and restarting the application at the appropriate operation.

If OPC Automation determines a connectivity problem with a remote NetView, it marks the operation with a status of E and an error code of S*xxx*. OPC Automation now automatically restarts the operation once it resolves the connectivity issue. However, manually changing the operation status causes the suppression of the automatic restart.

If OPC Automation determines that neither definition nor connectivity problems exist, it analyzes the request in the target NetView. After OPC Automation resolves and verifies the request before submission, it updates the status to S (started). Once the request is submitted and the requested action successfully completes, OPC Automation updates the status to C (completed). If the requested result does not occur within the time period specified, OPC Automation ends the operation with an E status and U*xxx* code, which indicates that the resolution requires user intervention.

Extending the Daily Plan

OPC Automation does not call EQQUX007/DRKUX007 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*nn* 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.

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.

Chapter 7. Request Handling in the OPC-EMS/Tracker System

Figure 18 describes the flow of OPC Automation for a request invoking a base function of the SA OS/390 defined subsystem.

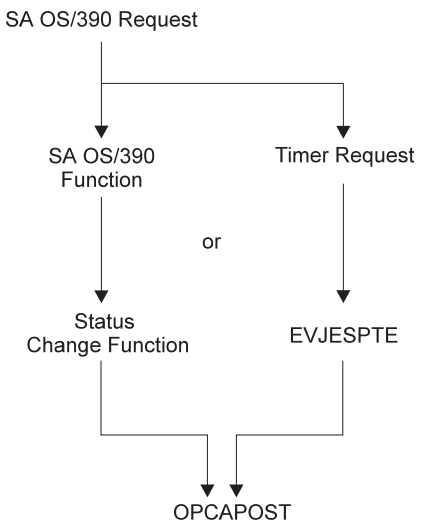


Figure 18. Request Flow for a Base SA OS/390 Function

The request module (EVJESPRQ), a part of the EMS/Tracker portion of OPC Automation, runs under the NetView-EMS/Tracker automated operator task. This module issues the SA OS/390 base command sent in the request buffer. The automation control file entry for each system defines the actual SA OS/390 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

A request sets a timer for every command issued. This ensures that no lockout condition occurs. The timer value specified in the automation control file 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 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 TIMEREND 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 TIMEREND 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 TIMEREND 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 TIMEREND 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. Each flag is related to the status change module or the TIMEREND module. When the modules are executed, OPC Automation examines the flags. If neither flag is on, OPC Automation 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-EMS/Tracker.

Chapter 8. Operations Control

This chapter discusses the EVJESPIN module and obtaining information from OPC. *AOC/MVS OPC Automation Operator and Scheduler Reference* describes operator commands and their actions.

EVJESPIN Module

This EVJESPIN module, which is also used as a command, provides two separate capabilities:

- Creates an OPC Automation status file record
- Resynchronizes or unlocks 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/A.

SYNC A second option is used when the OPC Automation status file record contains the proper status of UP or CTLDOWN. OPC indicates this status for a specific subsystem, when the actual subsystem is in a different status. This option issues the SA OS/390 command to change the status of the subsystem to match that in OPC Automation status file record. The SYNC function does not post to OPC on completion of the SA OS/390 function. OPC's requests for the subsystem synchronizes SA OS/390.

To add a new automation control file entry since last initializing NetView, you can create a single OPC Automation status file record. This option (CREATE) 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. The EVJSTS REQ=DEL command accomplishes this task.

Obtaining Information from OPC

OPC/A Release 2 provides an application program interface (API) that is the standard in OPC/ESA. This 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 to manually post events that have occurred.

You can also use this interface when manual intervention is required, for example, as 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-PCS/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-PCS/Controller task. Sometimes, when a user-written module utilizes this function, another NetView requires the information. OPC Automation maintains a control file entry to allow OPC Automation to determine the domain ID of the OPC/A-PCS NetView to which the request is sent. This entry has the domain ID of the NetView on the processor running OPC-PCS/Controller. In this manner, all the SA OS/390 applications can easily determine where to direct OPC-PCS/Controller requests.

A user-written task or CLIST can also access the OPC-PCS/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 56,
- “OPCAMOD” on page 59,
- “OPCACAL” on page 52,
- “OPCACOMP” on page 55,
- “OPCAPOST” on page 63, and
- “OPCSRST” on page 64.

Chapter 9. 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*nn*) is defined in the control file as SYSPLEX 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*nn* 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 reinvokes the EQQUX007/DRKUX007 exit. OPC Automation attempts no other recovery.

Part 3. Coding Formats and Data Areas

This part describes OPC Automation common routines which request information or perform tasks associated with OPC/ESA automation. You can use these common routines in automation procedures you create. Examples, sample routines, and data area information is given to show how this might be done.

Chapter 10. Specifying OPC Automation Functions

This chapter discusses the following:

- Defining SA OS/390 to OPC
- Defining OPC to SA OS/390
- Transferring information from OPC to SA OS/390
- Posting an operation in OPC from SA OS/390

Defining SA OS/390 to OPC

OPC allows the definition of automatic reporting general workstations. This is a class of workstation that OPC can manage, but is outside OPC's direct control. With the implementation of this workstation, NetView becomes a server to OPC. The workstation definition used by OPC is NV*xx*. The characters *xx* are used by OPC Automation to translate the workstation name to a NetView domain ID.

Using these definitions, OPC knows each NetView domain and can have work scheduled on it. The OPC definition of NetView workstations can use normal OPC specifications such as open hours, parallel servers, and special resources. OPC uses this information in its planning and management of the NetView workstations.

Defining OPC to SA OS/390

Since OPC is an automation-controlled subsystem, you use normal control file definitions to describe OPC to the SA OS/390 application. SA OS/390 informs OPC of status changes through the OPCAPOST command processor, which OPC calls from a NetView CLIST. This command processor includes the EQQUSINT/DRKUSINT module supplied with the OPC program product.

The EQQUSINT/DRKUSINT module transfers a status change of an operation in an OPC-defined application to the Tracker in the same system. The Tracker transfers this status change to the Controller through shared DASD or NJE links, depending on the location of the Controller.

Transferring Information from OPC to SA OS/390

To cause the request to be executed and for OPC to receive a status change once the request is completed, enough information must transfer from OPC to OPC Automation to allow routing of the request to the proper destination NetView domain. The minimum information that needed to accomplish this is:

- OPC-controlled application making the request
- Operation within the application that is to be notified
- NetView domain to which this request is to be sent
- Subsystem for which this function is to executed
- Function requested
- Request parameters

Use standard TSO dialogue panels to define this request to OPC. The requested function and any request parameters are not standard OPC fields. Enter these fields into the operation text field, delimited by blanks. Figure 19 shows an example of the OPC panel with the operator text field used as a request field.

```
--------------------------------- 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_____  START___________________
```

Figure 19. OPC/A Operation Panel

The EQQUX007/DRKUX007 exit logic copies the information from the OPC panels to a request buffer. The PPI then transfers this buffer to NetView. Chapter 11, “Control File Entries Used by OPC Automation” on page 65 and Chapter 12, “Data Areas” on page 77 provide additional information about this process.

Figure 20 shows an example of the request buffer. Table 6 on page 79 and Table 7 on page 80 shows a detailed request buffer layout. In this example, OPC is processing the MAINT application. OPC makes a request to SA OS/390 to perform the function START for the RMF subsystem. When this action is completed, the SA OS/390 application changes the status of the NV04_20 operation in the MAINT application to C (completed).

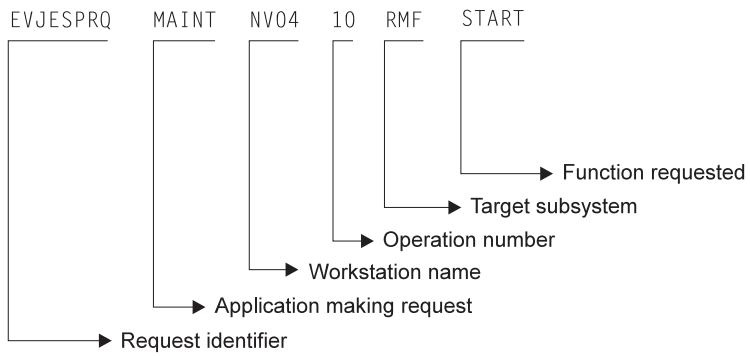


Figure 20. OPC-Generated Request Buffer

OPC Automation logic permits the inclusion of two optional parameters in the request buffer. The target system uses these parameters to build the required command, using dynamic substitution of command fragments stored in the control file.

Alternatively, these parameters pass control information to a user-supplied function. The contents of the request buffer passes to OPC Automation in the variable &EHKVAR1. User-supplied routines may use the contents of this variable to extract whatever information is needed.

Figure 21 shows an OPC-generated request buffer with optional parameters. Figure 27 on page 87 shows sample code for an operation using these optional parameters.

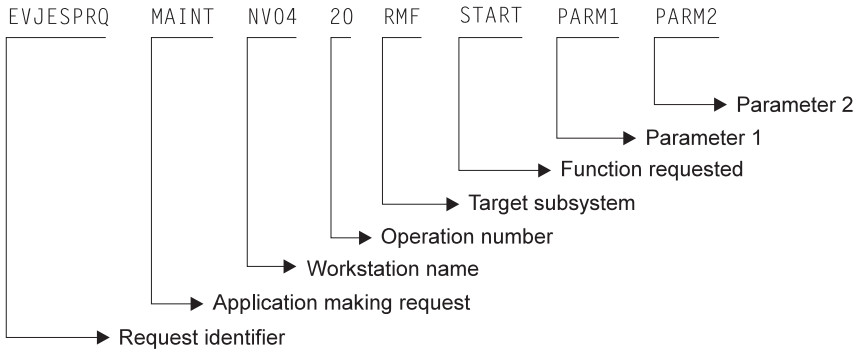


Figure 21. OPC-Generated Request Buffer with Optional Parameters

Posting an Operation in OPC from SA OS/390

SA OS/390 must inform OPC of status changes. The OPCAPOST command processor accomplishes this task. Although OPC Automation normally calls OPCAPOST from a NetView CLIST, you can also issue OPCAPOST as an operator command. However, an operator should use OPCACMD instead, since this command provides a full-screen interface to OPC.

The OPCAPOST command processor includes the EQQUSINT/DRKUSINT module supplied with the OPC program product. This module transfers a status change of an operation in an OPC-defined application to the Tracker resident in the same system as OPC Automation making the request. The Tracker transfers this status change to the Controller through shared DASD or OPC/A-NEC, depending on the location of the OPC-PCS/Controller.

The OPCAPOST command processor indicates to OPC to change a current operation in the R (ready) status to an S (started) status when the request is accepted. When the request is completed, the OPC status is changed to a C (completed) or E (error) status. OPCAPOST uses the DRKUSINT module, shipped as part of OPC, to post certain status changes to OPC-PCS/Controller from any system containing a copy of OPC-EMS/Tracker.

The OPCAPOST command processor requires input fields to identify the specific application, workstation, operation, and completion code, which are posted in the OPC-PCS/Controller current plan. Although OPCAPOST is used by OPC Automation modules, you can issue it as an operator command. For example, OPC Automation issues OPCAPOST to provide resynchronization between SA OS/390 and OPC after a failure. Normally, you should use the OPCACOMP rather than the OPCAPOST module when completing your own processing, so that the OPC Automation status file is also updated with the completion information.

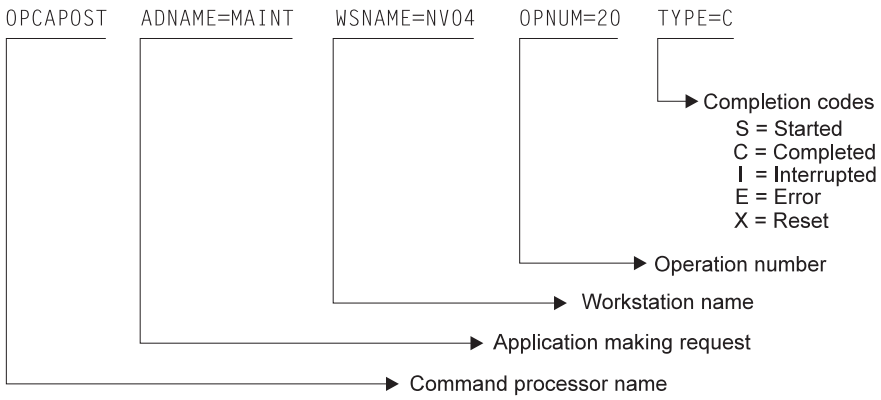


Figure 22. OPCAPOST Command Processor Request

For an E (ended-in-error) status, OPC Automation adds an additional field with an error code. The example shown in Figure 22 would change to the example in Figure 23.

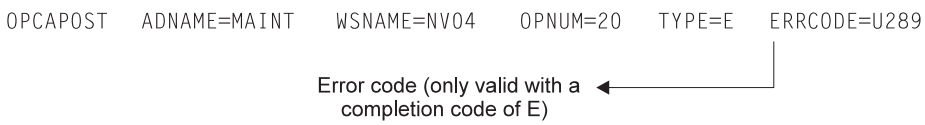


Figure 23. OPCAPOST Command with Optional Error Code

See “OPCAPOST” on page 63 for a detailed description of the OPCAPOST command processor, all acceptable input parameters, and return codes.

The OPCAPOST command processor must run as an authorized application. The command processor sets up the user and input parameter values for the EQQUSINT/DRKUSINT module and calls EQQUSINT/DRKUSINT to issue an OPC event record.

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 then you may receive an EQQZ038E message indicating that the OPC message log is not available, because the log dataset (DD EQQMLOG refer chapter 15) is required by EQQYCOM exclusively for each invocation.

EVJESHUT

Purpose

Use the EVJESHUT routine in a RECYCLE operation defined in the control file. A RECYCLE operation is one in which a subsystem which is currently UP is brought down and immediately restarted. EVJESHUT will issue the appropriate SHUTSYS 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 5” on page 68 to see how to include EVJESHUT in an OPCACMD control file entry.

Format

```
EVJESHUT  subsys scope
```

Parameters

subsys

The name of a valid subsystem defined to the automation

scope

The scope of the SHUTSYS request. The automation platform recognizes only three valid values for SCOPE: ALL, CHILDREN, and ONLY. These values are discussed in the SA OS/390 base documentation on the SHUTSYS command.

Usage Notes

If the REQSTAT flag on the ENVIRON OPCAO statement is set to NO, and the requested 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 a SETSTATE command to start the requested subsystem. If the REQSTAT flag is set to YES, then the subsystem must be in UP or RUNNING status for the EVJESHUT to be issued. With REQSTAT=YES, 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 two sample CLISTs: EVJECCAL (CLIST) and EVJERCAL (REXX).

With APAR OW23552 the OPCACAL command shows message EVJ440I with the century in four-digit-format `mm/dd/yyyy`, for example, 05/22/1997.

Format

```
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.

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 PF key, making it more user-friendly.

Format

```
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 the NetView operation. Call this routine to perform the necessary cleanup and to post the status back to OPC. OPCACOMP combines the OPCAPOST function with an update to the OPC Automation status file for user-written subsystem operations.

Format

```
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

Use this command in your extensions to OPC Automation.

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

```
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/A application group name — up to 8 characters.

OWNER=*name*
OPC/A 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 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

```
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.

OPNO=*nnnn*
Operation number — 4 digits (required).

STATUS=*s*
Occurrence status (required). Valid statuses:

R Ready
S Started
C Completed
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=*value*
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 for Occurrence Changes

```
OPCAMOD SUBSYS=subsystem,ADID=id,IA=yymmddhhmm,
        IANEW=yymmddhhmm,DEADLINE=yymmddhhmm,PRIORITY=nnnn,
        ERRCODE=cccc,STATUS=s
```

Parameters for Occurrence Changes

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:

W Waiting
C Completed

Example 1: Occurrence Change

```
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 2: Occurrence Change

```
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 DRK/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

```
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 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 to manipulate the availability of OPC special resources. It is similar to the SRSTAT operator command, as presented in the *SA OS/390 OPC Automation Operator and Scheduler Reference*.

Format

```
OPCSRST SUBSYS=subsys,SRNAME=srname,AVAIL=Y|N
```

Parameters

subsys
Subsystem ID — 4 characters.

srname
Special resource name — up to 44 chracters.

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 Never got an SRNAME keyword
1010 Never got an AVAIL keyword
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.

Chapter 11. Control File Entries Used by OPC Automation

This section shows the following control file entries used by OPC Automation:

- OPCA CODE
- OPCACMD
- OPCA DOMAINID
- ENVIRON OPCAO
- OPCAPARM
- OPCA PCS

OPCA CODE

Purpose

The OPCA CODE entry defines the parameters used for various requests.

Format

```
subsys OPCA CODE=(request,parm1, parm2,'expstatus,timerint,timerid')
```

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.

expstatus
Expected status of the subsystem at the completion of the request. The expstatus value must be one of the following values: UP, RUNNING or CTLDOWN.

timerint
Timer interval in minutes. The maximum value permitted is 1439 (24 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*.

Usage Notes

This entry processes commands in more complex operations of OPC Automation.

Example 1

```
RMF  OPCA,CODE=(START,,,'UP,3,RMFUTMER')
```

Example 2

```
RMF  OPCA,CODE=(START,,,'UP,3,RMFUTMER'),
          CODE=(STOP,,,'CTLDOWN,2,RMFDTMER')
```

Example 3

In this example, an additional parameter was added to the START command in OPC to indicate that an alternate procedure is to be used to start the subsystem. This OPCA code entry is used in conjunction with a user-written CLIST, specified in the OPCACMD entry. An OPCAPARM entry is also required. See the OPCAPARM example 3 on page “Example 3” on page 74 for details of that entry. See the OPCACMD example 3 on page “Example 3” on page 67 for details of that entry and the sample CLIST.

This example shows two entries, one for a warm start, and one for cold start, of a subsystem called CICS1. The warm and cold parameters were added to the operations text in OPC, to passed over to OPC Automation. Presumably the two operations will take differing amounts of time, so the timer intervals are different.

```
CICS1     OPCA,CODE=(START,COLD,,'UP,10,CICS1TMR')
CICS1     OPCA,CODE=(START,WARM,,'UP,5,CICS1TMR')
```

Example 4

```
RMF  OPCA,CODE=(RECYCLE,,,'UP,5,RMFRTIMER')
```

This example shows a RECYCLE type of operation (bring the subsystem down and restart it immediately) being defined. See also Example 5 on page “Example 5” on page 68 for the corresponding OPCACMD definition.

OPCACMD

Purpose

The OPCACMD entry specifies the actual automation command to be issued for a request.

Format

```
subsys OPCACMD,CMD=(request,parm1,command)
```

Parameters

request
Request specified in the OPC-operation definition.

parm1
Parameter 1 to be used by the request.

command
Actual command to be built.

