

System Automation for OS/390



OPC Automation Programmer's Reference and Operator's Guide

Version 2 Release 1

System Automation for OS/390



OPC Automation Programmer's Reference and Operator's Guide

Version 2 Release 1

Fourth Edition (October 2000)

This edition applies to System Automation for OS/390 Version 2 Release 1 (5645-006), and to AOC/MVS OPC Automation Version 1 Release 4 (5685-151), IBM licensed programs, and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

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

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

Internet: s390id@de.ibm.com
World Wide Web: <http://www.s390.ibm.com/os390/>

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.

© Copyright International Business Machines Corporation 1990, 2000. All rights reserved.

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

Contents

| | |
|----------------------------------------------------|------------|
| Figures | v |
| Tables | vii |
| Notices | ix |
| Trademarks | ix |
| About This Book. | xi |
| Who Should Use This Book | xi |
| What's in This Book? | xi |
| Notation for Format Descriptions | xi |
| Related Publications | xii |
| The System Automation for OS/390 Library | xii |
| Related Product Information | xiii |

Part 1. Introducing OPC Automation 1

Chapter 1. Principal Concepts of SA OS/390 3

| | |
|----------------------------------------------------------------|---|
| Automation Policies | 3 |
| Goal-Driven Automation | 4 |
| Dependencies, Request Propagation, and Desired State | 4 |
| Persistency of Requests and Conflicting Requests | 6 |
| Triggers | 7 |
| Service Periods | 8 |
| Application Groups | 8 |
| SA OS/390 and the NetView Message Automation Table | 9 |

Chapter 2. Functions of OPC Automation 11

| | |
|-------------------------------------------------------------------|----|
| Basic Concept of OPC Automation. | 11 |
| Defining SA OS/390 to OPC | 12 |
| Defining OPC and OPC-Controlled Subsystems to SA OS/390 | 12 |
| System Initialization with OPC Automation | 12 |
| NetView Interface to OPC Automation | 13 |
| Request and Confirmation Transaction Flow | 13 |
| Request Buffers and OPC Automation Log Entries | 15 |
| OPC Automation Special Resources | 16 |
| Possible Uses of OPC Automation. | 17 |
| Changing Online Hours of Availability | 17 |
| Cycling Individual Online Databases | 18 |
| Scheduling Time for Testing | 19 |
| Distributing and Updating Data Across Multiple Systems | 19 |
| Complex Application Recovery. | 20 |

Chapter 3. Structure of OPC Automation. 23

| | |
|--------------------------|----|
| Flow Overview | 23 |
| Initialization | 23 |

| | |
|---------------------------------------------------------|----|
| Request Flow | 23 |
| Automated Operator Tasks | 31 |
| Initialization | 31 |
| Startup of OPC Components | 31 |
| Startup of OPC-Controlled Subsystems | 32 |
| Request Handling in the OPC Controller System | 33 |
| Request Handling in the OPC Tracker System | 35 |
| Completion and Timer Flags. | 36 |
| Operations Control | 36 |
| EVJESPIN Module | 36 |
| Obtaining Information from OPC | 37 |
| Automated Recovery | 38 |

Part 2. Customizing OPC Automation 41

Chapter 4. Customizing OPC Automation. 43

| | |
|----------------------------------------------------------------------------|----|
| Hardware and Software Requirements | 43 |
| Installation Related Steps | 43 |
| Step 1: Updating MPFLSTxx. | 43 |
| Step 2: Add Libraries to OPC and Recycle | 43 |
| NetView Related Steps | 44 |
| Step 1: Add OPC/ESA Data Sets and Allocate EQQMLOG | 44 |
| Step 2: Check the NetView Message Automation Table for Conflicts | 45 |
| OPC Automation Initial Customization | 45 |
| Step 1: Basic OPC Automation Common Policy Definitions. | 45 |
| Step 2: Integrate Existing Exit 7 with OPC Automation | 46 |
| Step 3: Initializing the OPC Automation Status File | 47 |
| OPC Automation Test Scenario | 47 |
| Problem Determination Suggestions | 48 |

Chapter 5. Setting Up OPC Automation 49

| | |
|---------------------------------------------------------|----|
| Defining Automated OPC Applications | 49 |
| Defining Information for OPC Automation in OPC. | 49 |
| Example of an Application Making a Request | 52 |
| Executing OPC Requests with OPC Automation | 56 |
| OPC Requests and MESSAGES/USER DATA Keywords | 57 |
| Request Parameters and the &EHKVARi Variables | 59 |
| Request Types | 61 |

Chapter 6. MESSAGES/USER DATA Entries and USER E-T Pairs for OPC Automation. 65

| | |
|----------------------------------------------------|----|
| Translating Format Descriptions | 65 |
| OPC-Specific MESSAGES/USER DATA Keywords | 70 |

| | |
|--------------------|----|
| OPCA | 71 |
| OPCACMD | 74 |
| OPCAPARM | 78 |

Chapter 7. OPC Automation Common Routines and Data Areas 81

| | |
|------------------------------------------|----|
| OPC Automation Common Routines | 81 |
| EVJESHUT | 82 |
| OPCACAL | 83 |
| OPCACMD | 84 |
| OPCACOMP | 86 |
| OPCALIST | 87 |
| OPCAMOD | 89 |
| OPCAPOST | 92 |
| OPCSRST | 93 |
| Data Areas | 93 |
| Requestor ID Block (&EHKVAR9) | 94 |
| Request Buffer | 95 |

Chapter 8. Guidelines for User-Written Operations 97

| | |
|--------------------------------------------------------------------|-----|
| User Functions Related to an SA OS/390-Defined Subsystem | 97 |
| Flow of Control | 97 |
| Implementing Completion of a Request | 98 |
| Non-Subsystem Operations | 101 |
| Flow of Control | 102 |
| Parameters Passed to a User Exit | 103 |
| Interaction with CICS Automation | 103 |
| Interaction with IMS Automation | 104 |

Part 3. Using OPC Automation 107

Chapter 9. Operator's Guide 109

| | |
|--------------------------------------------------|-----|
| How the Operator Uses OPC Automation | 109 |
| Monitoring the Health of the System | 109 |
| Problems in an OPC-Defined Application | 110 |

Chapter 10. OPC Automation Operator Commands 117

| | |
|--------------------------------------------------|-----|
| OPC Automation Main Menu and Tutorials | 118 |
| SDF—Status Display Facility Panel | 119 |

| | |
|------------------------------------------------------------|-----|
| OPC Monitor Panel | 120 |
| OPC Automation: Applications in Error Panel | 121 |
| Critical Messages Panel | 122 |
| Tapes Panel | 124 |
| TSO Users Status Panel | 126 |
| Batch Display Panel | 128 |
| OPCACMD—Interacting Dynamically with OPC | 129 |
| OPCAQRY—Display Status of Operations | 131 |
| Selecting Actions | 132 |
| OPCAPOST—Posting an OPC Operation from SA OS/390 | 134 |
| SRSTAT—Determining OPC Special Resource Status | 135 |
| EVJESPIN—Initialization | 136 |