If the command contains imbedded blanks or commas, enclose it in single quotes. If the command contains single quotes, enclose it in double quotes.

Usage Notes

This entry is necessary for subsystems that OPC Automation manages. Use SA OS/390 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

```
RMF  OPCACMD,CMD=(START,,'SETSTATE RMF,RESTART,START=YES')
```

Example 2

```
RMF  OPCACMD,CMD=(START,,'SETSTATE RMF,RESTART,START=YES'),
             CMD=(STOP,,'SHUTSYS RMF,VERIFY=NO,RESTART=CTL,SCOPE=ONLY')
```

Example 3

In this example, an additional parameter was added to the START command in OPC to indicate that an alternate procedure is to be used to start the subsystem. This OPCACMD entry is used in conjunction with a user-written CLIST, called MYCLIST, which is given below as and example. An OPCAPARM entry is also required, as is an OPCA CODE entry.

See the OPCAPARM example 3 on page “Example 3” on page 74 for details of that entry. See also the OPCA code example 3 on page “Example 3” on page 66 for details of that entry.

In the example below, &EHKVAR1 contains the WARM or COLD parameter passed from OPC:

```
CICS1 OPCACMD,CMD=(START,,'MYCLIST CICS1 &EHKVAR1')
```

In the MYCLIST sample below, WARM and COLD are passed as parameters, though WARM is taken by default if COLD is not used. This example is a sample only and could be expanded in your environment to include other parameters and additional function:

```
/* MYCLIST SAMPLE  */
PARSE UPPER ARG CICSNAME STARTTYPE
'SETSTATE 'CICSNAME',RESTART,START=NO,SCOPE=ONLY'
 IF STARTTYPE = 'COLD'
    THEN 'MVS S 'CICSNAME',COLD'
 ELSE 'MVS S 'CICSNAME',WARM'
EXIT 0
```

Example 4

```
JOBX OPCACMD,CMD=(UXCINITS,,'MVS $TI20-30,C=P')
```

This example shows a user extension (see “Nonsubsystem Operations” on page 84) in which an operation for JOBX sends the request UXCINITS over from OPC. The 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 5

```
RMF  OPCACMD,CODE=(RECYCLE,,'EVJESHUT RMF ONLY')
```

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 a SHUTSYS command and verify that it is accepted or else post the operation in error. See also Example 4 on page “Example 4” on page 66 for the corresponding OPCA CODE definition.

OPCA DOMAINID

Purpose

The OPCA DOMAINID automation control file relates an OPC automatic workstation to a NetView domain ID.

Format

```
OPCA DOMAINID
         [,CODE=(workstation,,,domainid)]
         [,CODE=(workstation,,,domainid)]
```

Parameters

workstation
OPC definition — 4 characters.

Workstation names are of the form NV*xx* where *xx* is any 2 characters.

domainid
Specify the parameter as either a NetView domain ID (up to five characters) or the keyword SYSPLEX.

If a domain ID is specified, it indicates that all requests via the specified OPC workstation are to be handled by the specified domain.

If SYSPLEX is specified, then each request initiates a search of all known System Automation domains in the local sysplex (the sysplex where the OPC Controller is running) for the job name in the request. The request is then forwarded to the domain which has the application defined and in the "most up" status.

Usage Notes

Make the workstations unique; however, they may map to the same NetView.

The use of the SYSPLEX keyword requires the following on both controller and tracker systems:

- OPC Automation APAR OW35607
- System Automation OS/390 (SA OS/390) Version 1 Release 3 or later

This entry is required at every domain where an OPC Controller may run.

Example 1

```
OPCA    DOMAINID,
          CODE=(NV06,,,AOFS6),
          CODE=(NV00,,,AOF01),
          CODE=(NV01,,,XBAOC)
```

This example shows three OPC workstations mapped to their respective NetView domains.

Example 2

```
OPCA    DOMAINID,
          CODE=(NV06,,,AOFS6),
          CODE=(NV00,,,SYSPLEX),
          CODE=(NV01,,,XBAOF),
          CODE=(NV02,,,XBAOF)
```

This example shows a NetView domain with more than one workstation defined for a domain and a SYSPLEX entry.

ENVIRON OPCAO

Purpose

The ENVIRON OPCAO entry checks status on START and STOP requests, and specifies retention of critical messages.

Format

```
ENVIRON OPCAO,REQSTAT=NO|YES
              MSGKEEP=hh:mm|PERM
              OPRESET=hh:mm|NEVER
```

Parameters

REQSTAT

This entry controls checking of subsystem status on OPC Automation START, STOP, and RECYCLE requests. If not coded, OPC Automation will enforce the rule that START operations will be ended in error if the subsystem is already UP, and likewise for STOP operations if the subsystem is down. Code REQUEST=NO if you wish to have OPC Automation proceed as normal in this case. However, even with REQSTAT=NO, status values that represent conditions which require operator intervention, such as STOPPED, HALTED, BROKEN, STUCK or ZOMBIE, will not be treated as normal. In such case, a STOP or RECYCLE operation will be terminated and posted to error in OPC.

YES is the default.

MSGKEEP

This entry specifies retention of OPC Automation critical messages automated with the DFCRIT command, and OPC error messages.

The defaut is PERM, which means that messages will not be deleted on a timed basis.

PERM is the default.

OPRESET

OPRESET specifies how long the NetView interface to OPC may be unavailable before OPC Automation will not reset operations which ended in error while it was down. The NetView interface consists of both the NetView+SA OS/390+OPC Automation address space and its associated NetView SSI address space, over which requests flow from OPC to OPC Automation. When the interface is down OPC Automation sets any operations destined for a NetView workstation to error status with an error code of UNTV. If no value is coded, or OPRESET=NEVER is coded, then no operations will be reset when the interface becomes available again.

NEVER is the default.

Usage Notes

A time value is required for OPCRESET. Otherwise, OPC Automation will not automatically reset operations which failed with a UNTV error code when it, or OPC, is restarted.

Example 1

```
ENVIRON OPCAO,REQSTAT=YES,
              MSGKEEP=04:00,
              OPRESET=00:30
```

In this example, subsystem operations will be checked for current status (REQSTAT), and if the subsystem is already in the expected status coded on the OPCA CODE statement, then the operation will be posted in error to OPC with a return code of U001. In addition, the critical messages put into the display facility (SDF) with DFCRIT, will be automatically purged from SDF (on the target and the focal point) every 4 hours, without operator intervention. Finally, should any operations fail with a UNTV error code because NetView or the PPI is unavailable when they became READY in OPC, they will be automatically reset only if NetView or the PPI is brought back within 30 minutes of their arrival in READY status.

OPCAPARM

Purpose

The OPCAPARM entry defines the additional parameters used for various requests. This entry is entirely optional and need not be coded if the OPCA CODE entry is sufficient to define all the necessary parameters.

Format

```
subsys OPCAPARM,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.

Usage Notes

This entry processes commands in more complex operations of OPC Automation.

Example 1

```
RMF OPCAPARM,CODE=(START,,,',,')
```

In this example, no additional parameters are needed for the requested operation (START). An entry of this sort is entirely optional, and need not be coded at all.

Example 2

```
OPCA OPCAPARM,CODE=(UXCICSRQ,TESTCICS,ALLFILES,'CX10AA,DFHGRP1,')
```

In this example, OPC schedules UXCICSRQ, a user extension, and passes TESTCICS and ALLFILES as parameters in the operation text. The user-coded module UXCICSRQ utilizes CIX10AA and DFHGRP1 when it builds its commands.
See “Nonsubsystem Operations” on page 84.

Example 3

In this example, an additional parameter was added to the START command in OPC to indicate that an alternate procedure is to be used to start the subsystem. This OPCAPARM entry is used in conjunction with a user-written CLIST, specified in the OPCACMD entry. An OPCA code entry is also required. See the OPCA code in “Example 3” on page 66 for details of that entry. See the OPCACMD example in “Example 3” on page 67 for details of that entry and the sample CLIST.

```
CICS1     OPCAPARM,CODE=(START,COLD,,',,')
CICS1     OPCAPARM,CODE=(START,WARM,,',,')
```

OPCA PCS

Purpose

The OPCA PCS automation control file entry locates the NetView domain on which the OPC Controller resides and specifies the MVS subsystem name for the OPC Controller system.

Format

```
OPCA PCS,
      DOMAIN=domainid,SUBSYS=pcsname
```

Parameters

domainid

Specify the parameter as either a NetView domain ID (up to five characters) or the keyword SYSPLEX.

If a domain ID is specified, it indicates that the OPC Controller always runs on the specified domain.

The keyword SYSPLEX indicates that the OPC Controller may be running on any one of the systems in the local sysplex.

pcsname

The MVS subsystem name of the OPC Controller as defined in the IEFSSNxx member.

Note: When defining the OPC Controller to AOC this name must be used as the subsystem name and job name.

Usage Notes

The use of the SYSPLEX keyword requires the following on both controller and tracker systems:

- OPC Automation APAR OW35607
- System Automation OS/390 (SA OS/390) Version 1 Release 3 or later.

Prior to OPC Automation APAR OW35607 this entry is required at both OPC Controller and OPC Tracker domains.

After applying OPC Automation APAR OW35607 this entry is required:

- At every domain where an OPC Controller may run
- Within the same sysplex as the OPC Controller, at any other domain where it is desired to run OPC Automation operator commands (e.g., OPCACMD).

This OPCA PCS statement is used by users of all variants of OPC (TME 10 OPC, OPC/ESA and OPC/A).

Example 1

```
OPCA PCS,
      DOMAIN=AOF01,
      SUBSYS=OPCA
```

Example 2

```
OPCA PCS,
      DOMAIN=SYSPLEX,
      SUBSYS=OPCC
```

Chapter 12. Data Areas

This chapter shows the following:

- Subsystem status file OPC Automation entry (EVJSTS)
- Requestor ID block (EHKVAR9)
- Request buffer

Subsystem Status File OPC Automation Entry (EVJSTS)

Table 4. Subsystem Status File OPC Automation Entry (EVJSTS)

Field	Length	Value	Description
Last Completed	1	U|D	Last successfully completed status request is UP or CTLDOWN
Sequence Number #1	4	Numeric 0000–9999	Sequence number of last completed status resulting in an UP or CTLDOWN status
Timer Flag	1	null|1|0	If 1, this subsystem operation was ended by timer; if null, TIMEREND task never used for this subsystem.
Comp	1	null|1|0	If 1, this subsystem operation was ended by status change; if null, completion task never used for this subsystem
Sequence Number #2	4	Numeric 0000–9999	Sequence number of present request
Request Buffer	8		STOP/START/RECYCLE
PARM1	8		Optional for user written code
PARM2	8		Optional for user written code
Expected Result	24		UP/CTLDOWN
Timer ID	8	pointer|0	If it exits, pointer to outstanding timer ID for this request; if none, 0

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 5 shows the lengths and values of the variables.

Table 5. Lengths and Values of Task Global Variable (EHKVAR9)

Variable	**Name of Subsystem**	**Sequence #**	**Module Name**	**Domain ID**
Length (characters)	8	4	8	5
Values	AOC 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 5.

```
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 6 shows the request buffer layout for standard subsystem operations:

Table 6. 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 7 shows the request buffer layout for nonsubsystem, user extension (UXaaaaaa) operations:

Table 7. Request Buffer Layout for Nonsubsystem, 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. Figure 6 on page 14 shows an example of a resulting request buffer.

Chapter 13. 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 OS/390 command or function that performs for an automation-controlled subsystem.
- 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 OS/390-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.

Relating to the SA OS/390 Defined Subsystem

OPC Automation provides support for stopping and starting the SA OS/390-defined subsystems. Certain environments require you to issue a command or to perform a function outside the scope of SA OS/390. 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 automation control file. 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 issuing 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 OS/390 command needs issuing, use the normal OPC Automation functions. In the request, instead of issuing the SA OS/390 command, OPC passes control to a user CLIST or command processor specified in the automation control file. The request module passes all information available to the user-supplied module. When the user-written module completes processing, it calls the OPCACOMP command to indicate the completion of the operation. Figure 24 shows the flow including the user responsibilities.

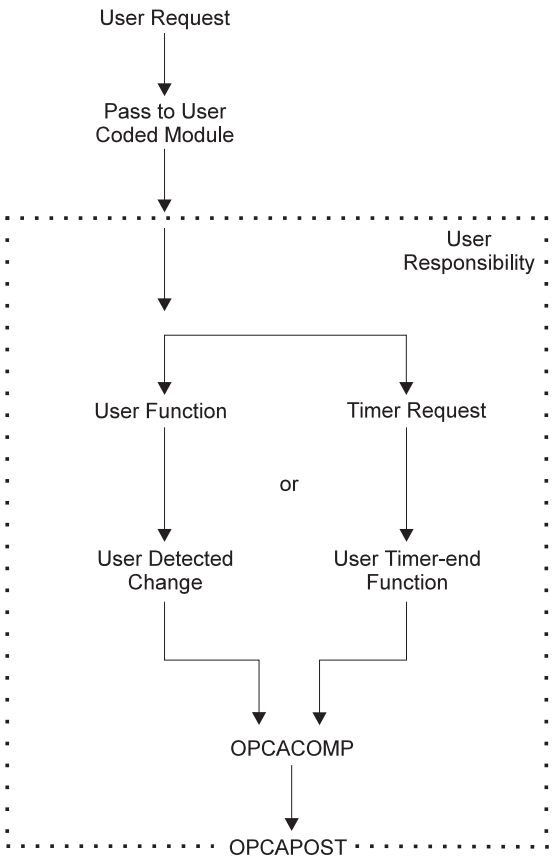


Figure 24. Request Flow for a User Function

If the user-supplied module does not issue any commands and returns control to OPC Automation synchronously, the user is responsible for completing the operation by calling OPCACOMP once the results of these commands are analyzed. To simplify implementation, you may plan to only use the timer function or 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.

The timer and completion validation are a user responsibility. Once the user code determines that the function is completed, OPC Automation calls the OPCACOMP module with a good completion code, or a bad completion and associated return code. The OPCACOMP module ensures that actions are accomplished in the correct sequence, does some housekeeping, updates the SA OS/390 status file, and calls the OPCAPOST command processor to return the specified completion code to OPC.

Parameters Passed to User-Supplied Module

When OPC Automation calls the user-supplied module from the request function, several OPC Automation and user parameters are passed in NetView task global variables. Following is summary of these variables:

&EHKVAR1 Parameter 1 value

&EHKVAR2 Parameter 2 value

&EHKVAR7 Expected status, timer interval, timer id

&EHKVAR8 OPC

&EHKVAR9 Requestor ID block; see “Requestor ID Block (EHKVAR9)” on page 78

The automation control file parameter file entry stores the values of Parameter 1, Parameter 2, and the name of the user-supplied module.

Do not modify the information in the task globals. OPC uses information in &EHKVAR7 if the timer function check module is purged. The SA OS/390 problem determination uses information in &EHKVAR8. The user-supplied module requires information in &EHKVAR9 module to call OPCACOMP correctly when it completes.

Completing a User-Supplied Module (OPCACOMP)

After the user-supplied module completes its work, it should update the subsystem status file buffer and set the COMP flag. OPC also needs posting with the results of the user-supplied module. To avoid errors and to ensure synchronization, OPC Automation carries out these tasks in the OPCACOMP function.

Flow of Control

Code these operations similar to START and STOP operations included with OPC Automation. The automation control file contains the name of the command. These entries are in the following form:

```
jobname OPCACMD,CMD=(xxxxxxxx,,'userfunc &EHKVAR1')
```

In the preceding example, the *xxxxxxxx* represents any character string meaningful to the scheduler. The *xxxxxxxx* string consists of up to eight characters and should not start with the characters ‘UX’. These 8 characters are arbitrary, but the USERFUNC is not. USERFUNC here stands for the name of a user-written CLIST, which will be passed the data in EHKVAR1.

Specify a timer function on the OPCAPARM and OPCA CODE entries, as shown in the following example:

```
jobname OPCAPARM,CODE=(xxxxxxxx,,',,utimermod')
jobname OPCA,CODE=(xxxxxxxx,,'ustat,3,utimer')
```

You, the user, must supply the UTIMERMOD module to determine success or failure of the operation. OPC Automation passes USTAT, a value to use as necessary, with three minutes as the interval. UTIMER is the unique timer ID. Your UTIMERMOD routine must use these values to construct its own timer to call OPCACOMP to verify completion.

When your timer module is called, it is passed the following parms in the order shown:

1. Jobname
2. Expected status
3. Application ID
4. Workstation ID
5. Operation number

Your timer routine calls OPCACOMP with the jobname, operation number, and result of C or E. If the operation is not successful, include an error code containing four characters of your choosing.

If you do not code the OPCA and OPCAPARM control file entries correctly, complete with user-timer specifications, the variables EHKVAR7, EHKVAR8, and EHKVAR9 are not set and your OPCACOMP request fails.

Nonsubsystem Operations

Operations of this type, containing user extensions named UX*xxxxxx*, allow you to perform commands that are independent of a specific subsystem. Figure 25 shows the flow for these types of operations.

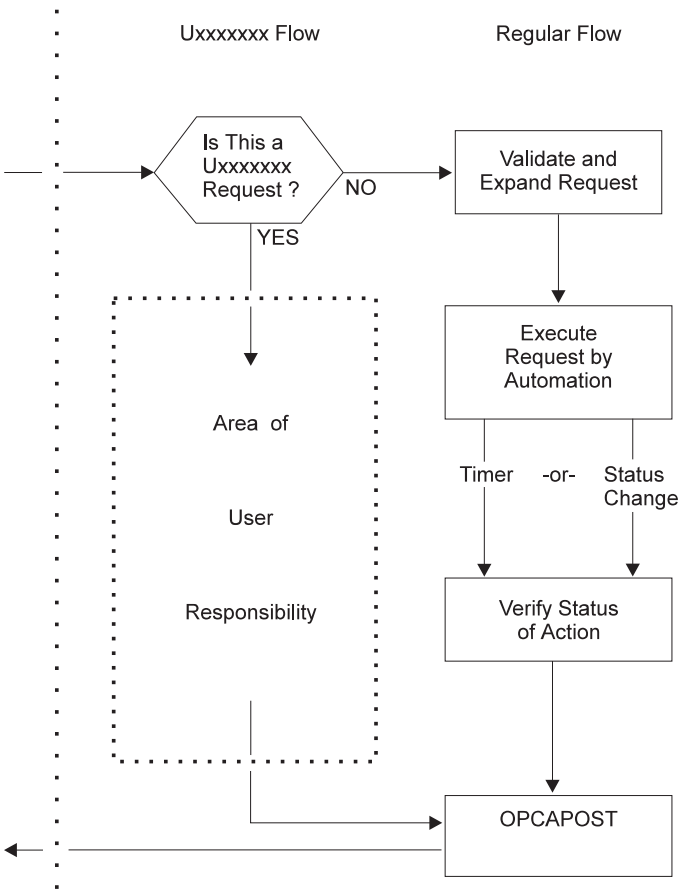


Figure 25. User Exit UXxxxxxx Flow

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 26 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.

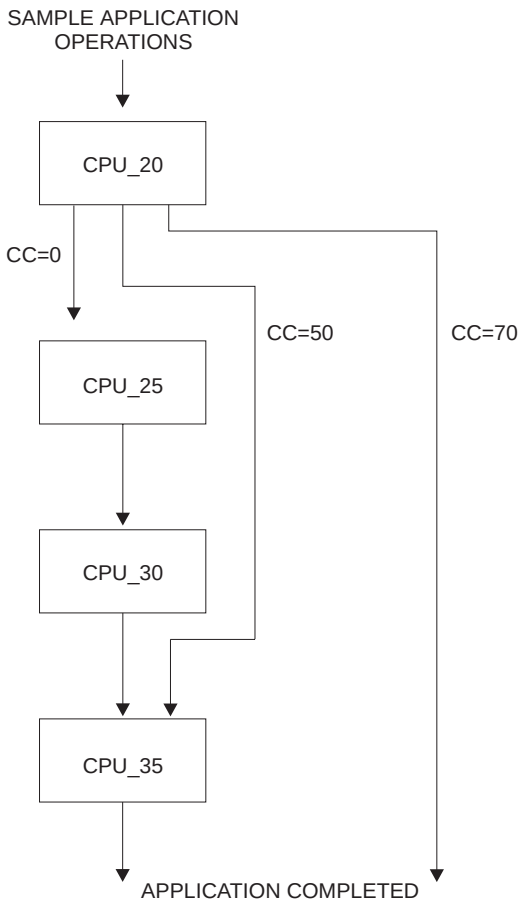


Figure 26. Condition Code Driven Application Flow

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

OPC invokes the UX*xxxxxx* user exit by specifying UX*xxxxxx* as the request text in OPC. The name of the command is found by an entry in the automation control file. These entries are in the following form:

```
OPCA  OPCACMD,CMD=(UXxxxxxx,,'userfunc &EHKVAR1')
```

Code different commands by jobname, as the following shows:

```
JOB1  OPCACMD,CMD=(UXxxxxxx,,'userfunc &EHKVAR1')
JOB2  OPCACMD,CMD=(UXxxxxxx,,'userfunc &EHKVAR1')
```

In the preceding examples, the *xxxxxx* represents any character string, meaningful to the scheduler and containing up to six characters. OPC Automation calls the USERFUNC, the name of the user command module.

The flow of control is a user responsibility for a user exit. When coding a user exit, take care to ensure that the flow is maintained. Failure to do so halts the OPC application until intervention occurs.

Parameters Passed to a User Exit

When OPC calls the UX*xxxxxx* command, OPC passes a NetView task global (&EHKVAR1), containing the request buffer, to the command as input parameters.

Unlike a subsystem operation, the request buffer for a user extension contains the entire operation description field (TXTOP) from OPC. See Table 6 on page 79 and Table 7 on page 80 for details.

Completing a User Exit (OPCAPOST)

An OPC Automation user exit is terminated by issuing the OPCAPOST command with a completion code. This returns the control of the application processing to OPC. Figure 27 on page 87 shows more details.

```
/*** REXX ***********************************************************/
/*------------------------------------------------------------------*/
/*                                                                  */
/*      COPYRIGHT= 5685-151                                         */
/*                 CONTAINS RESTRICTED MATERIALS OF IBM             */
/*                 (C) COPYRIGHT IBM CORP. 1995                     */
/*                 LICENSED MATERIALS - PROPERTY OF IBM             */
/*                 REFER TO COPYRIGHT INSTRUCTIONS                  */
/*                 FORM NUMBER G120-2083.                           */
/*                                                                  */
/*------------------------------------------------------------------*/
/**  EVJERUX1                                                      */
/**                                                                */
/**  Sample clist to demonatrate using the OPCAO User Exit Calls   */
/**                                                                */
/**  This clist will check for adequate spool space before allowing */
/**  an OPC/A application to proceed.                              */
/**                                                                */
/**  OPC/A Definitions:                                            */
/**                                                                */
/**    Create an operation on a NVxx workstation with operation    */
/**    on a dummy job name (DUMMYJOB for example) with operation   */
/**    text of UXCKSPL ("UX" is fixed, the remainder can vary).    */
/**                                                                */
/**  Control File entries required:                                */
/**                                                                */
/**    DUMMYJOB  OPCACMD,CMD=(UXCKSPL,,"EVJERUX1 &EHKVAR1")        */
/**                                                                */
/** APAR#                                                          */
/** ---------------------------------------------------------------- */
/** PN10372  11/19/91   PROVIDE SAMPLE CLISTS ILLUSTRATING THE     */
/**                     USER EXIT PROCESSING CAPABILITIES.         */
/**                                                                */
/*******************************************************************/

 trace off
 parse source . Invoc Ident .
 parse upper arg parms                    /* for audit or debugging */
 "MSG LOG "Ident " : PARMS = "parms       /* purposes, a footprint  */

 /*****************************************************************/
 /* EHKVAR1 expands to the following: Calling module (EVJESPRQ)   */
 /*                                   Application Name            */
 /*                                   Workstation (NVxx)          */
 /*                                   Operation Number            */
 /*                                   Subsys (DUMMYJOB here)      */
 /*                                   Request (UXCKSPL here)      */
 /*****************************************************************/

 parse upper arg module Adname Wsname Opnum Subsys Request
```

Figure 27 (Part 1 of 2). Typical Code Required for Nonsubsystem Requests

```
/*********************************************************************/
/* Get waittime from AOC or ACO                                      */
/*********************************************************************/
 "GLOBALV GETC WAITTIME"
 if Waittime = "" then
   Waittime = 29

/*********************************************************************/
/* The body of the processing - issue an MVS $D SPL command          */
/* and wait for a response.                                          */
/*********************************************************************/

 "TRAP AND SUPPRESS MESSAGES $HASP646"
 "MVS $DSPL"
 "WAIT "Waittime" SECONDS FOR MESSAGES"

/*********************************************************************/
/* If the queue is more than 50 percent full post in error,          */
/* otherwise post success.                                           */
/*********************************************************************/
 select
   when event() = "M" then
     do
       "TRAP NO MESSAGES"
       "MSGREAD"
       "GETMLINE JES2MSG" 1
       parse var Jes2msg msgnum percent morewords
       if percent < 50 then call PostOk
                       else call PostERR
     end
   otherwise                                    /* Timeout, error,  */
     call PostERR                               /* post as failure  */
 end
exit 0
POSTOK:
"OPCAPOST ADNAME="Adname "WSNAME="Wsname "OPNUM="Opnum "TYPE=C"
return

POSTERR:
"OPCAPOST ADNAME="Adname "WSNAME="Wsname "OPNUM="Opnum "TYPE=E"||,
" ERRCODE=USPL"
return
```

Figure 27 (Part 2 of 2). Typical Code Required for Nonsubsystem Requests

Interaction with CICS Automation

The following example shows how to use the CEMTPPI command of the AOC/MVS CICS Automation Feature 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 operations:

UXCICSOP Requests CICS to open a file.

UXCICSCL Requests CICS to close a file.

Although OPC Automation requires 'UX' in the operation text, you may vary the remaining part of the name. The preceding example uses CICSOP and CICSCL for this portion of the name.

The example selects the CLIST names of CICSOPEN and CICSCLOS. Using these names, the control file entries are as follows:

```
OPCA  OPCACMD,CMD=(UXCICSOP,,'CICSOPEN &EHKVAR1')
OPCA  OPCACMD,CMD=(UXCICSCL,,'CICSCLOS &EHKVAR1')
```

The example uses the CICS subsystem name and the file name as parameters to the request. These parameters are optional and flexible. Thus, the OPC jobname field could serve as the CICS name. Figure 28 shows the 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 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    NOJOB___  UXCICSCL CICS01 PAYROLL____
```

Figure 28. Defining Sample CICS Application in OPC Automation

Figure 29 shows the REXX code for CICSOPEN and includes the source as EVJERUX2.

```
/*** REXX ***************************************************************/
/*--------------------------------------------------------------------*/
/*                                                                    */
/*      COPYRIGHT= 5685-151                                           */
/*                 CONTAINS RESTRICTED MATERIALS OF IBM               */
/*                 (C) COPYRIGHT IBM CORP. 1995                       */
/*                 LICENSED MATERIALS - PROPERTY OF IBM               */
/*                 REFER TO COPYRIGHT INSTRUCTIONS                    */
/*                 FORM NUMBER G120-2083.                             */
/*                                                                    */
/*--------------------------------------------------------------------*/
/**                                                                   */
/**  EVJERUX2/CICSOPEN                                                */
/**                                                                   */
/**                                                                   */
/**  Sample clist using AOC CICS to open a CICS file                  */
/**  APAR#                                                            */
/** ----------------------------------------------------------------- */
/**                                                                   */
/**********************************************************************/

 parse source . Invoc Ident .              /* Finds name of clist    */
 parse upper arg Parms                     /* for audit or debugging */
 "MSG LOG "Ident " : PARMS = "Parms        /* purposes, a footprint  */

 /*********************************************************************/
 /* EVJERUX2 adds a new error code of 'UCIC' to be used with the      */
 /* OPCAPOST command if the command to CICS is unsuccessful.          */
 /*                                                                   */
 /* EHKVAR1 expands to the following: Calling module (EVJESPRQ)       */
 /*                                   Application Name                */
 /*                                   Workstation (NVxx)              */
 /*                                   Operation Number                */
 /*                                   Subsys (OPC Job name)           */
 /*                                   Request (UXCICSOP)              */
 /*                                   parm1 (subsystem)               */
 /*                                   parm2 (file name)               */
 /*********************************************************************/

 trace off
 parse upper arg Module Adname Wsname Opnum Subsys Request,
   Cics_subsys Cics_filename

 /*************************************************************/
 /* Build the CEMT command.  See AOC CICS Programmer's Ref,   */
 /* SC23-3814 for more information.                           */
 /*************************************************************/

 'GLOBALV GETC WAITTIME'
 'FLUSHQ MESSAGES'
 Messages = 'EVE790* EVE1* DSI594*'
```

Figure 29 (Part 1 of 4). CICSOPEN Exec

```
/*************************************************************/
/* See AOC CICS Operator's Guide, SC23-3815, for more info.   */
/* DSI594I  WAIT state entered warning                       */
/* EVE790I  response from PPI (data in EVE791, end in EVE792)*/
/* EVE120I  command accepted for by PPI task                 */
/* EVE129E  NACK from PPI converse request to CICS           */
/* EVE136E  error on PPI request (see return code)           */
/* EVE1xx   treat all other EVE1xx messages as errors        */
/*************************************************************/
'TRAP AND SUPPRESS MESSAGES 'Messages
/*************************************************************/
/*  Note:  You may change the preceeding command to          */
/*  'TRAP  MESSAGES 'Messages                                */
/*  if you wish to see the messages returned in the log      */
/*************************************************************/
Eveselnm = Cics_subsys
"CEMTPPI "Cics_subsys" SET FILE("Cics_filename") OPEN"
if Rc = 0 then
  do
    call PROCESS_PPI_RESPONSE
    Saverc = Rc
    exit Saverc
  end
else
  do                          /* CEMPTPPI failed - tell user and quit */
    Saverc = Rc
    Evemsghdrd = 'ON'
    'GLOBALV PUTT EVEMSGHDRD'
    Errormsg = eveemsg(551,'Y',Ident,'EVJERUX2',Saverc)
    exit Saverc
  end

/*************************************************************/
/* End mainline                                              */
/*************************************************************/
```

Figure 29 (Part 2 of 4). CICSOPEN Exec

```
PROCESS_PPI_RESPONSE:
 /*******************************************************/
 /* Determine response from CEMTPPI command             */
 /*******************************************************/

 'WAIT 'Waittime' SECONDS FOR MESSAGES'
 Msgwait = Ok
 do while Msgwait = Ok
   select
     when event() = 'T' then
       do                  /* CEMPTPPI timed out - tell user and quit */
         Evemsghdrd = 'ON'
         'GLOBALV PUTT EVEMSGHDRD'
         Errormsg = eveemsg(550,'Y',Ident,'CEMTPPI')
         Msgwait = No
         return 4
       end
     when event() = 'M' then
       do                                /* What message did we get?  */
         'MSGREAD'
         'GETMSIZE  MCOUNT'
         Lbuffer = ''
         'GETMLINE LBUFFER 1'
         select
           when POS('DSI594',Lbuffer) ¬= 0 then
             'WAIT CONTINUE'
           when POS('EVE120',Lbuffer) ¬= 0 then
             'WAIT CONTINUE'
           when (POS('EVE129',Lbuffer) ¬= 0 |  ,
             POS('EVE136',Lbuffer) ¬= 0 ) then
               do
                 /*   PPI ERROR */
                 Errormsg = evjemsg(003,'Y','CEMTPPI',Eveselnm,'PPI_NOT_ACTIVE')
     "OPCAPOST ADNAME="Adname "WSNAME="Wsname "OPNUM="Opnum "TYPE=E"||,
                   " ERRCODE=UCIC"
                 exit Rc
                 Msgwait = No
                 return 4
               end
           when POS('EVE1',Lbuffer) ¬= 0 then
             do
               Errormsg = Lbuffer
               Msgwait = No
               return 8
             end
           when POS('EVE790',Lbuffer) ¬= 0 then
             do
               call PPI_OUTPUT
               Msgwait = No
             end
           end
       end
   end
   if Msgwait=Ok then
     'WAIT CONTINUE'
 end
return
```

Figure 29 (Part 3 of 4). CICSOPEN Exec

```
PPI_OUTPUT:
 do I = 1 to Mcount
   'GETMLINE LBUFFER ' I+1
   /*******************************************************/
   /* EVE790I output is in unformatted buffer             */
   /* scan to see what's in it                            */
   /*******************************************************/
   select
     when POS('EVE790',Lbuffer) ¬= 0 then
       do
         nop                              /* This is just the heading */
       end
     when POS('RETURN CODE 000 FROM CEMT COMMAND',Lbuffer) ¬= 0 then
       do
         nop                                      /* Normal response  */
       end
     when POS('EVE792',Lbuffer) ¬= 0 then
       do
         'TRAP NO MESSAGES'
         'FLUSHQ'
         return 0
       end
     when POS('EVE1',Lbuffer) ¬= 0 then
       do
         'TRAP NO MESSAGES'
         parse var Lbuffer X Cicsmsg
         Errormsg = evjemsg(030,'Y','CEMTPPI',Cicsmsg)
         return 16
       end
     when POS(' NOT ',Lbuffer) ¬= 0 then
       do
         /* File Not found                */
         Errormsg = evjemsg(003,'Y','CEMTPPI',Cics_filename,'NOT_FOUND')
         "OPCAPOST ADNAME="Adname "WSNAME="Wsname "OPNUM="Opnum "TYPE=E"||,
           " ERRCODE=UCIC"
         exit Rc
       end
     when POS(' OPE ',Lbuffer) ¬= 0 then
       do
         /* File is open                  */

         Errormsg = evjemsg(002,'Y','CEMTPPI',"'"Cics_filename" OPEN'")
         "OPCAPOST ADNAME="Adname "WSNAME="Wsname "OPNUM="Opnum "TYPE=C"
         exit Rc
       end
     when POS(' CLO',Lbuffer) ¬= 0 then
       do
         /* File is closed                */
         Errormsg = evjemsg(003,'Y','CEMTPPI',Cics_filename,'CLOSED')
         "OPCAPOST ADNAME="Adname "WSNAME="Wsname "OPNUM="Opnum ||,
           " TYPE=E" " ERRCODE=UCIC"
         exit Rc
       end
     otherwise
       do
         /* Unexpected response in buffer  */
         Errormsg = evjemsg(030,'Y','CEMTPPI',Lbuffer)
       end
   end
 end
return
```

Figure 29 (Part 4 of 4). CICSOPEN Exec

The CICSCLOS exec is identical except for the command text on the CEMTPPI command.

Interaction with IMS Automation

The following example shows how to use the IMSCMD of the AOC/MVS IMS Automation Feature 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 EVJERUX4 and EVJERUX5. If CICS 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 operations:

UXIMSSDB Requests to start a database.

UXIMSPDB Requests to stop a database.

Although OPC Automation requires “UX” in the operation text, you may vary the remaining part of the name. This example uses IMSSDB and IMSPDB for this portion of the name.

The example selects the CLIST names EVJERUX4 and EVJERUX5. Using these names, the control file entries are as follows:

```
OPCA  OPCACMD,CMD=(UXIMSSDB,,'EVJERUX4 &EHKVAR1')
OPCA  OPCACMD,CMD=(UXIMSPDB,,'EVJERUX5 &EHKVAR1')
```

The example uses OPCA to identify the request to be issued, but you could also use the OPC job name field instead. These parameters are the IMS subsystem name and the database name as parameters to the request. Figure 30 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    IMSSTART  UXIMSSDB IMS05Z_________
```

Figure 30. Defining Sample IMS Application in OPC Automation

Figure 31 shows the REXX code for EVJERUX4, which is used to open an IMS database.

```
/*** REXX ***************************************************************/
/*----------------------------------------------------------------------*/
/*                                                                      */
/*      COPYRIGHT= 5685-151                                             */
/*                 CONTAINS RESTRICTED MATERIALS OF IBM                 */
/*                 (C) COPYRIGHT IBM CORP. 1995                         */
/*                 LICENSED MATERIALS - PROPERTY OF IBM                 */
/*                 REFER TO COPYRIGHT INSTRUCTIONS                      */
/*                 FORM NUMBER G120-2083.                               */
/*                                                                      */
/*----------------------------------------------------------------------*/
/**                                                                     */
/**  EVJERUX4                                                           */
/**                                                                     */
/**                                                                     */
/**  Sample clist using AOC IMS to START a database in IMS              */
/**  APAR#                                                              */
/** ------------------------------------------------------------------- */
/**                                                                     */
/************************************************************************/

 parse source . Invoc Ident .               /* Finds name of clist    */
 parse upper arg Parms                      /* for audit or debugging */
 "MSG LOG "Ident " : PARMS = "Parms         /* purposes, a footprint  */

 /*********************************************************************/
 /* EVJERUX4 adds a new error code of 'UIMS' to be used with the     */
 /* OPCAPOST command if the command to IMS  is unsuccessful.         */
 /*                                                                   */
 /* EHKVAR1 expands to the following: Calling module (EVJESPRQ)      */
 /*                                   Application Name                */
 /*                                   Workstation (NVxx)              */
 /*                                   Operation Number                */
 /*                                   Subsys (OPC Job name)           */
 /*                                   Request (UXIMSSDB)              */
 /*                                   parm1 (subsystem)               */
 /*                                   parm2 (database name)           */
 /*********************************************************************/

 trace o
 parse upper arg Module Adname Wsname Opnum Subsys Request,
   Ims_subsys Ims_dbname

 'GLOBALV GETC WAITTIME'
 'FLUSHQ MESSAGES'
 Messages = 'EVI690* EVI1* DSI594*'
```