Chapter 11. Resynchronization and Recovery Considerations 137

| | |
|-----------------------------------------------------------------|-----|
| Examples and Scenarios | 138 |
| Loss of Contact Between OPC and OPC Automation | 138 |
| Backup on a Different Processor | 138 |
| Long Term Outage | 139 |
| Example Using Doubly-Defined NetView Domain IDs | 140 |
| Automated Recovery Functions | 141 |
| OPC Actions in a Loss of Contact Situation | 141 |
| OPC Automation Actions in a Loss of Contact Situation | 141 |

Appendix. Status Display Facility Enhancements 143

| | |
|------------------------------------------------------------|-----|
| Coding Reference | 143 |
| CLISTs Used to Implement the Supplied Extensions | 143 |
| DFUPDT | 145 |
| DFCOPY | 146 |
| DFCRIT | 147 |
| EVJEAB11 | 148 |

Glossary of Terms 149

Index 157

Figures

| | | | |
|---------------------------------------------------------------------------------------------|----|----------------------------------------------------------------------------|-----|
| 1. Example of Start Dependencies | 5 | 35. Code Processing Panel of the Customization Dialogs | 69 |
| 2. Example of Conflicting Requests | 6 | 36. OPCACMD in a USER E-T Pair | 70 |
| 3. Example of a Request Involving a Group | 9 | 37. Request Flow for a Subsystem-Related User Function | 98 |
| 4. NetView-OPC Interface Flow | 14 | 38. User Exit UXxxxxxx Flow | 101 |
| 5. NetView Log Entry of an OPC Generated Request | 16 | 39. Condition Code Driven Application Flow | 102 |
| 6. Example of Cycling Individual Online Databases | 18 | 40. OPCACMD Entry for Interaction with CICS | 104 |
| 7. NetView-OPC Interface Flow. | 24 | 41. Defining Sample CICS Application in OPC | 104 |
| 8. EQQUX007 Exit | 25 | 42. OPCACMD Entry for Interaction with IMS | 105 |
| 9. PPI Dispatcher | 26 | 43. Defining Sample IMS Application in OPC | 105 |
| 10. Verify Module. | 26 | 44. OPC Automation Monitor Panel | 110 |
| 11. Request Module | 28 | 45. OPC Automation: Applications in Error Panel | 111 |
| 12. Status Change Module | 29 | 46. SA/OPC - Display or Modify OPC Data Panel | 111 |
| 13. Timer Module. | 30 | 47. Entering MAINT in the Application Field | 112 |
| 14. OPCAPOST Command Processor | 30 | 48. Display or Modify OPC Data Panel | 113 |
| 15. OPC/ESA Startup During IPL Process | 32 | 49. Entering B (Browse) in the CMD Field | 113 |
| 16. Request Handling in the OPC Controller Processor | 34 | 50. SA/OPC - OPC Occurrence Data Panel | 114 |
| 17. Request Flow for a Base SA OS/390 Function | 35 | 51. Entering C (Change) in the CMD Field | 114 |
| 18. Sample NVxx Workstation Definition in OPC | 50 | 52. SA/OPC - Main Menu | 118 |
| 19. Defining the MAINT Application in OPC | 51 | 53. SA OS/390 Support Systems—Status Display Facility Panel | 119 |
| 20. 'Operations' OPC Panel Showing OPC Automation Requests | 51 | 54. OPC Monitor Panel | 120 |
| 21. Request Using Optional Parameters | 52 | 55. Applications in Error Panel | 121 |
| 22. Browsing Operations Including OPC Automation Requests | 52 | 56. Critical Messages Panel | 122 |
| 23. RMF Maintenance Application Primary Panel in OPC | 53 | 57. Critical Message—Detail Status Display | 123 |
| 24. Operations in the MAINT Application | 54 | 58. Tape Unit Display Panel | 124 |
| 25. Operations Text Detail Panel | 54 | 59. Tape User—Detail Status Display | 125 |
| 26. Using Time as a Dependency | 55 | 60. TSO Users Status Panel | 126 |
| 27. OPC/ESA Operations Panel | 57 | 61. TSO User—Detail Status Display | 127 |
| 28. Specifying the Command for a Request | 58 | 62. Batch Job Display Panel | 128 |
| 29. Specifying Expected Status and Time Interval | 59 | 63. Batch Jobs—Detail Status Display | 128 |
| 30. Specifying a Command that Requires Parameter Information | 60 | 64. SA/OPC - Display or Modify OPC Data Panel | 129 |
| 31. Specifying Expected Status and Time Interval for Different Request Parameters | 61 | 65. SA/OPC - Display or Modify OPC Data Panel (Operation Summary). | 130 |
| 32. Message Processing Panel of the Customization Dialogs 1 | 66 | 66. Operation Detail Panels | 130 |
| 33. CMD Processing Panel of the Customization Dialogs | 67 | 67. SA/OPC - Operation Status Display Panel | 131 |
| 34. Message Processing Panel of the Customization Dialogs 2 | 68 | 68. OPC Automation: Operation Status Detail Panel | 133 |
| | | 69. Mapping of NVxx Workstations to Domain IDs | 140 |

Tables

| | | | |
|-------------------------------------------------------------------|------|--------------------------------------------------------------------------------------------|-----|
| 1. System Automation for OS/390 Library | xii | 5. Request Buffer Layout for Standard Subsystem Operations | 95 |
| 2. Related Products Books | xiii | 6. Request Buffer Layout for Nonsubsystem, User Extension (UXaaaaaaa) Operations | 95 |
| 3. OPC Automation Items Defined in OPC | 49 | 7. OPC Automation Commands | 117 |
| 4. Lengths and Values of Task Global Variable (EHKVAR9) | 94 | | |

Notices

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

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

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

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

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

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

Trademarks

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

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

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

About This Book

This book describes how to customize and operate OPC Automation. OPC Automation is a feature of SA OS/390 that brings together batch and online console automation into a common focal point. OPC Automation automates, simplifies, and standardizes console operations and the management of component, application, and production related tasks.

Who Should Use This Book

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

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

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

Installing and customizing OPC Automation requires a programmer's understanding of NetView, OPC, SA OS/390, and OPC Automation, because most of the definitions take place in these programs. Also, you will modify JCL, command lists, and programs for some of the automation functions

- Operators and administrators who manage and monitor OPC.

These users will mainly need part 1 and part 3.

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

What's in This Book?

This book contains the following:

Part 1. Introducing OPC Automation

Explains some main concepts of SA OS/390 and describes the functions of OPC Automation.

Part 2. Customizing OPC Automation

Describes the customization of OPC Automation and contains reference sections for MESSAGES policy items and for the programming interface.

Part 3. Using OPC Automation

Describes the operator interface of OPC Automation.

Notation for Format Descriptions

The reference sections of this manual contain format descriptions of commands and of entries in the SA OS/390 policy database. The notation used for these descriptions is as follows:

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

{A|B|C}

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

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

[A|B|C]

indicates that you may enter A, B, or C, or you may omit the operand.

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

[A|B|C]

indicates that if no operand is specified, B is assumed.