Figure 31 (Part 1 of 4). Exec to Open an IMS Database

```
/************************************************************/
/* See AOC IMS Operator's Guide, SC23-3818, for more info.  */
/* DSI594I  WAIT state entered warning                      */
/* EVI690I  response from PPI (data in EVI691, end in EVI692)*/
/* EVI120I  command accepted for by PPI task                */
/* EVI150E  command in progress by IMS (timeout)            */
/* EVI129E  NACK from PPI converse request to CICS          */
/* EVI136E  error on PPI request (see return code)          */
/* EVI1xx   treat all other EVI1xx messages as errors       */
/************************************************************/
'TRAP AND SUPPRESS MESSAGES 'Messages
/************************************************************/
/*  Note:  You may change the preceeding command to         */
/*  'TRAP  MESSAGES 'Messages                               */
/*  if you wish to see the messages returned in the log     */
/************************************************************/

/************************************************************/
/* Build the IMSCMD /START command.  See AOC IMS Programmer's */
/* Reference  SC23-3817 for more information.               */
/* Since the results of the /START are NOT returned across  */
/* the PPI, we must also do /DIS to verify that the command  */
/* succeeded.                                               */
/************************************************************/
Eviselnm = Ims_subsys
"IMSCMD "Ims_subsys" /START DATABASE "Ims_dbname
if Rc = 0 then
  do
    call PROCESS_PPI_RESPONSE
  end
else
  do                          /* IMSCMD failed - tell user and quit */
    Saverc = Rc
    Evimsghdrd = 'ON'
    'GLOBALV PUTT EVIMSGHDRD'
    Errormsg = eviemsg(551,'Y',Ident,'EVJERUX4',Saverc)
    exit Saverc
  end

/************************************************************/
/* Build the IMSCMD /DIS command.                           */
/************************************************************/

"IMSCMD "Ims_subsys" /DIS DATABASE "Ims_dbname
if Rc = 0 then
  do
    call PROCESS_PPI_RESPONSE
    Saverc = Rc
    exit Saverc
  end
else
  do                          /* IMSCMD failed - tell user and quit */
    Saverc = Rc
    Evimsghdrd = 'ON'
    'GLOBALV PUTT EVIMSGHDRD'
    Errormsg = eviemsg(551,'Y',Ident,'EVJERUX4',Saverc)
    exit Saverc
  end

/************************************************************/
/* End mainline                                             */
/************************************************************/
```

Figure 31 (Part 2 of 4). Exec to Open an IMS Database

```
PROCESS_PPI_RESPONSE:
 /*********************************************************/
 /* Determine response from IMSCMD command                */
 /*********************************************************/

 'WAIT 'Waittime' SECONDS FOR MESSAGES'
 Msgwait = Ok
 do while Msgwait = Ok
   select
     when event() = 'T' then
       do                    /* IMSCMD timed out - tell user and quit */
         Evimsghdrd = 'ON'
         'GLOBALV PUTT EVIMSGHDRD'
         Errormsg = eviemsg(550,'Y',Ident,'IMSCMD')
         Msgwait = No
         return 4
       end
     when event() = 'M' then
       do                                  /* What message did we get?  */
         'MSGREAD'
         'GETMSIZE  MCOUNT'
         Lbuffer = ''
         'GETMLINE LBUFFER 1'
         select
           when POS('DSI594',Lbuffer) ¬= 0 then
             'WAIT CONTINUE'
         when POS('EVI120',Lbuffer) ¬= 0 then
           'WAIT CONTINUE'
         when (POS('EVI129',Lbuffer) ¬= 0 |  ,
           POS('EVI136',Lbuffer) ¬= 0 ) then
             do
               /*   PPI ERROR */
               Errormsg = evjemsg(003,'Y','IMSCMD',Eviselnm,'PPI_NOT_ACTIVE')
     "OPCAPOST ADNAME="Adname "WSNAME="Wsname "OPNUM="Opnum "TYPE=E"||,
                 " ERRCODE=UIMS"
               exit Rc
               Msgwait = No
               return 4
             end
         when POS('EVI1',Lbuffer) ¬= 0 then
           do
             Errormsg = Lbuffer
             Msgwait = No
             return 8
           end
         when POS('EVI690',Lbuffer) ¬= 0 then
           do
             call PPI_OUTPUT
             Msgwait = No
           end
         end
       end
   end
   if Msgwait=Ok then
     'WAIT CONTINUE'
 end
return
```

Figure 31 (Part 3 of 4). Exec to Open an IMS Database

```
PPI_OUTPUT:
 do I = 1 to Mcount
   'GETMLINE LBUFFER ' I+1
   /*********************************************************/
   /* EVI690I output is in unformatted buffer               */
   /* scan to see what's in it                              */
   /*********************************************************/
   select
     when POS('EVI690',Lbuffer) ¬= 0 then
       do
         nop                              /* This is just the heading */
       end
     when POS('EVI692',Lbuffer) ¬= 0 then
       do
         'TRAP NO MESSAGES'
         'FLUSHQ'
         return 0
       end
     when POS('EVI150E',Lbuffer) ¬= 0 then       /* command running   */
       do
         return 0
       end
     when POS('EVI1',Lbuffer) ¬= 0 then
       do
         'TRAP NO MESSAGES'
         parse var Lbuffer X Imsmsg
         Errormsg = evjemsg(030,'Y','IMSCMD',Imsmsg)
         return 16
       end
     when POS(' INVALID ',Lbuffer) ¬= 0 then
       do
         /* Database Not found                 */
         Errormsg = evjemsg(003,'Y','IMSCMD',Ims_dbname,'NOT_FOUND')
         "OPCAPOST ADNAME="Adname "WSNAME="Wsname "OPNUM="Opnum "TYPE=E"||,
           " ERRCODE=UIMS"
         exit Rc
       end
     when POS(' STOPPED',Lbuffer) ¬= 0 then
       do
         /* Database is stopped            */
         Errormsg = evjemsg(003,'Y','IMSCMD',Ims_dbname,'STOPPED')
         "OPCAPOST ADNAME="Adname "WSNAME="Wsname "OPNUM="Opnum ||,
           " TYPE=E" " ERRCODE=UIMS"
         exit Rc
       end
     when POS(' UP ',Lbuffer) ¬= 0 then
       do
         /* Database is started by default */

         Errormsg = evjemsg(002,'Y','IMSCMD',"'"Ims_dbname" STARTED'")
         "OPCAPOST ADNAME="Adname "WSNAME="Wsname "OPNUM="Opnum "TYPE=C"
         exit Rc
       end
     when POS('EVI691I',Lbuffer) ¬= 0 then nop
     otherwise
       do
         /* Unexpected response in buffer  */
         Errormsg = evjemsg(030,'Y','IMSCMD',Lbuffer)
       end
   end
 end
return
```

Figure 31 (Part 4 of 4). Exec to Open an IMS Database

Part 4. Planning and Installation

Chapter 14. Installation

Complete the steps listed below for installation. Use the following table to track your completed work:

Table 8. Installation Check List	
√	**Step**
	1. Load OPC Automation libraries from the distribution tape.
	2. Modify MPFLST.
	3. Update IEAAPF*xx* in SYS1.PARMLIB to provide APF authorization.
	4. Define subsystem allocatable consoles.
	5. Check the subsystem name table to ensure that the NetView SSI is first.
	6. Add libraries to OPC and recycle.

Step 1: Load OPC Automation Libraries

Using the information in OPC Automation program directory, load OPC Automation libraries from the distribution tape.

Step 2: Updating MPFLST

Add the following message entry in MPFLSTxx in SYS1.PARMLIB to trap all TME 10 OPC and OPC/ESA messages:

```
EQQ*,SUP(NO),AUTO(YES)
```

OPC/A users must add these entries instead of the one above:

```
DRK*,SUP(NO),AUTO(YES)
CSY*,SUP(NO),AUTO(YES)
```

All users must add an additional entry for OPC Automation messages:

```
EVJ*,SUP(NO),AUTO(YES)
```

This step is necessary to permit MVS console messages to flow to NetView where they may be automated.

Step 3: Updating IEAAPF*xx* in SYS1.PARMLIB

You must APF authorize all the NetView libraries from the //STEPLIB concatenation in the NetView start procedure. To accomplish this, a library's name must appear in the list of authorized libraries in IEAAPF*xx*, the APF member of SYS1.PARMLIB. After you update IEAAPF*xx*, you must re-IPL MVS.

Ensure that the USERLINK library used when link-editing OPC Automation is authorized. If you specify an unauthorized library on a STEPLIB or concatenate

unauthorized libraries with authorized libraries, all libraries are treated as if they are unauthorized.

Note: For more information on APF authorization, refer to *MVS/ESA Initialization and Tuning*, as appropriate for your system.

Step 4: Defining Subsystem Allocatable Consoles

The NetView program's use of the MVS subsystem interface allows issuing MVS system operator commands from the operator station task used by NetView operators and from an automation task. For each active task that can issue MVS system operator commands, a subsystem allocatable console is required for NetView.

Each OPC Automation operator can use a subsystem allocatable console. Make sure you increase the number of subsystem allocatable consoles by the number of OPC automation operators.

Note: The *MVS/ESA Installation: System Generation Reference* contains information on defining subsystem allocatable consoles for your system. Also refer to the *MVS/ESA Input Output Configuration Program User's Guide and Reference*. Consult *MVS/ESA Initialization and Tuning*, as appropriate for your system.

Step 5: Check the Subsystem Name Table

Use the first active NetView SSI for PPI communication. If an SSI for a NetView, other than the one running OPC Automation, is higher in the table, then that SSI will be used for the PPI, disrupting OPC Automation program-to-program communications.

Check the subsystem name table in MVS SYS1.PARMLIB, member IEFSSN*xx*, to verify that the NetView SSI that is used by OPC Automation is first in the list, ahead of all other NetView subsystem names.

Note: Non-NetView subsystems, such as JES2, can precede this entry.

Step 6: Add Libraries to OPC and Recycle

Add your SEVJMOD library and the NetView CNMLINK library containing CNMNETV to the OPC steplib. Alternately, you may add these libraries to LINKLST. You should have already APF authorized these libraries.

A recycle of OPC is required for installing the exit 7 modules. This is EQQUX007 for OPC/ESA, or DRKUX007 for OPC/A. If you are using an existing exit 7, you can combine this exit with OPC Automation-supplied modules. See “Step 3: Integrate Existing Exit 7 with OPC Automation” on page 113 for details.

For OPC/ESA, you must specify the CALL07(YES) parameter in the OPC/ESA initialization parameters.

Other initialization parameters must be specified in the OPC initialization member (EQQPARM for OPC/ESA or DRKPARM for OPC/A) so that OPC will issue some of its messages to the MVS console. These messages will in turn be automated by OPC Automation.

For OPC/ESA, you must specify the following in EQQPARM:

```
ALERTS WTO(ERROROPER,OPCERROR)
```

For OPC/A, you must specify the following in DRKPARM:

```
TYPE(ERROROPER,OPCERROR)
```

In addition, you must edit the OPC-supplied message members for certain messages.

The following messages are automated and may require changes to the TME 10 OPC or OPC/ESA supplied message members in the SEQQMSG0 data set or to the OPCMLIB data set for OPC/A.

TME 10 Message	OPC V2 Member	OPC/ESA Message	OPC/ESA Member	OPC/A Message	OPC/A Member
EQQW065I	EQQW06	EQQW065I	EQQW06	DRKWOO5I	DRKW00
EQQW011I	EQQW01	EQQW011I	EQQW01	DRKW011I	DRKW11
EQQN013I	EQQN01	EQQN013I	EQQN01	CSYN013I	CSYN01
EQQZ086I	EQQZ08	EQQZ086I	EQQZ08	DRKZ006I	DRKZ006
EQQE026I	EQQE02	EQQE026I	EQQE02	CSYE026I	CSYE02
EQQE036I	EQQE03	EQQE036I	EQQE03	CSYE036I	CSYE03
EQQZ128I	EQQZ12				
EQQZ201I	EQQZ20				

Modify these message members to include WTO=YES for the indicated message IDs. Full details for customizing OPC can be found in *TME 10 OPC Customiztion and Tuning*, *OPC/ESA Installation and Customization*, or in *OPC/A Installation and Customization*.

Chapter 15. Merge NetView Related Members

This chapter describes how to build OPC Automation parameter data sets and assemble the code that allows OPC Automation to operate in the NetView environment.

Use the following table to track your completed work:

Table 9. Merging the NetView Related Members Check List

√	**Step**
	1: Add OPC Automation data sets to NetView JCL.
	2: Copy OPC Automation sample members to the target library.
	3: Merge Status Display Facility members.
	4: Merge EVJCFG into the control file.
	5: Merge EVJCMD into DSICMD.
	6: Merge the NetView Message Table.
	7: Merge EVJOPF into DSIOPF.
	8: Merge the NetView profile data set.
	9: Merge EVJDMN into DSIDMN.

Step 1: Add OPC Automation Data Sets to NetView JCL

Add the following libraries to your NetView JCL procedure. Review the NetView JCL procedure to verify that OPC Automation data set block size does not cause a problem in the concatenation. Remember to specify a block size equal to or larger than the largest data set in the concatenation chain in the DCB parameter on the DD statement (DCB=BLKSIZE=...).

DD	**DSN**	**Description**
DSICLD	AOCOPC.V1R4M0.SEVJNCL1	Command Lists
CNMPNL1	AOCOPC.V1R4M0.SEVJNPN1	Panels
DSIMSG	AOCOPC.V1R4M0.SEVJMSG	Communication Messages
STEPLIB[1]	AOCOPC.V1R4M0.SEVJMOD OPCESA.V1R2M1.SEQQLMD0	APF authorized link library containing OPC Automation and OPC/ESA modules

Note:

[1] See the description of STEPLIB in “Add OPC/ESA Data Sets and Allocate EQQMLOG” on page 106.

Note: For performance reasons, these data sets should be high in the concatenation chain.

Next, depending on your environment, select one of the following data sets to install.

- Add OPC/ESA data sets
- Add OPC/A data sets

Add OPC/ESA Data Sets and Allocate EQQMLOG

If you are running OPC/ESA, add JCL for OPC/ESA data sets used by NetView as shown:

```
//*
//* OPC/ESA MISC DATA SETS
//*
//EQQMLOG  DD  DSN=AOCOPC.V1R4M0.AOF01.EQQMLOG,DISP=SHR
//EQQMLIB  DD  DSN=OPCESA.V1R2M0.SEQQMSG0,DISP=SHR
//EQQDUMP  DD  DUMMY
```

EQQMLIB
OPC/ESA data set containing message text. Please review your OPC/ESA procedures to find the correct dataset name for EQQMLIB.

EQQDUMP
OPC/ESA dump data set, used in the event of an abend. (In the example above, a dummy is used.)

EQQMLOG
Run sample job EVJSJ011 to allocate an EQQMLOG data set. There are comments in EVJSJ011 to guide you. A unique log data set must be allocated for each NetView which runs OPC Automation, and these may not be shared. One cylinder of space should be adequate, because this data set is used only for OPC API error messages, and it is reopened each time NetView initializes the interface.

STEPLIB
The OPC/ESA load library SEQQLMD0 must be accessible to the NetView procedure. This can be accomplished by adding this library as STEPLIB to the NetView procedure, or by adding the library to LINKLST. However, if OPC/A and OPC/ESA are installed in the same system, then a STEPLIB for SEQQLMD0 must be used, or else the NetView procedure will not pick up the correct version of CSYYCOM (EQQYCOM) and various commands to OPC/ESA will fail.

Add OPC/A Data Sets and Allocate DRKMLOG

If you are running OPC/A, add JCL for OPC/A data sets used by NetView as shown:

```
//*
//* OPC/A MISC DATA SETS
//*
//DRKMLOG  DD  DSN=AOCOPC.V1R4M0.AOF01.DRKMLOG,DISP=SHR
//DRKMLIB  DD  DSN=OPCA.OPCMLIB,DISP=SHR
//CSYDUMP  DD  DUMMY
```

DRKMLIB
OPC/A data set containing message text. Please review your OPC/A procedure to find the correct dataset name for DRKMLIB.

CSYDUMP

OPC/A dump data set, used in the event of an abend. (In the example above, a dummy is used.)

DRKMLOG

Run sample job EVJSJ011 to allocate an DRKMLOG data set. There are comments in EVJSJ011 to guide you. A unique log data set must be allocated for each NetView which runs OPC Automation, and these may not be shared. One cylinder of space should be adequate, because this data set is used only for OPC API error messages, and it is reopened each time NetView initializes the interface.

Step 2: Copy OPC Automation Sample Members to the Target Library

__ Run sample job EVJSJ010 (in SEVJSAMP) to copy the sample members that may require customization from the sample library to your DSIPARM data set. Tailor the JCL to reflect your operational DSIPARM data set.

Step 3: Merge Status Display Facility Members

__ 1. Copy EVJTREE from your DSIPARM data set into your existing AOFTREE member. EVJTREE contains the tree structure for OPC Automation Status Display Facility panels. See “Tree Structure for Panels” on page 123 for a copy of EVJTREE.

Note: If you are using the %INCLUDE facility for multiple tree structures, you must copy EVJTREE into each structure, as appropriate.

__ 2. Copy EVJPNLS into your existing Status Display Facility panel definition member (AOFPNLS). AOFPNLS contains a list of %INCLUDE statements for all of the Status Display Facility panels used with OPC Automation.

Step 4: Merge EVJCFG into the Control File

__ Copy EVJCFG into your operational SA OS/390 control file. This member contains an %INCLUDE statement that causes the processing of member EVJCFG01 by SA OS/390.

Step 5: Merge EVJCMD into DSICMD

__ Merge EVJCMD into DSICMD before the END statement. EVJCMD contains the command definitions required for OPC Automation. See Appendix E, “Sample OPC Automation Command Synonyms” on page 147 for a copy of EVJCMD. You can also just **%INCLUDE** it in DSICMD.

Step 6: Merge and Update the Automation Table

__ 1. Merge OPCMSG00 into your NetView/SA OS/390 message table.

OPCMSG00 must be merged with the SA OS/390 message table that is loaded early during NetView initialization. The SA OS/390 sample is AOFMSG00. To accomplish this task, insert the statement %INCLUDE OPCMSG00, after the %INCLUDE for AOFMSGSY in member AOFMSG00.