- Lowercase italicized items are variables; substitute your own value for them.
- Uppercase items must be entered exactly as shown.
- Parentheses must be entered as shown.
- Where operands can be abbreviated, the abbreviations are shown in capital letters. For example, ALL can be entered as A or ALL.
- Commas are used as delimiters between parameters. The last parameter does not require a comma after it. Because of this, we place the comma in front of a parameter to show that if you add this parameter, you need a comma, as for example in

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

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

Related Publications

The System Automation for OS/390 Library

The following table shows the information units in the System Automation for OS/390 library:

Table 1. System Automation for OS/390 Library

| Title | Order Number |
|-------------------------------------------------------------------------------------------------|---------------------|
| <i>System Automation for OS/390 General Information</i> | GC33-7036 |
| <i>System Automation for OS/390 Licensed Program Specifications</i> | SC33-7037 |
| <i>System Automation for OS/390 Planning and Installation</i> | SC33-7038 |
| <i>System Automation for OS/390 Customizing and Programming</i> | SC33-7035 |
| <i>System Automation for OS/390 Defining Automation Policy</i> | SC33-7039 |
| <i>System Automation for OS/390 User's Guide</i> | SC33-7040 |
| <i>System Automation for OS/390 Messages and Codes</i> | SC33-7041 |
| <i>System Automation for OS/390 Operator's Commands</i> | SC33-7042 |
| <i>System Automation for OS/390 Programmer's Reference</i> | SC33-7043 |
| <i>System Automation for OS/390 CICS Automation Programmer's Reference and Operator's Guide</i> | SC33-7044 |
| <i>System Automation for OS/390 IMS Automation Programmer's Reference and Operator's Guide</i> | SC33-7045 |
| <i>System Automation for OS/390 OPC Automation Programmer's Reference and Operator's Guide</i> | SC23-7046 |

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

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

This softcopy collection includes 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

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. Related Products Books

| Title | Order Number |
|--------------------------------------------------------------------------------------|---------------------|
| <i>MVS/ESA MVS Configuration Program Guide and Reference</i> | GC28-1817 |
| <i>MVS/ESA Planning: Dynamic I/O Configuration</i> | GC28-1674 |
| <i>MVS/ESA Support for the Enterprise Systems Connection</i> | GC28-1140 |
| <i>MVS/ESA Planning: APPC Management</i> | GC28-1110 |
| <i>MVS/ESA Application Development Macro Reference</i> | GC28-1822 |
| <i>MVS/ESA SP V5 System Commands</i> | GC28-1442 |
| <i>MVS/ESA SPL Application Development Macro Reference</i> | GC28-1857 |
| <i>OS/390 Hardware Configuration Definition: User's Guide</i> | SC28-1848 |
| <i>OS/390 Information Roadmap</i> | GC28-1727 |
| <i>OS/390 Information Transformation</i> | GC28-1985 |
| <i>OS/390 Introduction and Release Guide</i> | GC28-1725 |
| <i>OS/390 JES Commands Summary</i> | GX22-0041 |
| <i>OS/390 Licensed Program Specifications</i> | GC28-1728 |
| <i>OS/390 Printing Softcopy Books</i> | S544-5354 |
| <i>OS/390 Starting Up a Sysplex</i> | GC28-1779 |
| <i>OS/390 Up and Running!</i> | GC28-1726 |
| <i>Planning for the 9032 Model 3 and 9033 Enterprise Systems Connection Director</i> | SA26-6100 |
| <i>Resource Access Control Facility (RACF) Command Language Reference</i> | SC28-0733 |
| <i>S/390 MVS Sysplex Overview – An Introduction to Data Sharing and Parallelism</i> | GC23-1208 |
| <i>S/390 MVS Sysplex Systems Management</i> | GC23-1209 |
| <i>S/390 Sysplex Hardware and Software Migration</i> | GC23-1210 |
| <i>S/390 MVS Sysplex Application Migration</i> | GC23-1211 |
| <i>S/390 Managing Your Processors</i> | GC38-0452 |
| <i>Tivoli/Enterprise Console User's Guide Volume I</i> | GC31-8334 |
| <i>Tivoli/Enterprise Console User's Guide Volume II</i> | GC31-8335 |
| <i>Tivoli/Enterprise Console Event Integration Facility Guide</i> | GC31-8337 |
| <i>Tivoli for OS/390 NetView V1R3 Administration Reference</i> | SC31-8222 |
| <i>Tivoli for OS/390 NetView V1R3 Application Programmer's Guide</i> | SC31-8223 |

Table 2. Related Products Books (continued)

| Title | Order Number |
|-------------------------------------------------------------------------------------------------------------|---------------------|
| <i>Tivoli for OS/390 NetView V1R3 APPN Topology and Accounting Agent Guide</i> | SC31-8224 |
| <i>Tivoli for OS/390 NetView V1R3 Automation Guide</i> | SC31-8225 |
| <i>Tivoli for OS/390 NetView V1R3 AON Customization Guide</i> | SC31-8662 |
| <i>Tivoli for OS/390 NetView V1R3 AON User's Guide</i> | GC31-8661 |
| <i>Tivoli for OS/390 NetView V1R3 Bridge Implementation</i> | SC31-8238 |
| <i>Tivoli for OS/390 NetView V1R3 Command Reference Vol. 1</i> | SC31-8227 |
| <i>Tivoli for OS/390 NetView V1R3 Command Reference Vol. 2</i> | SC31-8227 |
| <i>Tivoli for OS/390 NetView V1R3 Customization Guide</i> | SC31-8228 |
| <i>Tivoli for OS/390 NetView V1R3 Customization: Using Assembler</i> | SC31-8229 |
| <i>Tivoli for OS/390 NetView V1R3 Customization: Using Pipes</i> | SC31-8248 |
| <i>Tivoli for OS/390 NetView V1R3 Customization: Using PL/I and C</i> | SC31-8230 |
| <i>Tivoli for OS/390 NetView V1R3 Customization: Using REXX and the NetView Command List Language</i> | SC31-8231 |
| <i>Tivoli for OS/390 NetView V1R3 Data Model Reference</i> | SC31-8232 |
| <i>Tivoli for OS/390 NetView V1R3 Installation and Administration Guide</i> | SC31-8236 |
| <i>Tivoli for OS/390 NetView V1R3 Messages and Codes</i> | SC31-8237 |
| <i>Tivoli for OS/390 NetView V1R3 MultiSystem Manager User's Guide</i> | SC31-8607 |
| <i>Tivoli for OS/390 NetView V1R3 NetView Graphic Monitor Facility User's Guide</i> | GC31-8234 |
| <i>Tivoli for OS/390 NetView V1R3 NetView Management Console User's Guide</i> | GC31-8665 |
| <i>Tivoli for OS/390 NetView V1R3 User's Guide</i> | SC31-8241 |
| <i>Tivoli for OS/390 NetView V1R3 Planning Guide</i> | GC31-8226 |
| <i>Tivoli for OS/390 NetView V1R3 RODM and GMFHS Programmer's Guide</i> | SC31-8233 |
| <i>Tivoli for OS/390 NetView V1R3 Security Reference</i> | SC31-8606 |
| <i>Tivoli for OS/390 NetView V1R3 SNA Topology Manager and APPN Accounting Manager Implementation Guide</i> | SC31-8239 |
| <i>Tivoli Management Platform Reference Guide</i> | GC31-8324 |
| <i>TSO/E REXX/MVS User's Guide</i> | SC28-1882 |
| <i>TSO/E REXX/MVS Reference</i> | SC28-1883 |
| <i>VM/XA SP GCS Command and Macro Reference</i> | SC23-0433 |
| <i>VSE/SP Unattended Node Support</i> | SC33-6412 |
| <i>VTAM Messages and Codes</i> | SC31-6493 |
| <i>VTAM Network Implementation Guide</i> | SC31-6404 |
| <i>VTAM Network Implementation Guide</i> | SC31-6434 |

Part 1. Introducing OPC Automation

This part describes principal concepts of SA OS/390, including some NetView-related information, and gives an overview of the facilities offered by OPC Automation.

Chapter 1. Principal Concepts of SA OS/390

This section sketches some fundamentals of SA OS/390[®]. For more detailed information see the SA OS/390 documentation.

Automation Policies

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

All applications and systems that you want to include in automation must be defined to SA OS/390 in an automation *policy database*. This database contains the objects to be managed by SA OS/390, and the rules according to which automation of these objects proceeds. You access the policy database from the so-called *customization dialogs*. The customization dialogs are described in *System Automation for OS/390 Defining Automation Policy*.

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

Policy objects have automation-related properties and are associated with one another; these properties and connections are called *policy items*. For example, there is a policy item STARTUP for applications that specifies how SA OS/390 is to start the application.

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

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

The names of the resources have the following format:

resource_name/entry_type[/system_name]

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

The policy database must be converted into an *automation control file* (ACF) in order to be accessible to SA OS/390.

Goal-Driven Automation

A basic concept of SA OS/390 is to distinguish between the *desired* state of a resource and (broadly speaking) its *actual* state. Every resource has a desired state, which is either AVAILABLE or UNAVAILABLE; AVAILABLE is the default. This desired state, which is also called the automation *goal*, can be different from the actual state; a resource whose desired state is to be running (AVAILABLE), can actually be down. SA OS/390 always tries to keep the actual state in line with the desired state, but sometimes this is not possible.

SA OS/390 is called *goal driven* because all requests that can be made to it from the outside refer to the desired state of the target resource. When an operator passes a start request for a resource to SA OS/390, this is a request to set the desired state of the resource to AVAILABLE. It is up to SA OS/390 to decide whether (1) this is at all possible, and if so, whether (2) the actual state can be modified accordingly:

1. Making a request does not automatically lead to a change of the desired state of the target resource. Rather, SA OS/390 compares the *priority* of the new request with that of the last successful request. Only when the new request has a higher priority does SA OS/390 change the desired state of the resource. Note that this presupposes that the old request is still available. For more details on this topic, see “Persistence of Requests and Conflicting Requests” on page 6.
2. The latter decision mainly depends on the *dependencies* between the target resource and other resources, and on the *triggers* that may have been associated with it. Dependencies and triggers are defined in the policy database. For more information, see “Dependencies, Request Propagation, and Desired State”, and “Triggers” on page 7.

Dependencies, Request Propagation, and Desired State

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

Such dependencies can be specified in the policy database. The only restriction is that the dependent and the supporting resource must belong to the same sysplex (they need *not* reside on the same system). SA OS/390 takes dependencies into account when it is requested to start or to stop a resource. By default, it will try to start/stop all resources on which the target resource of the request directly or indirectly depends. The mechanism by which this is accomplished is called *request propagation*. It is best explained by an example.

Example 1: Let A, B, and C be resources so that A can only be started when B is running, and B can only be started when C is running. C is supposed

to have no start dependencies. Suppose, furthermore, that A, B, and C are all actually down, and that this conforms to their desired state (which is UNAVAILABLE).

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

This situation is displayed in Figure 1. The labels of the arrows specify the dependency type. **MakeAvailable/WhenAvailable** is the format in which SA OS/390 specifies that the dependent (lower) resource, which is referred to by **MakeAvailable**, can only be started when the supporting (upper) resource, referred to by **WhenAvailable**, is running.

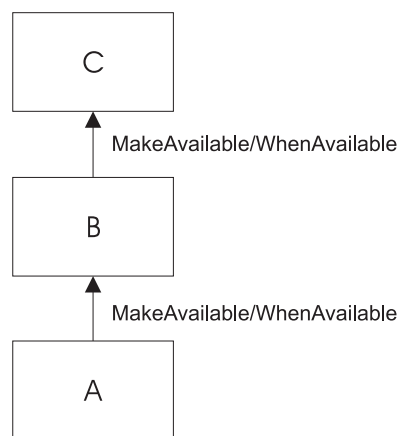


Figure 1. Example of Start Dependencies

When SA OS/390 receives a request to start A, the following chain of events will occur:

1. The request is propagated:
 - a. Since A can only be started when B is running, a start request is put to B.
 - b. Since B can only be started when C is running, a start request is put to C.
2. In response to these requests, the desired state of all three resources is changed to AVAILABLE.
3. SA OS/390 tries to change the actual state of the resources according to their desired state:
 - a. At first, only C, which has no start dependencies, can be started. B and A cannot be started because C and B are not yet running.
 - b. Then B will be started, because C is now available.
 - c. Finally, A is started.

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

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

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

To reflect this modification, the upper arrow label of Figure 1 on page 5 would have to be changed to **MakeAvailable/WhenDown**. This expresses that the dependent (lower) resource can only be started when the supporting (upper) resource is unavailable (down).

In example 2, the request must be transformed when propagated from B to C, because in order to start B and then A, C must be down. Therefore, SA OS/390 would put a *stop* request to C in this case, and the desired state of C would be set to UNAVAILABLE.

By propagating requests, SA OS/390 actively supports the start or stop request. You can also switch off request propagation for a resource. If this were to be done for resource A in example 1, then A would not be started because B is not available, and SA OS/390 would do nothing to start B. In this case A would only be started after B had been started, directly or indirectly, through another request.

Persistency of Requests and Conflicting Requests

Requests (and the votes derived from them) are persistent. They are stored in SA OS/390 and continue to be taken into account until you explicitly remove them. This implies that there can be more than one request (vote) for the same resource at the same time, and these requests (votes) can be contradictory, as shown in the following example.

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

Figure 2 contains a graphical presentation of example 3.

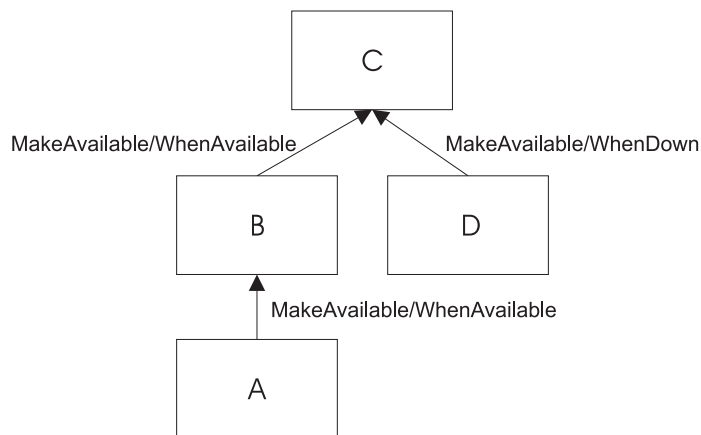


Figure 2. Example of Conflicting Requests

Now assume that first a request to start A and then a request to start D are passed to SA OS/390. The first request results in setting the desired state of C to AVAILABLE. Thereafter the propagation of the start request for D results in a vote to stop C. Since votes are persistent, the previous vote to start C is still existent,

and we have two contradictory votes for C. In such a situation, SA OS/390 uses the *priority* of the original requests to decide which one of the two wins.