__ 2. Merge OPCMSG01 into your NetView/SA OS/390 message table.

OPCMSG01 must be merged with the SA OS/390 message table that is used during steady-state NetView operation. The SA OS/390 sample is AOFMSG01. To accomplish this task, insert the statement %INCLUDE OPCMSG01, after the %INCLUDE for AOFMSGSY in member AOFMSG01. Please note that OPCMSG01 contains %INCLUDE statements for a number of other members that include:

EVJMCON1 Messages that may be in conflict with other entries in your NetView message table.

EVJMOPCE OPC/ESA messages used by OPC Automation

EVJMOPCA OPC/A messages used by OPC Automation

__ 3. If you are currently using a customized NetView automation table and it contains messages prefixed with EQQ, DRK, or CSY, make sure that these are not in conflict with OPC Automation message table entries in member EVJMOPCE or EVJMOPCA. Document and resolve any conflicts.

__ 4. Additional messages that may have conflicts are highlighted in the OPC Automation message table entries in member EVJMCON1. Review those messages to verify that there are no conflicts. Note that some message IDs have “early out” logic in the automation table, such as MSGID=‘IEF’. Document and resolve any conflicts.

Note: This step requires an understanding of the operation of the NetView automation table. Refer to the *NetView Administration Reference* manual for additional information.

Step 7: Merge EVJOPF into DSIOPF

__ Merge member EVJOPF from your DSIPARM data set into DSIOPF, or use **%INCLUDE** to add it to DSIOPF.
Use EVJOPF to define OPC Automation autotasks.

Step 8: Merge the NetView Profile Data Set

__ This is the profile for the automated operators. To merge this data set, copy EVJPRFAO into your NetView profile data set (DSIPRF).

Step 9: Merge EVJDMN into DSIDMN and Update

The purpose of EVJDMN is to add the EVJNTASK to DSIDMN. To merge this member:

__ 1. Merge EVJDMN from your DSIPARM data set into DSIDMN, or use **%INCLUDE** to add it to DSIDMN.

__ 2. Verify that there are enough VTAM APPL statements in your VTAM definitions to allow all of the operators to log on.

Chapter 16. OPC Automation Initial Customization

This chapter describes the definitions that occur in NetView.

Step 1: Basic OPC Automation Common Control File Definitions

For each NetView domain, set up OPC Automation automation environment according to the following checklist:

__ 1. Verify the defining of the system environment as described in the SA OS/390 base documentation.

Note: Establish a working and tested SA OS/390 before beginning the customization of OPC Automation.

__ 2. Add AUTOOPS entries to SA OS/390 using the SA OS/390 customization dialogs for the OPC Automation operator tasks:

```
AUTOOPS OPCAOPR1,ID=AUTOPCP,MSG=(CSY*,DRK*,CSZ*,EQQ*,EVJ*)
AUTOOPS OPCAOPR2,ID=AUTOPCE,MSG=(CSY*,DRK*,CSZ*,EQQ*,EVJ*)
```

For the automated operator task OPCAOPR1, the operator ID must be AUTOPCP. For the automated operator task OPCAOPR2, you may specify whatever operator ID meets your installation standards.

__ 3. Add subsystem information to the OPC Automation customization dialogs for the OPC/ESA Controller (or the OPC/A PCS), and for the OPC/ESA Tracker (or the OPC/A EMS).

Alternately, you can uncomment the sample entries in EVJCFG01. (It is unlikely, however, that the samples will be adequate to your system's needs without further customization.)

___ 4. Customize the following control file entries:

OPCA PCS Use the worksheet shown in "Step 3: Define the OPC Environment" on page 127, and see "OPCA PCS" on page 75 for information on these parameters.

OPCA DOMAINID Use the worksheet shown in "Step 1: Define the Workstations" on page 125, and see "OPCA DOMAINID" on page 69 for details on completing the domain table.

__ 5. You must define OPCA CODE and OPCACMD entries for each subsystem operation you wish to automate from OPC. (See Chapter 11, "Control File Entries Used by OPC Automation" on page 65.) In addition, the subsystems must be properly defined to OPC Automation before you can automate them. (See "Define Operations on the Workstation" on page 114.)

Step 2: Customizing the Status Display Facility

Notes:

1. Before using these steps, refer to the SA OS/390 base documentation for a complete overview on customizing the Status Display Facility.
2. Instructions that explain how to merge the Status Display Facility members were given in “Step 2: Copy OPC Automation Sample Members to the Target Library” on page 107 and “Step 3: Merge Status Display Facility Members” on page 107.

__ 1. Your completed tree should look similar to the following:

```
1 SY1
  2 JES
  2 RMF
  2 VTAM
  2 TSO
  /* OPC Automation ENTRIES FOLLOW */
  2 OPCERR
  2 BATCH
  2 TSOUSERS
  2 SYSTEM
    3 MESSAGES
    3 IO
      4 TAPES
      4 ONLINE
```

__ 2. Add EVJPNLS to your existing AOFPNLS.

__ 3. If the system name, as defined with the ENVIRON SETUP,SYSNAME= control file entry, is not SY1, edit the following members and change SY1 to your system name in the Status Display Facility status field component name.

```
SY1BATCH    SY1SYS2
SY1CMSGB    SY1SYS2B
SY1CMSGS    SY1SYS2C
SY1MSGS     SY1SYS2X
SY1MSGSB    SY1SYS2Y
SY1MSGSC    SY1SYS22
SY1MSGS2    SY1TAPE
SY1OPCA     SY1TSOA
SY1SYS      SY1TSOU
EVJD0001
```

If you have an automated change tool, you may change the string SY1. to your system name followed by a period.

__ 4. Customize the Status Display Facility main panel to add individual status fields to display the new descriptor types added by OPC Automation (those are listed above: OPCERR, BATCH, TSOUSERS, MESSAGES, TAPES, and ONLINE) or else to add EVJD0001 as a down panel. Chapter 3, “Problems in an OPC-defined application” of the *AOC/MVS OPC Automation Operator's Guide and Scheduler Reference* has examples of what this customization might look like.

For example, to add a ‘O’ to the Status Display Facility SYSTEM panel, so that when you position the cursor underneath it and press PF8 (the SDF

"DOWN" PF Key), panel EVJD0001 is shown; edit SYSTEM and add lines similar to the following:

```
SF(SY1.OPC,07,75,76,N,,EVJED0001)
ST(O)
```

This would position the 'O' in row 7 of the panel in column 75. Change the row, position, and root name (SY1) to fit into your system.

Step 3: Integrate Existing Exit 7 with OPC Automation

OPC Automation supplies EQQUX007/DRKUX007 to detect workstations used for NetView communication. The following modules are used as part of that process:

- DRKUX007
- EQQUX007
- UX007001
- UX007002

EQQUX007/DRKUX007 is the exit driver program. It calls other modules in turn, as if OPC is calling each module directly. The driver searchs for UX007001 through UX007010. UX007001 and UX007002 are supplied with OPC Automation. If you have an existing exit 7, rename your module from EQQUX007 or DRKUX007 to UX007003. The called routines are passed to the same parameters that calls EQQUX007. Note that the parameter CALL07(YES) is necessary for OPC/ESA.

If you wish to add additional exit 7 modules, then use the next available name, such as UX007004. This makes it easier to integrate exits supplied by various products. Also, since modules are loaded dynamically by the exit driver on each invocation, you may add, delete, or modify an exit module without recycling OPC.

Step 4: Initializing the OPC Automation Status File

To create a status file from NetView, enter the command:

```
EVJESPIN CMD=INIT
```

Do not use this command until you have created the OPCA CODE entries that you intend to use for testing. You can enter it manually, or it will be issued automatically every time OPC Automation is reinitialized.

OPC Automation Test Scenario

Define Operations on the Workstation

For a test scenario, assume that you have a subsystem that can shut down and restart at will. The example uses RMF. If you use another subsystem, then edit the control file entries as necessary.

Additional details and typical OPC definitions are shown in Chapter 2 of *AOC/MVS OPC Automation Operator's Guide and Scheduler Reference*.

For details of OPC use, refer to the *OPC/ESA User's Guide*.

In OPC, define a test application that has an operation step on the NVxx workstation and operation text of START or STOP. This text matches the OPCACMD entries in the control file. For example, the text START RMF, defined as an operation on NV06, sends the request to SA OS/390 for processing as the SETSTATE command coded in the RMF OPCACMD entry:

```
RMF OPCACMD,CMD=(START,,'SETSTATE RMF,DOWN,START=YES')
RMF OPCACMD,CMD=(STOP,,'SHUTSYS RMF,RESTART=CTL,VERIFY=NO,SCOPE=ONLY')
```

When the NV06 operation with text STOP RMF becomes ready, then AOFS6 will execute the SHUTSYS command as specified in the control file entry. The RMF OPCA CODE entries for START and STOP are used to specify timer names and times in minutes. In EVJCFG01, the timer allows three minutes for RMF to get to the UP status and one minute for the CTLDOWN status. If the subsystem is not in the expected status when the timer CLIST is executed, the timer module will post an error status for the operation.

Test NetView Commands

On the automation NetView, try the following commands:

OPCA	You should see the main tutorial panel for OPC Automation.
EVJESPIN CMD=INIT	You should receive a message indicating successful completion.
OPCAQRY	You should see statuses of requests between OPC and NetView. Use the browse option to see detail.
OPCACMD	Fill in the screen with an OPC application, you should see data for that application.

Problem Determination Suggestions

The following suggestions are offered for problem determination and resolution. Review NetView and OPC documentation for specific problem determination and resolution assistance.

- Is the EVJTOPPI task active?
- Are there error messages in the NetView log?
- Determine status using OPCQRY.
- EVJTRACE ON may help with resolving problems.

Part 5. Appendixes

Appendix A. Status Display Facility Enhancements

Coding Reference

CLISTs Used to Implement the Supplied Extensions

CLIST	Alias	Function
EVJEAB00	DFTAPO	Finds online tape drives
EVJEAB01	DFBTCH	Process batch job start/end
EVJEAB02	DFTAPM	Finds online tape drives
EVJEAB03	DFTSOU	Processes TSO logon/logoff
EVJEAB04	DFTSOR	Displays TSO users for restart purposes
EVJEAB05	DFTAPK	Tape check; sees if tape was mounted
EVJEAB06	DFCRIT	Critical message processor
EVJEAB07	DFUPDT	Status Display Facility update
EVJEAB10	DFDELT	Status Display Facility delete

DFUPDT Status Display Facility update CLIST.

Use this routine to build your own displays. See the syntax and discussion that follows “DFUPDT” on page 119.

DFTAPK Tape check. Determines if tape mounts are resolved.

DFTAPM Tape message processor. Intercepts the following messages:

- **IEF233A** Volume mount message
- **IEF234E** Volume dismount message
- **IEF251I** Job cancelled message
- **IEC501A** Volume mount message
- **IEC502E** Volume dismount message
- **IEC701D** Volume to be labelled request
- **IEC705I** Tape mounted message
- **TMS001** Volume mount message (OEM product)
- **TMS002** Volume dismount message (OEM product)

DFTAPO Tape online. Finds all online tape units for panel display.

DFDELT Status Display Facility delete used by the various CLISTs.

DFCRIT Critical message processor.

Call DFCRIT for each message that you wish to appear in the critical message facility. The parameters are the text of the message that are added to the Status Display Facility. See the syntax and discussion that follows “DFCRIT” on page 121.

DFTSOU Captures TSO logons and logoffs. The following messages are processed:

IEF125I TSO user logged on

IEF126I TSO user logged off

IEF450I TSO user abend

DFTSOR TSO Refresh. See all TSO users logged on the system to build display at NetView initialization or Status Display Facility recycle.

DFBTCH Captures batch job start and stop. The following messages are processed:

$HASP373 Job started

IEF404I Job ended

IEF450I Abend

IEF453I Job failed

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

```
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*.

DFCOPY

Purpose

DFCOPY synchronizes SDF components between the target and focal point systems.

Format

```
DFCOPY component,domain
```

Parameters

component

The SDF component to be copied. It can be of the form *sysname.component*, but if *sysname* is not specified, then it will be set to the running system.

domain

The destination domain. Entries from the running system will be sent to the SDF on that system.

Usage Notes

Both parameters are required.

Examples

If you are on target system CNM0T (sysname = TGT) and wish to send all the OPCERR entries to CNM0F, your focal point:

```
DFCOPY OPCERR,CNM0F
```

Also, the command:

```
DFCOPY TGT.OPCERR,CNM0F
```

is equivalent. When this is executed, any OPCERR entries will be copied. If there were no entries on CNM0T, and CNM0F had residual data, then CNM0F's entries will be deleted.

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

```
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 a 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.

Examples

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' & TEXT=MESSAGE
  THEN EXEC(CMD('DFCRIT TYPE=U,'MESSAGE) ROUTE(ALL *));
```

EVJEAB11

Purpose

EVJEAB11 is a command used in certain Status Display Facility panels to synchronize data in a distributed environment. It can replace the standard SDFDEL command to delete items by positioning the cursor under the item, with the added function of deleting them from other systems as well. It is typically defined in a panel definition as:

```
PFK9('EVJEAB11 &SNODE,&ROOT.&COMPAPPL,&RV,&DATA')
```

Format

EVJEAB11 *sendernode,component,ref_value,data*

Parameters

The input parameters for EVJEAB11 are those for AOC/MVS Status Display Facility programming of PF keys for use with the detail screen. For more information, consult “Customizing the Status Display Facility” section of the *System Automation for OS/390 Customization* manual. The following parameters are all required for EVJEAB11 to function correctly and must be coded as shown above.

&SNODE
Sender node — the NetView from which this message originated

&ROOT
Root — the SDF root ot system name of the originating NetView

&COMPAPPL
Component — the SDF descriptor under which this message is saved

&RV
Reference Value — the SDF reference value of this entry

&DATA
Data — the actual messsage text

Usage Notes

EVJEAB11 deletes the item under the cursor, and then attempts to delete from the originating system: If the entry came from this system, and this system is a target system in a distribute environment, then the entry at the focal point will be deleted. If the entry came from a target system, then it will be deleted at the target system.

Tree Structure for Panels

EVJTREE

EVJTREE, a sample file that contains the tree structure for OPC Automation Status Display Facility panels, is copied from your DSIPARM data set. It is shipped with OPC Automation.

```
/*********************************************************************/
/*      COPYRIGHT= 5685-151                                          */
/*                 CONTAINS RESTRICTED MATERIALS OF IBM              */
/*                 (C) COPYRIGHT IBM CORP. 1990, 1995                */
/*                 LICENSED MATERIALS - PROPERTY OF IBM              */
/*                 REFER TO COPYRIGHT INSTRUCTIONS                   */
/*                 FORM NUMBER G120-2083.                            */
/*                                                                   */
/*********************************************************************/
/* APAR#                                                             */
/* ----------------------------------------------------------------- */
/* $01=OW07970 09/30/94 JS EVJTREE MEMBER DOES NOT CONTAIN AN ENTRY  */
/*                         TO ALLOW EVJD0001 TO BE CALLED AS DOWN    */
/*                         PANEL FROM AOC SYSTEM PANEL IN SDF        */
/*                                                                   */
/*                                                                   */
/*********************************************************************/
/* SAMPLE EHKTREE ADDITIONS FOR OPC FEATURE          */
/* MERGE THIS WITH YOUR EXISTING EHKTREE ENTRY       */
  2 OPC
    3 OPCERR
    3 BATCH
    3 TSOUSERS
    3 MESSAGES
    3 IO
      4 TAPES
      4 ONLINE
```

Appendix B. OPC Automation Worksheets

This appendix contains worksheets that help you define:

- Workstations
- Operations
- OPC environment

Step 1: Define the Workstations

Use this worksheet to define the workstations. A workstation name is needed to correspond with each automation NetView domain ID in use. You must name the workstations NV*nn*, where *nn* is two digits. The suggested standard is the last two digits of the domain ID. These entries are used to build the control file OPCA DOMAINID entry.

Domain ID _____ Workstation Name _____
Domain ID _____ Workstation Name _____
Domain ID _____ Workstation Name _____
Domain ID _____ Workstation Name _____
Domain ID _____ Workstation Name _____
Domain ID _____ Workstation Name _____
Domain ID _____ Workstation Name _____
Domain ID _____ Workstation Name _____

Step 2: Define the Operations

Use these worksheets to define the operations. The operations define what OPC Automation does with each subsystem. Example operations of STOP and START are included in EVJCFG01.

Subsystem _______________ Operation name ____________ Command to be used: ____ Expected Status of subsystem: ____ Maximum Time to allow for operation to complete: _________________________ Unique Timer Name to be used for this operation: _________________________
Subsystem _______________ Operation name ____________ Command to be used: ____ Expected Status of subsystem: ____ Maximum Time to allow for operation to complete: _________________________ Unique Timer Name to be used for this operation: _________________________
Subsystem _______________ Operation name ____________ Command to be used: ____ Expected Status of subsystem: ____ Maximum Time to allow for operation to complete: _________________________ Unique Timer Name to be used for this operation: _________________________
Subsystem _______________ Operation name ____________ Command to be used: ____ Expected Status of subsystem: ____ Maximum Time to allow for operation to complete: _________________________ Unique Timer Name to be used for this operation: _________________________

Step 3: Define the OPC Environment

Use this worksheet to define the OPC environment. The operations define what OPC Automation does with each subsystem. Example operations of STOP and START are included in EVJCFG01.

OPC Subsystem name: _____

OPC started task for PCS/Controller: _____

OPC started task for EMS/Tracker: _____

NetView automation domain ID for system running PCS/Controller: _____

Appendix C. Sample OPC Automation Control File

EVJCFG01, a sample control file, is shipped with OPC Automation. “EVJCFG01” shows this file.

The OPC Automation control file has the following entries:

- AUTOOPS entries that support OPC Automation
- OPC workstation definitions
- OPC operation definitions
- Status Display Facility entries for OPC Automation

EVJCFG01