When the priority of the old start vote for A is higher than that of the new vote to start D, then the desired state of D will be changed to AVAILABLE, but that of C will remain AVAILABLE; accordingly, SA OS/390 will not try to stop C, and thus D cannot be started. If, on the other hand, the vote to stop C has the higher priority, then the desired state of C is changed to UNAVAILABLE, and SA OS/390 will try to stop C in accordance with its desired state, and then to start D. When two contradictory votes have the same priority, a start vote wins over a stop vote.

The persistency concept implies that the losing vote is not automatically discarded. If, for instance, the start request for A wins, the start request for D and the propagated stop vote for C continue to be stored in SA OS/390, and can still be fulfilled after the request for A, and therefore also the start vote for C which was derived from it, have been removed by an operator. After the removal, SA OS/390 will determine the desired state of C again and will set it to UNAVAILABLE in response to the stop vote propagated from the start request for D, if no other vote is pending for C. After that, C will be stopped, and then D will be started.

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

Triggers

Triggers specify necessary conditions for starting or stopping an application. They are defined independently of applications. In this way the same trigger can be associated with more than one application. Triggers are defined and linked to an application in the policy database.

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

A trigger condition consists of a set of *events*. An SA OS/390 event represents an external event that is relevant to the state of the application associated with the trigger. The information that the external event has or has not occurred is passed to SA OS/390 by *setting* or *unsetting* the SA OS/390 event; this must be done by an operator or by an automation procedure. A trigger condition is only satisfied when *all* its events are set.

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

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

When the request to start A arrives at SA OS/390, it will set off the same sequence of events as with example 1 up to step 2 on page 5. Since, however, the only startup condition of the trigger is not satisfied, C will not be started, and therefore B and A will not be started either. In order to start A, EVENT2 must be set, for

example, by an operator. This will lead to a re-evaluation of the startup condition. Since this condition is now satisfied, SA OS/390 will start C, and subsequently B and A.

Service Periods

So far we have always assumed that the start or stop requests are made by a human operator. However, SA OS/390 also provides the possibility to make start and stop requests at specified points in time independently of human intervention. The objects that are able to do this are called *service periods*. Service periods are defined in the policy database.

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

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

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

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

Application Groups

Modern applications often consist of more than one component, and these different components can be distributed among different systems. SA OS/390 provides the possibility to combine different components of an application on one or more systems within a sysplex into an *application group*. This allows you to start and stop a complex application by a single command, and to integrate it into automation processes as a whole.

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

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

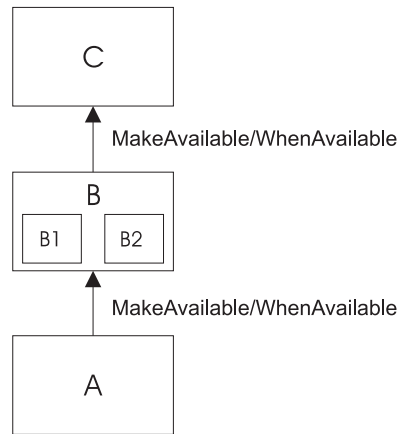


Figure 3. Example of a Request Involving a Group

Then, if you request A to be started, SA OS/390 will first, as before, propagate the request to group B and to application C. After C has been started and therefore group B can be started (step 3b on page 5 of example 1), a start vote will be propagated to every member of B. After the desired state of B1 and B2 has been set to AVAILABLE and both resources have been started, B will be considered available, and only then will SA OS/390 start A.

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

The group concept is also used to move applications from their primary system to a backup system when the primary system has failed (group type MOVE). In this case the members of the group are instances of the same application on different systems. In accordance with their purpose, MOVE groups are declared available when *exactly one* of their members is available. You assign preferences to the elements in order to determine which group member is to be started when a start request is put to the group, and which group member takes over when the currently available member is no longer restartable any more.

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

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

SA OS/390 and the NetView Message Automation Table

The implementation of SA OS/390 is based on NetView®. One important area, where SA OS/390 relies on NetView functionality, is the NetView Message Automation Table (MAT). This table serves to automate operator responses to messages that are sent to NetView. It contains instructions of the general form:

When message ABC arrives then issue command XYZ.

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

With applications controlled by SA OS/390, the command will typically be one of the generic routines that are shipped with SA OS/390 (see *System Automation for OS/390 Programmer's Reference*). Many of these routines retrieve information from the ACF and then act according to that information.

A typical example for such information is the MESSAGES/USER DATA policy item of the APPLICATION policy object. Within the MESSAGES/USER DATA policy item, you can associate a command with a message ID (see *System Automation for OS/390 Defining Automation Policy*). If you connect this message ID with the generic routine ISSUECMD in the MAT, then NetView will execute ISSUECMD when the application sends the message in question to NetView. ISSUECMD, in its turn, will search for the message ID in the ACF entry for this application, and if the message ID is associated there with a command, it will issue this command. For more information on ISSUECMD, see *System Automation for OS/390 Programmer's Reference*.

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

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

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

For more information on the MAT, see *Tivoli NetView for OS/390 Automation Guide*. OPC Automation also has some special generic routines, see "Chapter 7. OPC Automation Common Routines and Data Areas" on page 81.

Chapter 2. Functions of OPC Automation

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

Basic Concept of OPC Automation

OPC Automation is an extension of SA OS/390 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.

The strengths of OPC as compared with SA OS/390 are the following:

- OPC is able to control multiple streams of related activities without logical limits on the number of streams or the time required to execute those streams. With dependency control, OPC can separate each stream into many parallel substreams and, at a later time, bring them back together into a smaller number of streams or a single stream. During this processing, sophisticated mechanisms detect deviations and either correct them or bring them to the attention of the administrator.

On the other hand, SA OS/390 is not designed to manage complex streams of related operations in the same way that OPC is. SA OS/390 manages simple streams, such as single sets of procedures, by keeping control data in variables. This methodology becomes cumbersome when the streams are complex.

- OPC has very sophisticated scheduling capabilities and can manage time units such as months or years. Thus, you can take into account events such as holidays or financial quarter-end processing.

On the other hand, the maximum time unit that can be managed with the service periods of SA OS/390 is a week.

SA OS/390 has the following strengths as compared to OPC:

- SA OS/390 interacts with MVS™ console services and automates many of the functions that would otherwise have to be performed by console operators. OPC, on the other hand, lacks the ability to interact with console services. Although you can issue commands to the system console, OPC cannot receive responses and evaluate them. OPC can ask for an action, but cannot determine if the action is completed successfully or if problems are encountered.
- In contrast to OPC, SA OS/390 is able to perform complex startup, shutdown and recovery activities.