```
*********************************************************************
*                                                                   *
*       COPYRIGHT= 5685-151                                         *
*                  CONTAINS RESTRICTED MATERIALS OF IBM             *
*                  (C) COPYRIGHT IBM CORP. 1990, 1998            @02C*
*                  LICENSED MATERIALS - PROPERTY OF IBM             *
*                  REFER TO COPYRIGHT INSTRUCTIONS                  *
*                  FORM NUMBER G120-2083.                           *
*                                                                   *
*                                                                   *
*    PROGRAM NUMBER:   5685-151                                     *
*    DESCRIPTION: SAMPLE DSIPARM - CONTROL FILE FOR AOC OPC         *
*                                                                   *
*    APAR#                                                          *
* ----------------------------------------------------------------- *
* $02=OW35607,V1R4,08DEC98,APC(IG): OPC V2 Exploitation.            *
*                                   LI06 OPC Controller on the move *
* $01=OW11690 03/03/95 JS AOC OPC INITIALIZATION PROBLEMS AFTER AOC *
*                         OW03453 AND OW09496                       *
*                                                                   *
*                         Initial version for AOC OPC               *
*********************************************************************
*  It is recommended that the AUTOOPS entries below be added via    *
*  the AOC dialogs.  If desired, the ones below may be uncommented  *
*  instead.  All messages should be assigned to only one autotask @01C*
*********************************************************************
*AUTOOPS  OPCAOPR1,
*    ID=AUTOPCP,
*    MSG=(CSY*,DRK*,CSZ*,EQQ*,EVJ*)
*AUTOOPS  OPCAOPR2,
*    ID=AUTOPCE
*********************************************************************
*                                                                   *
* OPCA PCS,                        Required on all NetViews with an *
*                                  OPC Controller or OPC Stand-by   *
*                                  Controller.                      *
*                                                                   *
*   DOMAIN = domainid | SYSPLEX,   Specify the NetView domain where *
*                                  the OPC Controller runs or SYSPLEX *
*                                  to indicate that the OPC         *
*                                  Controller may be on any of the  *
*                                  systems defined in the LOCAL     *
*                                  sysplex.                         *
*                                                                   *
*   SUBSYS = xxxx                  Specify the four character MVS   *
*                                  task name for the OPC Controller.  *
```

```
*                                  This name must also be used for    *
*                                  the AOC Subsystem name of the      *
*                                  application                        *
*                                                                     *
***********************************************************************
OPCA      PCS,
    DOMAIN=AOF01,
    SUBSYS=OPCC
*
***********************************************************************
*                                                                     *
*       ENVIRON  OPCAO,                                               *
*                  REQSTAT = NO | YES,       Check subsys status      *
*                                 ---        on OPC START/STOP/RECYCLE *
*                  MSGKEEP = hh:mm | PERM,   Length of time to keep   *
*                                   ----     Critical Messages in SDF. *
*                  OPRESET = hh:mm | NEVER   Length of time NetView   *
*                                   -----    can be down and UNTV reset*
*                                                                     *
***********************************************************************
*
ENVIRON OPCAO,
    REQSTAT=YES,
    MSGKEEP=PERM,
    OPRESET=00:30
*
***********************************************************************
*   DOMAINID TABLE - RELATE OPC WORKSTATION TO NETVIEW DOMAIN.         *
***********************************************************************
OPCA      DOMAINID,
    CODE=(NV05,,,AOF05),
    CODE=(NV00,,,AOF10),
    CODE=(NV11,,,AOF01),
    CODE=(NVSP,,,SYSPLEX)
***********************************************************************
*                                                                     *
*  ENTRIES REQUIRED FOR INTERFACE OPERATIONS:                         *
*                                                                     *
*   OPCA                                                              *
*   OPCACMD                                                           *
*   OPCAPARM                                                          *
*                                                                     *
*  NOTE: With AOC V1R3 all OPCA, OPCAPARM, and OPCACMD entries for    *
*        the same subsystem (for example RMF below) MUST be grouped   *
*        into a single entry.  If multiple entries are used, only     *
*        the last one will be found and the other operations will fail.*
*                                                                     *
***********************************************************************
*                                                                     *
*  ENTRY OPCA,CODE=(REQUEST,PARM1,PARM2,'EXPSTATUS,TIMRINT,TIMERID')  *
*                                                                     *
***********************************************************************
*                                                                     *
*                                                                     *
RMF       OPCA,
    CODE=(START,,,'UP,3,RMFUTMER'),
    CODE=(STOP,,,'CTLDOWN,2,RMFDTMER'),
    CODE=(RECYCLE,,,'UP,5,RMFRTMER')
*
***********************************************************************
*ENTRY OPCAPARM,CODE=(REQUEST,PARM1,PARM2,'PARM1VALUE,PARM2VALUE,TIMER')
*
*These entries are now optional.  There is no need to code an OPCAPARM
*statement unless you need add an additional parameter beyond the
```

```
*operation type.                                                        *
************************************************************************
*
*RMF       OPCAPARM,
*    CODE=(START,,,',,'),
*    CODE=(STOP,,,',,')
*
************************************************************************
*ENTRY OPCACMD,CODE=(REQUEST,PARM1,COMMAND')
*
************************************************************************
*                                                                       *
RMF   OPCACMD,
    CMD=(START,,'SETSTATE RMF,RESTART,START=YES'),
    CMD=(RECYCLE,,'EVJESHUT RMF ONLY'),
    CMD=(STOP,,'SHUTSYS RMF,VERIFY=NO,RESTART=CTL,SCOPE=ONLY')
*                                                                       *
JOBX  OPCACMD,
    CMD=(UXCINITS,,'MVS $TI20-30,C=P')
*                                                                       *
************************************************************************
* KEY INTERFACE CLISTS                                                  *
*  It is recommended that the RESIDENT entries below be added via       *
*  the AOC dialogs.  If desired, the ones below may be uncommented      *
*  instead.                                                      @01C*
************************************************************************
* RESIDENT    EVJESPIN
* RESIDENT    EVJESPVY
* RESIDENT    EVJESPRQ
* RESIDENT    EVJESPSC
* RESIDENT    EVJESPTE
* RESIDENT    EVJESPCP
* RESIDENT    EVJESHUT
************************************************************************
* UTILITY CLISTS                                                        *
************************************************************************
* RESIDENT    EVJTRACE
* RESIDENT    EVJEMSG
* RESIDENT    EVJESUSF
************************************************************************
* OPERATOR CLISTS                                                       *
************************************************************************
* RESIDENT    EVJEAC00
* RESIDENT    EVJEAC03
* RESIDENT    EVJECGAA
* RESIDENT    EVJECGA1
************************************************************************
* MESSAGE PROCESSING CLISTS                                             *
************************************************************************
* RESIDENT    EVJEAB00
* RESIDENT    EVJEAB01
* RESIDENT    EVJEAB02
* RESIDENT    EVJEAB03
* RESIDENT    EVJEAB04
* RESIDENT    EVJEAB05
* RESIDENT    EVJEAB06
* RESIDENT    EVJEAB07
* RESIDENT    EVJEAB10
*
************************************************************************
* DISPLAY FACILITY ENTRIES                                              *
* After OW10863 / UW15395 is applied, these entries may be specified    *
* in the AOC dialogs and the ones below removed.                 @01A*
************************************************************************
```

```
*
* TAPES
*
SDF  RMOUNT,
     PR=430,
     CLEAR=Y,
     CO=P,
     HL=R
SDF  BMOUNT,
     PR=429,
     CLEAR=Y,
     CO=R,
     HL=R
SDF  SMOUNT,
     CLEAR=(Y,RV),
     REQ=NOADD
*
* TAPE UNITS
*
SDF  ONLINE,
     PR=550,
     CLEAR=Y
SDF  OFFLINE,
     CLEAR=(Y,RV),
     REQ=NOADD
*
* TSOUSERS
*
SDF  TSOLOGON,
     PR=550,
     CLEAR=Y
SDF  TSOLOGFF,
     CLEAR=(Y,RV),
     REQ=NOADD
*
* BATCH
*
SDF  BTCHBGN,
     PR=550,
     CLEAR=Y
SDF  BTCHEND,
     CLEAR=(Y,RV),
     REQ=NOADD
*
* OUTSTANDING OPC/A ERRORS
*
SDF  OPCERR,
     PR=330,
     CLEAR=Y,
     CO=Y,
     HL=N
SDF  OPCERRR,
     CLEAR=Y,
     REQ=NOADD,
     PR=330
*
* FOR CTLDOWN STATUS    --  PROVIDED BY AOC BASE             * @01D*
*
*SDF  CTLDOWN,
*     PR=250,
*     HL=R,
*     CLEAR=(Y,RV)
*********************************************************************
*                                                                   *
```

```
*   SAMPLE CONTROL FILE ENTRIES FOR CRITICAL MESSAGES             *
*                                                                 *
*******************************************************************
SYSTEM CRITMSGS,
       CODE=(*,,,SAVE)
SDF    CRITMSG,
       PR=500,
       CO=G
SDF    CRITMSGA,
       PR=500,
       CO=R
SDF    CRITMSGE,
       PR=501,
       CO=Y
SDF    CRITMSGW,
       PR=502,
       CO=T
SDF    CRITMSGI,
       PR=503,
       CO=G
*******************************************************************
*                                                                 *
* OPC SUBSYSTEM INFORMATION                                       *
*                                                                 *
*  It is recommended that all subsystem entries below be added via *
*  the AOC dialogs.  If desired, the ones below may be uncommented *
*  instead, but not all operands are coded.                       *
*******************************************************************************
* OPC/ESA CONTROLLER and STANDBY CONTROLLER                                    *
* The OPC Controller subsystem name and job name must be the same.             *
* All Stand-by controllers on a SYSPLEX must have the same subsystem           *
* name and job name as the primary OPC controller.                             *
*******************************************************************************
*SUBSYSTEM  OPCC,
*    JOB=OPCC,
*    DESC='PRODUCTION CONTROL SYSTEM ',
*    RESTARTOPT=ALWAYS,
*    PARENT=OPCT,
*    SHUTDLY=00:01
*** SUBSYSTEM THRESHOLDS ***
*THRESHOLDS OPCC,
*    CRIT=(03,00:10),
*    FREQ=(02,00:10),
*    INFR=(01,00:10)
*** SUBSYSTEM AUTOMATION FLAGS ***
*AUTOMATION FLAG ENTRIES GO HERE
*** SUBSYSTEM SHUTDOWN PROCESSING ***
*OPCC  SHUTINIT,
*    CMD=(,,'MVS $DMR0,''OPC CONTROLLER SHUTTING DOWN''')
*OPCC  SHUTNORM,
*    CMD=(PASS1,,'MVS P OPCC')
*************************************************************** @02A
* On a restart of SA OS/390 where an OPC controller might run we    @02C
* should:                                                           @02A
* 1) Ensure that it's not in STANDBY mode  (F OPCC,STATUS)          @02A
* 2) Get all UNTV errs and retry them if not out of time (EVJEAC01) @02C
*************************************************************** @02A
*                                                                 @02A
*OPCC  ACORESTART,
*    CMD=(,,'MVS F &SUBSJOB,STATUS'),
*    CMD=(OPCAOPR2,,'EVJEAC01')
*
*************************************************************** @02A
* When an OPC Controller becomes active we should:                  @02C
```

```
* 1) Get all UNTV errs and retry them if not out of time (EVJEAC01) @02A
* 2) Retry for operations in error S998 or S999      (EVJEAC02)     @02A
****************************************************************** @02A
*
*OPCC  UP,
*    CMD='EVJEAC01',
*    CMD='EVJEAC02'
************************************************************************
*** OPC/ESA TRACKER ***
*** SUBSYSTEM NAME ***
*SUBSYSTEM  OPCT,
*    JOB=OPCT,
*    DESC='EVENT MANAGER SYSTEM ',
*    RESTARTOPT=ALWAYS,
*    PARENT=JES,
*    SHUTDLY=00:01
*** SUBSYSTEM THRESHOLDS ***
*THRESHOLDS  OPCT,
*    CRIT=(03,00:10),
*    FREQ=(02,00:10),
*    INFR=(01,00:10)
************************************************************************
*    SUBSYSTEM SHUTDOWN PROCESSING                                     *
************************************************************************
*OPCT  SHUTINIT,
*    CMD=(,,'MVS $DMR0,''OPC TRACKER SHUTTING DOWN''')
*OPCT  SHUTNORM,
*    CMD=(PASS1,,'MVS P OPCT')
************************************************************************
*                                                                      *
*     END OF EVJCFG01 MEMBER                                           *
*                                                                      *
************************************************************************
```

Appendix D. Sample OPC Automation Message Table

This appendix shows the following tables:

OPCMSG00 Initial table to be merged with the SA OS/390 table AOFMSG00.

OPCMSG01 Production table to be merged with the SA OS/390 table AOFMSG01. This table includes:

- EVJMCON1 — Entries in conflict with other entries from SA OS/390.
- EVJMOPCE — OPC/ESA messages used by OPC Automation.
- EVJMOPCA — OPC/A messages used by OPC Automation.

OPCMSG00

```
************************************************************************ 00000102
*                                                                      * 00000202
*       COPYRIGHT= 5685-151                                            * 00000302
*                  CONTAINS RESTRICTED MATERIALS OF IBM                * 00000402
*                  (C) COPYRIGHT IBM CORP. 1995                        * 00000502
*                  LICENSED MATERIALS - PROPERTY OF IBM                * 00000602
*                  REFER TO COPYRIGHT INSTRUCTIONS                     * 00000702
*                  FORM NUMBER G120-2083.                              * 00000802
*                                                                      * 00000902
*    DESCRIPTION: SAMPLE DSIPARM - MSG AUTOMATION TABLE FOR AOC OPC    * 00001002
*    EVJTBL00 CHANGE ACTIVITY:                                         * 00001102
*    APAR#                                                             * 00001202
* ------------------------------------------------------------------- * 00001302
*                                                                        00001402
************************************************************************ 00001502
* FOCAL POINT HAS SWITCHED, FORWARD THE DATA                          * 00001602
************************************************************************ 00001702
 IF MSGID='AOF660I' & TEXT=MESSAGE                                      00001802
   THEN EXEC(CMD('EVJEAB14 'MESSAGE) ROUTE(ONE *))                      00001902
   CONTINUE(Y);                                                         00002002
*                                                                       00003002
************************************************************************ 00004002
* SYSTEM HAS RECONNECTED, SEE IF THERE ARE PENDING OPERATIONS.        * 00005002
************************************************************************ 00006002
*                                                                       00007002
 IF MSGID='AOF661I' & TOKEN(7) = SVDOM  & DOMAINID = %AOFDOM%           00008002
   THEN EXEC(CMD('EVJEAC02 'SVDOM) ROUTE(ALL *))                        00009002
   CONTINUE(Y);                                                         00009102
*                                                                       00009202
```

OPCMSG01

```
************************************************************************ 00010004
*                                                                      * 00020004
*       COPYRIGHT= 5685-151                                            * 00030004
*                  CONTAINS RESTRICTED MATERIALS OF IBM                * 00040004
*                  (C) COPYRIGHT IBM CORP. 1995                        * 00050004
*                  LICENSED MATERIALS - PROPERTY OF IBM                * 00060004
*                  REFER TO COPYRIGHT INSTRUCTIONS                     * 00070004
*                  FORM NUMBER G120-2083.                              * 00080004
*                                                                      * 00090004
*    DESCRIPTION: SAMPLE DSIPARM - MSG AUTOMATION TABLE FOR AOC OPC    * 00100004
*    EVJTBL01 CHANGE ACTIVITY:                                         * 00110004
*    APAR#                                                             * 00120004
* -------------------------------------------------------------------- * 00130004
*                                                                        00140004
************************************************************************ 00150004
* INCLUDE FOR CONFLICT MESSAGES                                       * 00160004
************************************************************************ 00170004
*                                                                        00180004
  %INCLUDE EVJMCON1                                                     00190004
*                                                                        00200004
************************************************************************ 00210004
* INCLUDE FOR OPC/ESA MESSAGES                                        * 00220004
************************************************************************ 00230004
*                                                                        00240004
  %INCLUDE EVJMOPCE                                                     00250004
*                                                                        00260004
*                                                                        00270004
************************************************************************ 00280004
* OPC/A USERS MAY COMMENT OUT THE ABOVE INCLUDE FOR OPC/ESA MESSAGES  * 00290004
* AND REPLACE IT WITH THE ONE BELOW FOR OPC/A MESSAGES                * 00300004
************************************************************************ 00310004
*                                                                        00320004
* %INCLUDE EVJMOPCA                                                     00330004
*                                                                        00340004
```

EVJMCON1

```
************************************************************************ 00000100
************************************************************************ 00000100
*                                                                      * 00000200
*       COPYRIGHT= 5685-151                                            * 00000300
*                  CONTAINS RESTRICTED MATERIALS OF IBM                * 00000400
*                  (C) COPYRIGHT IBM CORP. 1995                        * 00000500
*                  LICENSED MATERIALS - PROPERTY OF IBM                * 00000600
*                  REFER TO COPYRIGHT INSTRUCTIONS                     * 00000700
*                  FORM NUMBER G120-2083.                              * 00000800
*                                                                      * 00000900
*    DESCRIPTION: SAMPLE DSIPARM - MSG AUTOMATION TABLE FOR AOC OPC    * 00001000
*    EVJMCON1 CHANGE ACTIVITY:                                         * 00001100
*    APAR#                                                             * 00001200
* ------------------------------------------------------------------- * 00001300
* $01=OW11690 03/06/95 JS AOC OPC INITIALIZATION PROBLEMS AFTER AOC   *
*                         OW03453 AND OW09496                          *
*                                                                      *
************************************************************************ 00007500
************************************************************************ 00007200
*                                                                      * 00007300
*   NOTICE TO OPC AUTOMATION USERS:                                    * 00007400
*                                                                      * 00007500
*   YOU SHOULD EXAMINE YOUR EXISTING MESSAGE TABLE AND MAKE CHANGES    * 00007600
*   TO IT BASED UPON THIS SAMPLE, ADDING THESE SPECIFIC ENTRIES        * 00007700
*   WHERE THEY BEST FIT.  IT IS NOT ENOUGH TO MERELY ADD THIS TABLE    * 00007800
*   VIA THE %INCLUDE FUNCTION.                                         * 00007900
*                                                                      * 00008000
************************************************************************ 00008400
*                                                                        00008500
************************************************************************ 00009100
* FOCAL POINT HAS SWITCHED, FORWARD THE DATA                  @01C  * 00009200
************************************************************************ 00009300
 IF MSGID='AOF660I' & TEXT=MESSAGE & HDRMTYPE = 'U'                      00009400
   THEN EXEC(CMD('EVJEAB14 'MESSAGE) ROUTE(ONE *))                       00009500
        CONTINUE(Y);
*                                                                        00009600
************************************************************************ 00009900
* SYSTEM HAS RECONNECTED, SEE IF THERE ARE PENDING OPERATIONS.  @01C  * 00010000
************************************************************************ 00010100
*                                                                        00010400
 IF MSGID='AOF661I' & TOKEN(7) = SVDOM  & DOMAINID = %AOFDOM%           00010500
    & HDRMTYPE = 'U'                                                    00010500
   THEN EXEC(CMD('EVJEAC02 'SVDOM) ROUTE(ALL *))                        00010600
        CONTINUE(Y);
*                                                                        00010700
************************************************************************ 00010800
* AOF540I AUTOMATION INIT COMPLETED - NOW ADD OPC FEATURES TO SDF     * 00010900
* CHANGE FROM AOF532I TO AOF540I                                @01C * 00010900
************************************************************************ 00011000
 IF MSGID='AOF540I' & DOMAINID = %AOFDOM% & HDRMTYPE = 'U'               00011100
   THEN EXEC(CMD('DFTAPO ' ) ROUTE(ALL *))                              00011200
        EXEC(CMD('EVJEAB08' ) ROUTE(ALL *))                             00011300
        EXEC(CMD('DFTSOR ' ) ROUTE(ALL *))                              00011400
```