These two lists give an indication as to what the basic idea of OPC Automation is, namely to transfer the management of functions that are very complex or require consideration of time units greater than a week from SA OS/390 to OPC, but to leave to SA OS/390 any communication with console services or startup, shutdown, and recovery operations required for these functions.

The implementation of OPC Automation requires the introduction of new objects in OPC and in SA OS/390.

Defining SA OS/390 to OPC

On the OPC side, the concept of a workstation is extended. 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 a workstation to include NetView by allowing the definition of automatic reporting general workstations.

This is a class of workstations that OPC can manage, but which is outside OPC's direct control. With the implementation of this workstation class, NetView can become a server to OPC. The automatic reporting general workstations that represent a NetView domain must have the name NVxx. 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.

Each SA OS/390 NetView in your enterprise is represented by one OPC workstation. An OPC workstation may also represent all NetViews running SA OS/390 within the same sysplex where the OPC Controller is running. These NetView workstations then schedule and perform operations on behalf of batch applications.

Defining OPC and OPC-Controlled Subsystems to SA OS/390

On the SA OS/390 side, several new policy objects and new policy items for existing objects are introduced. These serve to

- associate OPC workstations with NetView domains:
This association is established through the WORKSTATION DOMAINS policy object (entry type ODM). See *System Automation for OS/390 Defining Automation Policy*.
- define OPC to SA OS/390:
This includes defining the OPC controller and tracker as applications of type OPC to SA OS/390 and specifying further information about the OPC controller (entry type OCS). See *System Automation for OS/390 Defining Automation Policy*.
- specify the applications that are to be controlled by OPC Automation, and specify the commands that are to be issued for these applications in order to fulfill a request from OPC.

The first of these tasks is implicitly performed by specifying the OPC related commands for the application. For that, the OPC-specific MESSAGES/USER DATA policy items OPCA, OPCACMD, and OPCAPARM must be used. See "Chapter 6. MESSAGES/USER DATA Entries and USER E-T Pairs for OPC Automation" on page 65.

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, SA OS/390 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 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 start up and shut down of subsystems independently of automation status changes.
 - UXxxxxxx 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.

Request and Confirmation Transaction Flow

Figure 4 on page 14 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 4 on page 14, the jobname is specified as RMF and the operation text is START.

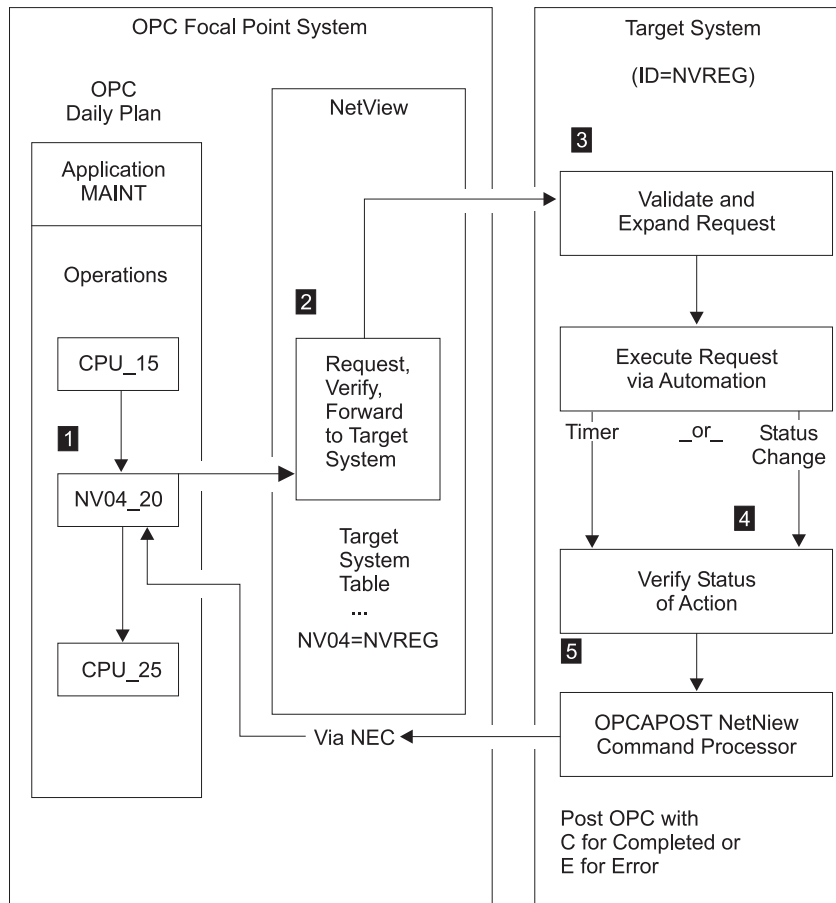


Figure 4. NetView–OPC Interface Flow. Syntax and definition errors, target system availability, recovery, and resynchronization via OPC API and NetView PPI are not shown in this example.

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

Figure 4 shows CPU_15 as the last batch step which needs processing prior to starting RMF. Once this completes properly, OPC dependency control makes the NV04_20 operation ready on the NV04 workstation. This causes the creation of a request buffer for the request to start RMF which is then forwarded to NetView Domain NVREG. For the request buffer, see “Request Buffers and OPC Automation Log Entries” on page 15.

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

1 The NetView PPI passes the request buffer to the PPI dispatcher task in SA 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 policy database.

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

3 The target NetView translates the request text into MVS console commands using information stored in the SA OS/390 policy database. In Figure 4 on page 14, the request module translates the request buffer into a command to start RMF. This start command must be specified with the OPCACMD keyword in the MESSAGES/USER DATA policy item of the RMF application (for details, see “OPC Requests and MESSAGES/USER DATA Keywords” on page 57). It can be an MVS START command, or the INGREQ SA OS/390 command (which is recommended). In the latter case, the command could be:

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

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

The command is dynamically generated using definitions in the SA OS/390 policy database. A check determines whether the command is properly accepted. During this process, WTOs and the OPCAPOST command report errors. OPCAPOST sends an error indication back to OPC. If the command is issued correctly, a timer request is made. The timer intercepts a condition, where the request does not execute in a reasonable amount of time, which is user-selectable.

4 A change-of-status SA 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 which passes the completion code to the OPC Tracker on this system. The OPC Tracker forwards the completion code to the OPC Controller.

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

This terminates the processing for this specific OPC operation. If the request is executed without problems, the operations status is set to C (completed), and normal OPC dependency control allows the next operation to start. See the CPU_25 batch job in Figure 4 on page 14.

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:

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

Request Buffers and OPC Automation Log Entries

Requests are specified to OPC Automation in the **Operation text** field of the **Operations** panel (see “Defining Applications for OPC Automation in OPC” on page 50 for details). The EQQUX007 exit logic creates a request buffer that contains the request and all additional information that is needed

- To pass the request to its proper destination, and

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

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

Figure 5 shows the request buffer log entry for the example of Figure 4 on page 14, if we suppose that an OPC controller running on domain NVDOM has made the request to OPC Automation to perform START for the RMF subsystem. When this action is completed, OPC Automation changes the status of the NV04_20 operation in the MAINT application to C (completed).

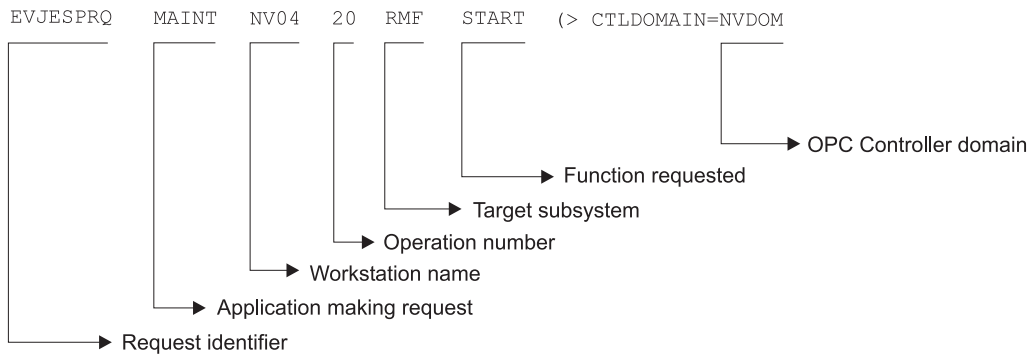


Figure 5. NetView Log Entry of an OPC Generated Request

The operation number is needed in order to inform OPC Automation which operation in OPC must be notified of the result of the request. In the example, the request contains no parameters. When a request does contain parameters they are inserted in the log entry between the request type (function) and the domain name of the requesting OPC controller. For more information on request parameters, see “Request Parameters and the &EHKVARi Variables” on page 59.

OPC Automation Special Resources

OPC Automation is able to globally control the creation and setting of OPC special resources based on the status of SA OS/390 monitored subsystems. This will allow OPC to resolve job scheduling dependencies based on the status of SA OS/390 managed resources.

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

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

where:

- | | |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| res_sys | is the MVS sysid of the system where the subsystem status change was detected. If the resource is a SYSPLEX application group, then the value SYSPLEX is used. |
| res_type | is one of APL, APG, SYS, SYG, or GRP. |
| res_name | is the name of the resource. |
| UP | is a literal that defines the resource as being available only when the resource has an observed status of AVAILABLE and a desired status of AVAILABLE. |

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

SOFTDOWN, HARDDOWN, STANDBY, UNKNOWN, SYSGONE,
and when its desired status is UNAVAILABLE.

Possible Uses of OPC Automation

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

- Controlling online systems
- Processing all batch work

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

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

OPC Automation allows changes which influence both batch and online systems through simple OPC dialogs. Because OPC manages both batch and online systems, these changes are needed only in one place. Because the processes are automated with SA OS/390 and OPC, no interoperator communications are required. In fact, in a highly automated environment, no operator intervention or awareness of these user-requested changes is necessary.

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

Changing Online Hours of Availability

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

- The help desk gets a request from a user group to extend the IMS hours of availability, for today only, from the original plan of 6 p.m. to 8 p.m. (extended service period).
- The help desk ensures that this extended service is acceptable within the service level agreement for this user group.
- The help desk now makes a change to the OPC current plan to reflect this extended IMS period.
- When the revised scheduled time is reached, now two hours later than usual, OPC executes the operation, requesting that OPC Automation stop IMS.
- OPC Automation requests that SA OS/390 application stops IMS.

- Once SA OS/390 has successfully stopped IMS, OPC Automation returns an operation-ended status to OPC, fulfilling OPC's dependencies on the online IMS.
- Jobs dependent on the termination of IMS are now released for execution.

The above scenario requires no intervention from the IMS/MTO or system console operators. No restructuring of the batch processing is necessary if the request is within planned service bounds.

Cycling Individual Online Databases

OPC Automation allows OPC to interact not only with the SA OS/390 functions, but also with the MVS and MTO consoles, which enables the scheduling of interrelated sequences of events. For example, it is possible to cycle individual databases rather than the complete online system. Figure 6 shows how this scheduling results in minimum disruptions to online applications.

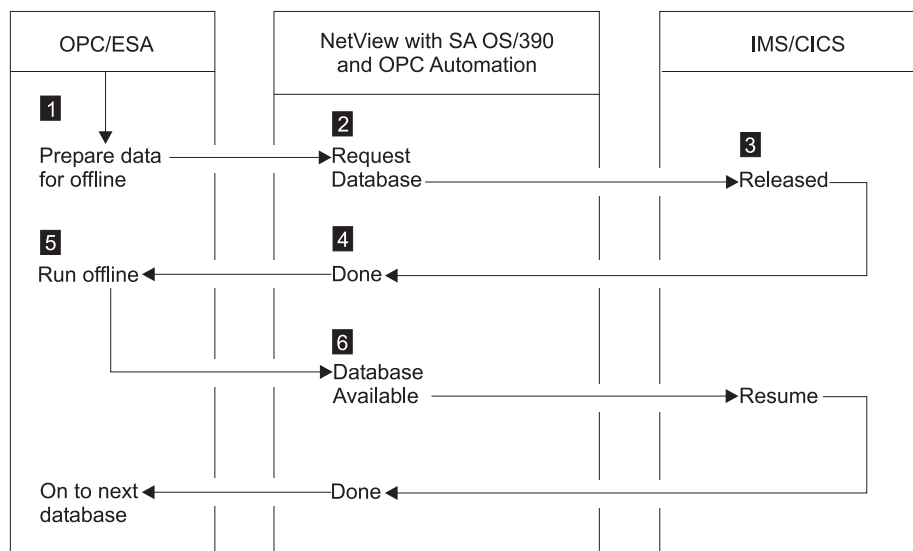


Figure 6. Example of Cycling Individual Online Databases

In Figure 6, the online databases are structured so that you can vary specific ones offline, without an impact to the system, as in the case of databases structured on a geographic or application basis. This process flows as follows:

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

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

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

OPC Automation does not directly support the preceding example, which requires some user-written modules. (See “Interaction with CICS Automation” on page 103 and “Interaction with IMS Automation” on page 104 for some examples of this type of user-written module.) OPC Automation transports the request to the appropriate system and prepares the information for the user code. OPC Automation then returns the resulting status to the operation in OPC. OPC Automation also ensures that the actions requested are serialized with other requests to that specific target subsystem and that the status of the subsystem is such that it can accept the requests.

Scheduling Time for Testing

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

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

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

Distributing and Updating Data Across Multiple Systems

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

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

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

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

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

1. A PTF is installed, tested, and found acceptable. This PTF is then applied to all copies of TSO in a multisystem environment.
2. The application is defined to OPC. In most cases, the application is simply updated since it is already defined.
3. OPC presents the batch jobs that control the SMP/E process to the systems programmer for modifications, if required.
4. OPC schedules the transmission of the jobs to the target systems using NJE. The scheduling can use a time when network traffic is low.
5. Once the jobs are in the target system, OPC dependency control is used to schedule the SMP/E job execution.
6. OPC ensures that the SMP/E jobs run correctly. If OPC encounters problems, the OPC application provides backout procedures.
7. After installing the PTFs, OPC selects the appropriate time to issue a request to OPC Automation to restart TSO.
8. Since OPC fully controls the process for this PTF update, you can inquire at any time to see the progress of the operations. If errors or problems occur, OPC Automation informs the SA OS/390 notification operator.