```
       CONTINUE(Y);
*                                                                     00011500
********************************************************************** 00011600
* AOF043I DDF TASK ACTIVE - RESYNCH - ADD TO EXISITING ENTRY     @01C * 00011700
********************************************************************** 00012000
*                                                                     00014400
 IF MSGID='AOF043I' & TOKEN(2)='AOFTDDF' & HDRMTYPE = 'U'             00014500
   THEN EXEC(CMD('DFTAPO ' ) ROUTE(ALL *))                           00014700
        EXEC(CMD('DFTSOR ' ) ROUTE(ALL *))                           00014800
        EXEC(CMD('EVJEAB08' ) ROUTE(ALL *))                          00014900
        NETLOG(Y 2 +ACOOPER);                                        00015000
*                                                                     00015100
********************************************************************** 00015200
*                                                                    * 00015300
* SAMPLE MESSAGE TABLE ENTRIES FOR TAPE MOUNT SDF DISPLAY            * 00015400
*                                                                    * 00015500
*                                                                    * 00015600
* REMOVE THESE ENTRIES IF YOU DO NOT WISH TO HAVE TAPE MOUNTS SHOWN  * 00015700
* IN YOUR SDF DISPLAY.                                               * 00015800
*                                                                    * 00015900
* NOTE: IF YOU WISH TO USE THE TAPES ATTENDED FEATURE OF             * 00016000
* AOC/MVS, THEN YOU MUST HAVE EACH OF THE FOLLOWING MESSAGES CALL    * 00016100
* AOFRSA0A BEFORE DFTAPM AS THE FOLLOWING EXAMPLE SHOWS:             * 00016200
*                                                                    * 00016400
*IF MSGID='IEF233A' & TEXT=MESSAGE                                   * 00017100
*  THEN EXEC(CMD('AOFRSA0A 'MESSAGE) ROUTE(ALL *))                  * 00017200
*       EXEC(CMD('DFTAPM 'MESSAGE) ROUTE(ALL *));                   * 00017300
*                                                                    * 00017400
*  PLEASE TAILOR THE FOLLOWING ENTRIES APPROPRIATELY                 * 00017500
*                                                                    * 00017600
********************************************************************** 00017700
 IF MSGID='IEF233A' & TEXT=MESSAGE                                   00017800
   THEN EXEC(CMD('DFTAPM 'MESSAGE) ROUTE(ALL *));                    00017900
*                                                                     00018000
 IF MSGID='IEF233D' & TEXT=MESSAGE                                   00018100
   THEN EXEC(CMD('DFTAPM 'MESSAGE) ROUTE(ALL *));                    00018200
*                                                                     00018300
 IF MSGID='IAT5210' & TEXT=MESSAGE                                   00018400
   THEN EXEC(CMD('DFTAPM 'MESSAGE) ROUTE(ALL *));                    00018500
*                                                                     00018600
 IF MSGID='TMS001' & TEXT=MESSAGE                                    00018700
   THEN EXEC(CMD('DFTAPM 'MESSAGE) ROUTE(ALL *));                    00018800
*                                                                     00018900
 IF MSGID='TMS002' & TEXT=MESSAGE                                    00019000
   THEN EXEC(CMD('DFTAPM 'MESSAGE) ROUTE(ALL *));                    00019100
*                                                                     00019300
 IF (MSGID='IEC705I' | MSGID='IEC501A' | MSGID = 'IEC101A'           00019400
   | MSGID='IEC502E' | MSGID='IEF234E') & TEXT=MESSAGE               00019500
   THEN EXEC(CMD('DFTAPM 'MESSAGE) ROUTE(ALL *));                    00019600
********************************************************************** 00015200
*                                                                    * 00015300
* SAMPLE MESSAGE TABLE ENTRIES FOR TAPE MOUNT SDF DISPLAY            * 00015400
*                                                                    * 00015600
* REMOVE THESE ENTRIES IF YOU DO NOT WISH TO HAVE TAPE MOUNTS SHOWN  * 00015700
* IN YOUR SDF DISPLAY.                                               * 00015800
*                                                                    * 00015900
* NOTE: IF YOU WISH TO USE THE TAPES ATTENDED FEATURE OF             * 00016000
* AOC/MVS, THEN YOU MUST HAVE EACH OF THE FOLLOWING MESSAGES CALL    * 00016100
```

```
* AOFRSORP BEFORE DFTAPM AS THE FOLLOWING EXAMPLE SHOWS:              * 00016200
*                                                                    * 00016400
*IF MSGID='IEC701D' & TEXT=MESSAGE                                   * 00017100
*  THEN EXEC(CMD('AOFRSORP 'MESSAGE) ROUTE(ALL *))                   * 00020600
*       EXEC(CMD('DFTAPM 'MESSAGE) ROUTE(ALL *));                    * 00017300
*                                                                    * 00017400
*  PLEASE TAILOR THE FOLLOWING ENTRIES APPROPRIATELY                 * 00017500
*                                                                    * 00017600
********************************************************************** 00017700
*                                                                      00019700
* IEC701D TAPE VOLUME TO BE LABELED                                  * 00020300
*                                                                      00020400
 IF MSGID='IEC701D' & TEXT=MESSAGE                                     00020500
   THEN EXEC(CMD('AOFRSORP 'MESSAGE) ROUTE(ALL *))                     00020600
        EXEC(CMD('DFTAPM 'MESSAGE) ROUTE(ALL *));                      00020700
*                                                                      00020800
* DEVICE VARIED OFFLINE/ONLINE, SO REFRESH TAPE ONLINE                 00019800
*   THIS ENTRY SHOULD NOT CALL ANY BASE PRODUCT ROUTINES               00019800
*                                                                      00019900
 IF (MSGID='IEF281I' | MSGID='IEE302I') & TEXT=MESSAGE                 00020000
   THEN EXEC(CMD('DFTAPO 'MESSAGE) ROUTE(ALL *));                     00020100
*                                                                      00020200
********************************************************************** 00020900
*                                                                    * 00021000
* SAMPLE MESSAGE TABLE ENTRIES FOR BATCH JOB SDF DISPLAY             * 00021100
*                                                                    * 00021200
*                                                                    * 00021300
* REMOVE THESE ENTRIES IF YOU DO NOT WISH TO HAVE BATCH JOBS SHOWN   * 00021400
* IN YOUR SDF DISPLAY.  JES3 USERS SHOULD REMOVE THE IEF ENTRIES     * 00021500
* AND UNCOMMENT THE HASP ENTRIES.   ALL USERS SHOULD REVIEW THE      * 00021600
* SAMPLE FILTERING ENTRIES GIVEN LATER IN THE TABLE.                 * 00021700
********************************************************************** 00021800
*                                                                      00021900
* BATCH JOB STARTED                                                    00022000
*                                                                      00022100
********************************************************************** 00024800
* SAMPLE ENTRIES SHOWING JOB NAME FILTERING IN MESSAGE TABLE         * 00024900
*   TOKEN(2) OF IEF403I/IEF404I CONTAINS JOBNAME, SO YOU MAY SELECT  * 00025000
*   ONLY SPECIFIC JOBS IF DESIRED BY CHANGING JOBNAME BELOW          * 00025100
********************************************************************** 00025200
*                                                                      00025300
*IF MSGID='IEF403I' & TOKEN(2)='jobname' & TEXT=MESSAGE               00025400
*    & DOMAINID = %AOFDOM%                                            00025500
*  THEN EXEC(CMD('DFBTCH 'MESSAGE) ROUTE(ALL *))                      00025600
*      CONTINUE(Y);                                                   00025700
*                                                                     00025700
*IF MSGID='IEF404I' & TOKEN(2)='jobname' & TEXT=MESSAGE               00025800
*    & DOMAINID = %AOFDOM%                                            00025900
*  THEN EXEC(CMD('DFBTCH 'MESSAGE) ROUTE(ALL *))                      00026000
*      CONTINUE(Y);                                                   00025700
********************************************************************** 00026100
* IF MESSAGES ARE ALREADY BEING AUTOMATED, YOU MAY ADD JOBS TO THE   * 00026200
* SDF DISPLAY FACILITY BY CALLING DFBTCH AFTER OTHER PROCESSING      * 00026300
* AS SHOWN BELOW OR USE THE GENERIC ENTIRES                          * 00026400
********************************************************************** 00026500
*                                                                      00026600
*IF MSGID='IEF403I' & TOKEN(2)='jobname' & TOKEN(2)=SVJOB & TEXT=MESSAGE00026700
*  THEN EXEC( CMD('ACTIVMSG JOBNAME=' SVJOB 'UP=YES') ROUTE(ALL *))    00094000
```

```
*       EXEC(CMD('DFBTCH 'MESSAGE) ROUTE(ALL *));                       00026900
*                                                                       00027000
*IF MSGID='IEF404I' & TOKEN(2)='jobname' & TOKEN(2)=SVJOB & TEXT=MESSAGE00027100
* THEN EXEC( CMD('TERMMSG JOBNAME=' SVJOB 'FINAL=YES') ROUTE(ALL *))    00094000
*       EXEC(CMD('DFBTCH 'MESSAGE) ROUTE(ALL *));                       00027300
*                                                                       00027400
********************************************************************** 00027500
*
*  GENERIC ENTRY FOR MESSAGE ID IEF403I
*
 IF MSGID='IEF403I' & TOKEN(2)=SVJOB & DOMAINID=%AOFDOM% & TEXT=MESSAGE 00091000
     THEN EXEC(CMD('DFBTCH 'MESSAGE)  ROUTE( ONE %AOFOPRECOPER%))       00092000
          CONTINUE(Y);                                                  00096000
*                                                                       00022600
* IF MSGID='$HASP373' & TEXT= .'STARTED - INIT'. & TEXT=MESSAGE         00022700
*   THEN EXEC(CMD('DFBTCH 'MESSAGE) ROUTE(ALL *));                      00022800
*        CONTINUE(Y);                                                   00096000
*                                                                       00022900
* BATCH JOB ENDED                                                       00023000
*   SEE ALSO THE ENTRY FOR IEF450I BELOW                                00023010
*                                                                       00023100
*
*  GENERIC ENTRY FOR MESSAGE ID IEF453I
*
 IF MSGID='IEF453I' & TOKEN(2)=SVJOB & DOMAINID=%AOFDOM% & TEXT=MESSAGE 00186000
     THEN  EXEC(CMD('DFBTCH 'MESSAGE)  ROUTE( ONE %AOFOPRECOPER%))      00187000
           CONTINUE(Y);                                                 00192000
*                                                                       00023500
 IF MSGID='IEF251I' & TEXT=MESSAGE                                      00023600
     & DOMAINID = %AOFDOM%                                              00023700
   THEN EXEC(CMD('DFBTCH 'MESSAGE) ROUTE(ALL *))                        00023800
        CONTINUE(Y);                                                    00023900
*                                                                       00023900
*  GENERIC ENTRY FOR MESSAGE ID IEF404I
*
 IF MSGID='IEF404I' & TOKEN(2)=SVJOB & DOMAINID=%AOFDOM% & TEXT=MESSAGE 00103000
     THEN  EXEC(CMD('DFBTCH 'MESSAGE) ROUTE( ONE %AOFOPRECOPER%))       00104000
           CONTINUE(Y);                                                 00108000
*                                                                       00024300
* IF MSGID='$HASP395' &  TEXT=MESSAGE                                   00024400
*   THEN EXEC(CMD('DFBTCH 'MESSAGE) ROUTE(ALL *))                       00024500
*       CONTINUE(Y);                                                    00024600
*                                                                       00024700
*                                                                     * 00027501
********************************************************************** 00027510
*                                                                     * 00027520
* SAMPLE MESSAGE TABLE ENTRIES FOR TSO USERS SDF DISPLAY              * 00027530
*                                                                     * 00027540
*                                                                     * 00027550
* REMOVE THESE ENTRIES IF YOU DO NOT WISH TO HAVE TSO USERS  SHOWN    * 00027560
* IN YOUR SDF DISPLAY.                                                * 00027570
********************************************************************** 00027580
*                                                                       00027590
* TSO USER HAS LOGGED ON OR OFF                                         00027600
*                                                                       00027700
 IF (MSGID='IEF125I' | MSGID='IEF126I') & TEXT=MESSAGE                  00027800
   THEN EXEC(CMD('DFTSOU ' MESSAGE) ROUTE(ALL *));                      00027900
*                                                                       00028000
```

```
*
*  GENERIC ENTRY FOR MESSAGE ID IEF450I
*
* SOMETHING--USER OR JOB--HAS ABENDED, SO CHECK BATCH AND TSO USERS     00028100
                                                                        00129000
 IF MSGID='IEF450I' & TOKEN(2)=SVJOB & DOMAINID=%AOFDOM% & TEXT=MESSAGE 00130000
      & TEXT='IEF450I' . 'ABEND=' SCODE UCODE .                         00131000
     THEN                                                               00132000
       NETLOG(Y) SYSLOG(Y) HOLD(N) BEEP(N) DISPLAY(N)                   00133000
       EXEC( CMD('AOCFILT ' SVJOB ' TERMMSG JOBNAME=' SVJOB             00134000
                 ',CODE1=' SVJOB ',CODE2=' SCODE ',CODE3=' UCODE )      00135000
           ROUTE( ONE %AOFOPRECOPER%))                                  00136000
       EXEC(CMD('DFBTCH 'MESSAGE)                                       00028400
           ROUTE( ONE %AOFOPRECOPER%))                                  00136000
       EXEC(CMD('DFTSOU 'MESSAGE)                                       00028500
           ROUTE( ONE %AOFOPRECOPER%));                                 00136000
                                                                        00137000
*                                                                       00028600
************************************************************************ 00028610
*                                                                      * 00028800
*   SAMPLE MESSAGE TABLE ENTRY FOR CRITICAL MESSAGES                   * 00028900
*                                                                      * 00029000
*   YOU MAY ADD YOUR SELECTIONS TO THE CRITICAL MESSAGE DISPLAY BY     * 00029001
*   INCLUDING ENTRIES FOR THEM LIKE THE ONE BELOW.  THE COLOR OF       * 00029002
*   MESSAGE WILL DEPEND UPON ITS "A", "E","I", OR "O" SUFFIX.          * 00029003
*   OR YOU MAY OVERRIDE THIS USING THE "TYPE=" PARAMETER AS SHOWN      * 00029004
*   BELOW:                                                             * 00029005
*                                                                      * 00029006
*  IF MSGID='IEC031I' & TEXT=MESSAGE                                   * 00029007
*    THEN EXEC(CMD('DFCRIT TYPE=E 'MESSAGE) ROUTE(ALL *));             * 00029008
*                                                                      * 00029009
************************************************************************ 00029010
 IF MSGID='IEC031I' & TEXT=MESSAGE                                      00029200
   THEN EXEC(CMD('DFCRIT 'MESSAGE) ROUTE(ALL *));                       00029300
*
```

EVJMOPCE

```
*********************************************************************** 00010000
*                                                                       * 00020000
*       COPYRIGHT= 5685-151                                             * 00030000
*                  CONTAINS RESTRICTED MATERIALS OF IBM                 * 00040000
*                  (C) COPYRIGHT IBM CORP. 1994,1998              @01C* 00050000
*                  LICENSED MATERIALS - PROPERTY OF IBM                 * 00060000
*                  REFER TO COPYRIGHT INSTRUCTIONS                      * 00070000
*                  FORM NUMBER G120-2083.                               * 00080000
*                                                                       * 00090000
*    DESCRIPTION: SAMPLE DSIPARM - MSG AUTOMATION TABLE FOR AOC OPC     * 00100000
*    EVJMOPCE CHANGE ACTIVITY:                                          * 00110000
*    APAR#                                                              * 00120000
* ----------------------------------------------------------------- * 00130000
* $01=OW35607,V1R4,06NOV98,APC(IG): OPC V2 Exploitation. Add EQQZ128I   00140000
*                                   EQQZ128I (standby mode)             00140000
*                                   EQQZ201I (Standby mode)             00140000
*********************************************************************** 00150000
*                                                                       00160000
*********************************************************************** 00170000
*     EVENT WRITER MESSAGE TABLE ENTRIES                                * 00180000
*********************************************************************** 00190000
*                                                                       00200000
* EQQW065I EVENT WRITER STARTED                                         00210000
*                                                                       00220000
 IF MSGID='EQQW065I' & TOKEN(3) = 'WRITER'                              00230000
   THEN EXEC(CMD('EHKESGUP  ' ) ROUTE(ALL *));                          00240000
*                                                                       00250000
* EQQW011I THE EVENT WRITER ENDED NORMALLY                              00260000
*                                                                       00270000
 IF MSGID='EQQW011I' & TOKEN(4) = 'WRITER' & JOBNAME = SVJOB            00280000
   THEN EXEC(CMD('EHKESRST  ' SVJOB) ROUTE(ALL *));                     00290000
*                                                                       00300000
*********************************************************************** 00310000
*       OPC CONTROLLER      MESSAGE TABLE ENTRIES                       * 00320000
*********************************************************************** 00330000
*                                                                       * 00340000
* OPC controller is up in controller mode (not standby mode)       @01C 00340000
* EQQN013I OPC/ESA JOB TRACKING IS NOW ACTIVE AND CURRENT PLAN     @01A 00350000
*          DD-NAME IS ddname                                       @01A 00350000
*                                                                  @01A 00340000
 IF MSGID='EQQN013I'                                                    00360000
   THEN EXEC(CMD('EHKESGUP  ' ) ROUTE(ALL *));                          00370000
*                                                                       00380000
* EQQZ006I NO ACTIVE SUBTASKS. OPC IS ENDING                            00390000
*                                                                       00400000
 IF MSGID='EQQZ086I' & JOBNAME = SVJOB                                  00410000
   THEN EXEC(CMD('EHKESRST  ' SVJOB) ROUTE(ALL *));                     00420000
*                                                                       00430000
 IF MSGID='EQQZ006I' & JOBNAME = SVJOB                                  00440000
   THEN EXEC(CMD('EHKESRST  ' SVJOB) ROUTE(ALL *));                     00450000
*                                                                       00460000
* OPC APPLICATION IN ERROR                                              00470000
*  ALERT APPLICATION HAS ENDED IN ERROR -                               00480000
```

```
*        REQUIRES   ALERTS TYPE(ERROROPER)   IN OPC/ESA PARMS          00490000
*                                                                       00500000
 IF MSGID='EQQE026I' & TEXT=MESSAGE                                    00510000
   THEN EXEC(CMD('EVJEAC05 'MESSAGE)  ROUTE(ONE *));                   00520000
*                                                                       00530000
*  ALERT A JOB HAS ENDED IN ERROR                                       00540000
*        REQUIRES   ALERTS TYPE(ERROROPER)   IN OPC/ESA PARMS          00550000
*                                                                       00560000
 IF MSGID='EQQE036I' & TEXT=MESSAGE                                    00570000
   THEN EXEC(CMD('EVJEAC03 'MESSAGE)  ROUTE(ONE *));                   00580000
*                                                                       00590000
* OPC/ESA JOB CHANGED FROM ERROR STATUS                                 00600000
* EQQUX007/UX007002 GENERATED MESSAGE                                   00610000
*                                                                       00620000
 IF MSGID='EVJ120I' & TEXT=MESSAGE                                     00630000
   THEN EXEC(CMD('EVJEAC04 'MESSAGE )  ROUTE(ONE *));                  00640000
*                                                                       00650000
********************************************************************* @01A 00310000
* OPC Controller has come up in standby mode.                         @01A 00320000
********************************************************************* @01A 00330000
*                                                                     @01A 00340000
* Initial message when OPC controller starts up in standby mode      @01A 00650000
* EQQZ128I OPC/ESA ACTIVE IN STANDBY MODE                            @01A 00650100
*                                                                     @01A 00650200
 IF MSGID='EQQZ128I' & JOBNAME = SVJOB                                 00650300
   THEN EXEC(CMD('AOCUPDT ' SVJOB',STATUS=HALTED') ROUTE(ONE *));      00650400
*                                                                     @01A 00650500
* Response to MVS F OPCC,STATUS command                              @01A 00650000
* EQQZ201I OPC/ESA STAND-BY mvs_task_name : FULLY_OPERATIONAL        @01A 00650600
*                                                                     @01A 00650700
 IF MSGID='EQQZ201I' & TOKEN(3) = 'STAND-BY' & JOBNAME = SVJOB         00650800
   THEN EXEC(CMD('AOCUPDT ' SVJOB',STATUS=HALTED') ROUTE(ONE *));      00650900
*                                                                     @01A 00651000
```

EVJMOPCA

```
************************************************************************ 00029500
*                                                                      * 00029600
*       COPYRIGHT= 5685-151                                            * 00029700
*                  CONTAINS RESTRICTED MATERIALS OF IBM                * 00029800
*                  (C) COPYRIGHT IBM CORP. 1995                        * 00029900
*                  LICENSED MATERIALS - PROPERTY OF IBM                * 00030000
*                  REFER TO COPYRIGHT INSTRUCTIONS                     * 00030100
*                  FORM NUMBER G120-2083.                              * 00030200
*                                                                      * 00030300
*    DESCRIPTION: SAMPLE DSIPARM - MSG AUTOMATION TABLE FOR AOC OPC    * 00030400
*    EVJMOPCA CHANGE ACTIVITY:                                         * 00030500
*    APAR#                                                             * 00030600
* ---------------------------------------------------------------------- * 00030700
*                                                                        00030800
************************************************************************ 00031100
*       OPC/A-EWTR(EMS1)  MESSAGE TABLE ENTRIES                        * 00031200
************************************************************************ 00031300
*                                                                        00031400
* DRKW005I EVENT WRITER STARTED                                         00031500
*                                                                       00031600
 IF MSGID='DRKW005I' & TOKEN(3) = 'WRITER'                              00031700
   THEN EXEC(CMD('EHKESGUP  ' ) ROUTE(ALL *));                          00031800
*                                                                       00031900
* DRKW011I THE EVENT WRITER ENDED NORMALLY                              00032000
*                                                                       00032100
 IF MSGID='DRKW011I' & TOKEN(4) = 'WRITER' & JOBNAME = SVJOB            00032200
   THEN EXEC(CMD('EHKESRST  ' SVJOB) ROUTE(ALL *));                     00032300
*                                                                       00032400
************************************************************************ 00032500
*       OPC/A-PCS(PCS1)     MESSAGE TABLE ENTRIES                      * 00032600
************************************************************************ 00032700
*                                                                      * 00032800
* CSYN012I OPCD JOB TRACKING EVENTS ARE NOW BEING LOGGED                00032900
*                                                                       00033000
 IF MSGID='CSYN013I'                                                    00033100
   THEN EXEC(CMD('EHKESGUP  ' ) ROUTE(ALL *));                          00033200
*                                                                       00033300
* DRKZ006I NO ACTIVE OPC/A SUBTASKS. OPC/A IS ENDING                    00033400
*                                                                       00033500
 IF MSGID='DRKZ006I' & JOBNAME = SVJOB                                  00033600
   THEN EXEC(CMD('EHKESRST  ' SVJOB) ROUTE(ALL *));                     00033700
*                                                                       00033800
* OPC/A JOB IN ERROR                                                    00033900
*  ALERT APPLICATION HAS ENDED IN ERROR -                               00034000
*        REQUIRES   ALERTS TYPE(ERROROPER)   IN OPCA PARMS              00034100
*                                                                       00034200
 IF MSGID='CSYE026I' & TEXT=MESSAGE                                     00034300
   THEN EXEC(CMD('EVJEAC05 'MESSAGE)  ROUTE(ONE *));                    00034400
*                                                                       00034700
*  ALERT A JOB HAS ENDED IN ERROR                                       00034800
*        REQUIRES   ALERTS TYPE(ERROROPER)   IN OPCA PARMS              00034900
*                                                                       00035000
 IF MSGID='CSYE036I' & TEXT=MESSAGE                                     00035100
```

```
   THEN EXEC(CMD('EVJEAC03 'MESSAGE)  ROUTE(ONE *));                  00035200
*                                                                     00035600
* OPC/A JOB CHANGED FROM ERROR STATUS                                 00035700
* DRKUX007/UX007002 GENERATED MESSAGE                                 00035800
*                                                                     00035900
 IF MSGID='UX007021' & TEXT=MESSAGE                                   00036000
   THEN EXEC(CMD('EVJEAC04 'MESSAGE )  ROUTE(ONE *));                 00036100
*                                                                     00036200
 IF MSGID='EVJ120I' & TEXT=MESSAGE                                    00036400
   THEN EXEC(CMD('EVJEAC04 'MESSAGE )  ROUTE(ONE *));                 00036500
*                                                                     00036600
```

Appendix E. Sample OPC Automation Command Synonyms

EVJCMD, a sample file that defines command synomyns, is shipped with OPC Automation.

EVJCMD

```
*********************************************************************
*    (C) COPYRIGHT IBM CORP. 1990, 1998                          @02C*
*    PROGRAM NUMBER:   5685-151                                     *
*    DESCRIPTION: SAMPLE DSIPARM - DSICMD ENTRIES FOR AOC OPC       *
*    EVJCMD - CHANGE ACTIVITY:                                      *
*  APAR#                                                            *
* ------------------------------------------------------------------ *
* $02=OW35607 12/01/98 APC(MP): OPC V2 EXPLOITATION.                *
*                               Add EVJECMD, EVJESLAV, EVJEFCTL,    *
*                                   EVJEALST                        *
* $01=OW24230 12/13/96 RFM MSGEVJ029I EVJSTS COMMAND ENCOUNTERED    *
*                          UNEXPECTED EVENT TIMEOUT FROM OPCAQRY    *
*                          (EVJECGAA) -- PERFORMANCE PROBLEM        *
*                                                                   *
* AOCOPC   04/25/94    INITIAL VERSION FOR AOC OPC                  *
*********************************************************************
*   ADDED FOR NEW STATUS FILE ORGANIZATION                          *
*********************************************************************
EVJSCPST   CMDMDL   MOD=EVJSCPST,TYPE=R,RES=Y,ECHO=Y          /* @01C*/
           CMDSYN   EHKPCST
           CMDSYN   EVJSTS
EVJSMPST   CMDMDL   MOD=EVJSMPST,TYPE=D,RES=Y,PARSE=N         /* @01C*/
           CMDSYN   EHKPMST
*********************************************************************
* MAIN PROCESSING ROUTINES                                          *
*********************************************************************
EVJESPCP   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
           CMDSYN   OPCACOMP
EVJESPIN   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
EVJESPRQ   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
EVJESPSC   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
EVJESPTE   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
EVJESPVY   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
EVJESUSF   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
EVJEMSG    CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
EVJEMSG1   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
EVJSASND   CMDMDL   MOD=EVJSASND,RES=Y,TYPE=R,ECHO=Y
*********************************************************************
* OPERATOR COMMANDS                                                 *
*********************************************************************
EVJE$UUP   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
           CMDSYN   UP
EVJE$UDN   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
           CMDSYN   DOWN
EVJE$URS   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
           CMDSYN   BOUNCE
```

```
|                     CMDSYN   $RESTART
|            *        CMDSYN   RESTART                                  /* @01C*/
|            EVJECGAA CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   OPCAQRY
|            *
|            EVJEAA00 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   OPC
|                     CMDSYN   OPCA
|                     CMDSYN   OPCAO
|            *
|            EVJEAB00 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   DFTAPO
|            *
|            EVJEAB01 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   DFBTCH
|            *
|            EVJEAB02 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   DFTAPM
|            *
|            EVJEAB03 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   DFTSOU
|            *
|            EVJEAB04 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   DFTSOR
|            *
|            EVJEAB05 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   DFTAPK
|            *
|            EVJEAB06 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   DFCRIT
|            *
|            EVJEAB07 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   DFUPDT
|            *
|            EVJEAB10 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   DFDELT
|            *
|            EVJEAB11 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|            *
|            EVJEAB12 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   DFCOPY
|            *
|            EVJEAB13 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|            *
|            EVJEAB14 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|            *
|            EVJEAC00 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|                     CMDSYN   OPCACMD
|            *                                                          /* @02A*/
|            EVJECMD  CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R                /* @02A*/
|            EVJESLAV CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R                /* @02A*/
|            *                                                          /* @02A*/
|            EVJEFCTL CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R                /* @02A*/
|                     CMDSYN   EVJRFCTL                                 /* @02A*/
|            *                                                          /* @02A*/
|            EVJEAC01 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|            EVJEAC02 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
|            EVJEAC03 CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
```

```
EVJEAC04   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
*
EVJMPM00   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
EVJMPM10   CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R
*
EVJESRST   CMDMDL   MOD=DSICCP,ECHO=Y,TYPE=R
           CMDSYN  SRSTAT
*
* POST ROUTINE
EVJSAOPS    CMDMDL   MOD=EVJSAOPS,RES=N
            CMDSYN  OPCAPOST
            CMDSYN  OPCPOST
* AOC/MVS STATUS CHANGE EXIT
EVJEXSTA    CMDMDL   MOD=DSICCP,TYPE=R
            CMDSYN  AOFEXSTA
* USER INTERFACE CMDS
EVJSACAL    CMDMDL   MOD=EVJSACAL,RES=N
            CMDSYN  OPCACAL
            CMDSYN  OPCCAL
EVJSADTL    CMDMDL   MOD=EVJSADTL,RES=N
            CMDSYN  OPCADTL
            CMDSYN  OPCDTL
EVJSALST    CMDMDL   MOD=EVJSALST,RES=N
EVJEALST    CMDMDL   MOD=DSICCP,ECHO=N,TYPE=R                  /* @02A*/
            CMDSYN  OPCALIST
            CMDSYN  OPCLIST
EVJSAMOD    CMDMDL   MOD=EVJSAMOD,RES=N
            CMDSYN  OPCAMOD
            CMDSYN  OPCMOD
EVJSAINS    CMDMDL   MOD=EVJSAINS,RES=N
            CMDSYN  OPCSRST
*****************************************************************
*       END OF AOC OPC ENTRIES                                  *
*****************************************************************
```

Appendix F. Sample OPC Automation Error Display Panel Source

EVJOPCA

EVJOPCA, a sample file that describes the OPC Automation error display panel source, is shipped with OPC Automation.

```
/*********************************************************************/
/*      COPYRIGHT= 5685-151                                          */
/*                 CONTAINS RESTRICTED MATERIALS OF IBM              */
/*                 (C) COPYRIGHT IBM CORP. 1990, 1995                */
/*                 LICENSED MATERIALS - PROPERTY OF IBM              */
/*                 REFER TO COPYRIGHT INSTRUCTIONS                   */
/*                 FORM NUMBER G120-2083.                            */
/*                                                                   */
/*********************************************************************/
/* OPC ERROR DISPLAY PANEL   PN40381 PN02975   */
P(SY1OPCA,24,80,SYSTEM,SYSTEM, , , )
TF(01,02,09,B,NORMAL)
TT(SY1OPCA)
TF(02,20,60,T,NORMAL)
TT(---- OPC: Applications in Error ----)
SF(SY1.OPCERR,04,02,27,N, , ,01)
ST( )
SF(SY1.OPCERR,04,28,53,N, , ,10)
ST( )
SF(SY1.OPCERR,04,55,80,N, , ,19)
ST( )
SF(SY1.OPCERR,06,02,27,N, , ,02)
ST( )
SF(SY1.OPCERR,06,28,53,N, , ,11)
ST( )
SF(SY1.OPCERR,06,55,80,N, , ,20)
ST( )
SF(SY1.OPCERR,08,02,27,N, , ,03)
ST( )
SF(SY1.OPCERR,08,28,53,N, , ,12)
ST( )
SF(SY1.OPCERR,08,55,80,N, , ,21)
ST( )
SF(SY1.OPCERR,10,02,27,N, , ,04)
ST( )
SF(SY1.OPCERR,10,28,53,N, , ,13)
ST( )
SF(SY1.OPCERR,10,55,80,N, , ,22)
ST( )
SF(SY1.OPCERR,12,02,27,N, , ,05)
ST( )
SF(SY1.OPCERR,12,28,53,N, , ,14)
ST( )
SF(SY1.OPCERR,12,55,80,N, , ,23)
ST( )
SF(SY1.OPCERR,14,02,27,N, , ,06)
```

```
ST( )
SF(SY1.OPCERR,14,28,53,N, , ,15)
ST( )
SF(SY1.OPCERR,14,55,80,N, , ,24)
ST( )
SF(SY1.OPCERR,16,02,27,N, , ,07)
ST( )
SF(SY1.OPCERR,16,28,53,N, , ,16)
ST( )
SF(SY1.OPCERR,16,55,80,N, , ,25)
ST( )
SF(SY1.OPCERR,18,02,27,N, , ,08)
ST( )
SF(SY1.OPCERR,18,28,53,N, , ,17)

ST( )
SF(SY1.OPCERR,18,55,80,N, , ,26)
ST( )
SF(SY1.OPCERR,20,02,27,N, , ,09)
ST( )
SF(SY1.OPCERR,20,28,53,N, , ,18)
ST( )
SF(SY1.OPCERR,20,55,80,N, , ,27)
ST( )
TF(24,01,48,T,NORMAL)
TT(PF1=HELP 2=DETAIL 3=END        6=ROLL 7=UP 8=DN)
TF(24,51,79,T,NORMAL)
TT(        10=LF 11=RT 12=TOP)
EP
```

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/A, 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/A 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. Comparable to the OPC/A PCS.

controlling processor. The processor on which the Production Control System (PCS) executes. See host processor.

critical path. The route within a network with the least amount of slack time.

critical resource. The term used in OPC 3.1 for 'workstation resource'. See workstation resource.

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/A to use when you do not specify a calendar in an application description. (2) A calendar that OPC/A uses if you have neither specified a calendar in an application description, nor defined your own default calendar. (3) The name (DEFAULT) given to your OPC/A Release 1 calendar by the migration program if you migrate to OPC/A Release 2. (OPC/A Release 1 allows you only one calendar, Release 2 allows you multiple calendars.)

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.

event. An action in the operations department that results in an operation's change of status and a change in the current schedule.

event handler. A separate load module that changes the status of operations.

event manager. The OPC/A function that processes all job tracking events and determines which of these are OPC/A related.

Event Management Subsystem (EMS). The Event Management Subsystem runs as an MVS subsystem, tracking and logging event records on DASD. It is required on every processor in an OPC/A configuration. Equivalent to the OPC/ESA Tracker.

event reader. An OPC/A task that reads event records from an event data set.

event tracking. See job tracking.

Event Triggered Tracking (ETT). A component of OPC/A that waits for specific events to occur; and when they occur, it adds a predefined application to the current plan. ETT recognizes two types of events: the reader event, which occurs when a job enters the JES reader, and the resource event, which occurs when the availability status of a special resource is set to 'yes'.

event writer. An OPC/A task that writes event records in an event data set.

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

feedback limit. A numeric value from 100 to 999 which defines the limits within which actual data collected in job tracking is fed back and used by OPC/A.

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/A 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 tracking. A function of OPC/A PCS that follows events in the operations department in real time and records status changes in the current schedule.

job setup. The preparation of a set of JCL statements for a job at an OPC/A workstation you defined for this purpose.

job submission. An OPC/A process that presents jobs to MVS for running on an OPC/A 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/A) 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

Network Event Communicator (NEC). The Network Event Communicator collects event information from event data sets and transmits it to the NEC executing on the OPC/A controlling processor. It is required on the controlling processor and on each remote processor where remote job tracking is used.

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/A. Operation Planning and Control/Advanced

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.

OPC/A remote processor. A processor connected to the OPC/A host processor by a VTAM network. An OPC/A event writer (EMS) and an event transmitter (NEC) are installed on the remote processor and transmit events to the OPC/A host processor by VTAM. With OPC/ESA, the Tracker combines these functions.

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/A 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.

Production Control System (PCS). The Production Control System contains the controlling functions, all dialogs, and OPC/A's own batch programs. The program controls the entire OPC/A installation, including remote sites. Equivalent to the OPC/ESA Controller.

program interface. An OPC/A interface that allows a user-written program to issue various types of requests to the OPC/A 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. Comparable to the OPC/A EMS and NEC components.

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

I

L

M

N

O

P

R

S

T

U

W

Communicating Your Comments to IBM

System Automation for OS/390
AOC/MVS OPC Automation
Programmer's Reference
and Installation Guide
Version 1 Release 4

Publication No. SC23-3820-02

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of the book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF form and either send it postage-paid in the United States, or directly to:

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

- If you prefer to send comments by FAX, use this number:
  - (Germany): 07031-16-3456
  - (Other countries): (+49)+7031-16-3456

- If you prefer to send comments electronically, use this network ID:

  IBM Mail Exchange: DEIBMBM9 at IBMMAIL

  Internet: s390id@de.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

System Automation for OS/390
AOC/MVS OPC Automation
Programmer's Reference
and Installation Guide
Version 1 Release 4

Publication No. SC23-3820-02

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 _____

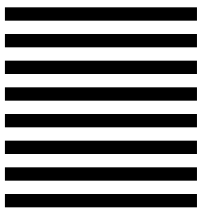
Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC23-3820-02

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

International Business Machines Corporation
Attn: Dept EHJ - BP/003D
6300 Diagonal Highway
Boulder, CO 80301-9151

Fold and Tape **Please do not staple** Fold and Tape

SC23-3820-02

Cut or Fold
Along Line

Program Number: 5685-151

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC23-3820-02