Complex Application Recovery

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

OPC Automation allows full or partial automation of this type of activity between systems and sites. In some cases, changing the NVxx-to-NetView domain ID relationship is adequate to transfer the control of the work load to a different system. However, the change may require some manual intervention for synchronization. "Chapter 11. Resynchronization and Recovery Considerations" on page 137 discusses several scenarios and the process of synchronization.

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

Step 1: Preparing the recovery system

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

Step 2: Starting the critical applications on the recovery system.

This can include the following:

- Loading databases and applications from tape-to-disk devices
- Starting the recovery system
- Updating data from checkpoint data, logs, or other sources
- Starting critical applications.

Step 3: Returning to the original production system

This is a reversal of the recovery process. These procedures are as complex as the original recovery process, but are scheduled and do not have the urgency of the original recovery.

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

- Prior to the need, a series of interdependent recovery applications are defined to OPC, but not scheduled.
- The decision to recover the critical applications at the backup site is made. The scheduler uses normal OPC panels to modify the current plan to schedule the first backup application.
- Before recovery, several factors, which can result in modifications to procedures and JCL, need considering. These modifications are then presented to operators at manual workstations with instructions in the operator instruction files of OPC. They are also presented to systems programmers at JCL workstations.
- The work load on the recovery system is stopped by scheduling a request to SA OS/390 to stop all subsystems other than JES.
- Once the subsystems are stopped, a series of jobs are scheduled to transfer data from disk-to-tape to accommodate the requirements of the critical applications that are recovered.
- Depending on the situation, the same system is reused or restructured, and then followed by an IML and IPL of the recovery system. If this is the case, the focal point implementation option of SA OS/390 is used to partially or completely automate this phase of the recovery. Regardless of the specifics, the result is an operating system platform ready to accept the recovery environment.
- OPC schedules a series of JES jobs that restore the databases from backup.
- OPC triggers NetView to issue the appropriate commands to start the subsystem.
- In some cases, NetView requires access to MTO functions to issue specific procedures before the DB/DC system can resume transaction processing. If that is the case, user-provided modules are required to fully automate the recovery.

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

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

Give special consideration to any synchronization procedure in an OPC Automation environment. For more information on the synchronization process, see "Chapter 11. Resynchronization and Recovery Considerations" on page 137.

Chapter 3. Structure of OPC Automation

This chapter explains the structure of OPC Automation in some detail.

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 section provides an introduction to these components and their interactions.

Initialization

Initialization involves the following two sequences:

1. Initialization of the OPC components.
2. Initialization of OPC Automation functions in each NetView. "Startup of OPC-Controlled Subsystems" on page 32 describes this. OPC Automation initialization includes the automated recovery sequences described in "Automated Recovery" on page 38.

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 13 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.

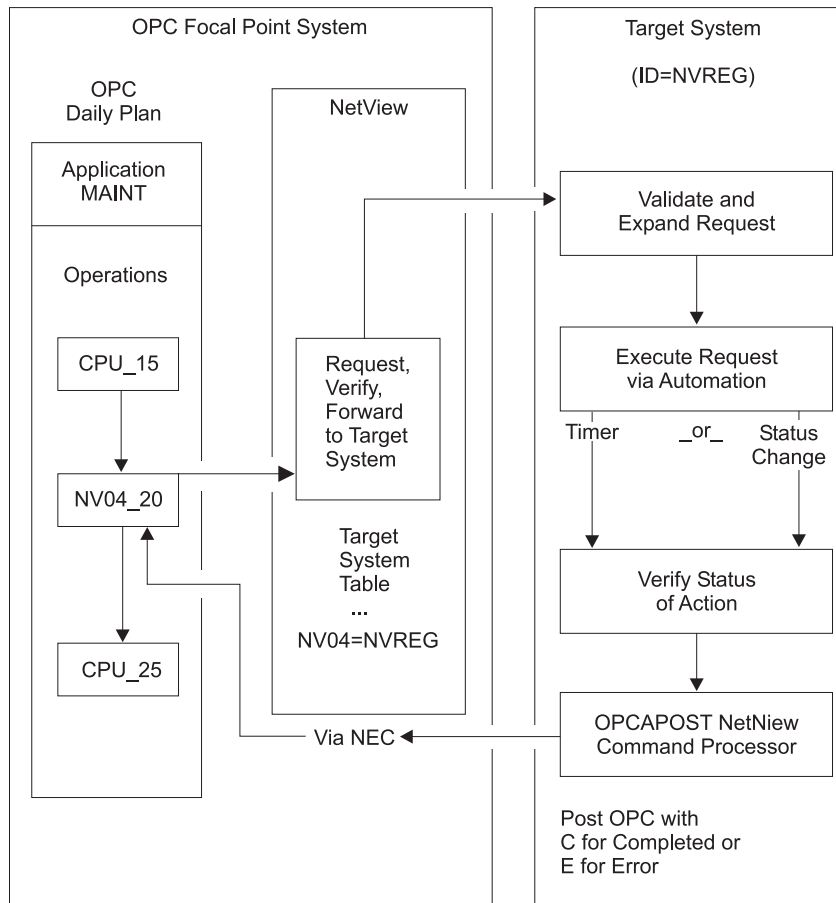


Figure 7. NetView-OPC Interface Flow. Syntax and definition errors, target system availability, recovery, and resynchronization via OPC API and NetView program-to-program interface are not shown in this example.

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. NVxx specifies a NetView automatic general workstation with a NetView domain index of xx, which is resolved in the Controller NetView into the target NetView domain ID through the definitions in the SA OS/390 policy database. OPC can define the NVxx 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 EQQUX007 exit in the OPC/ESA controller.

EQQUX007 Exit

Each change of status on any workstation calls the EQQUX007 exit, which checks for a NVxx workstation proceeding to the R (ready) status. The EQQUX007 exit

ignores all other conditions (operations going to A or * are counted as ready). Figure 8 shows the flow of this process.

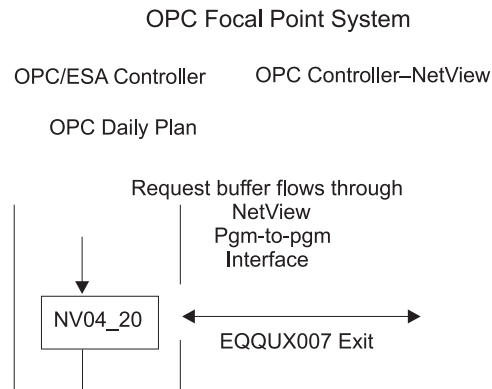


Figure 8. EQQUX007 Exit

When an NV xx workstation moves to the R status, the workstation generates a request buffer. Fields pointed to by registers in the EQQUX007 exit provide all of the data for the request buffer. For the layout of the fields in the request buffer, see Table 5 on page 95 and Table 6 on page 95.

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

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

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

Program-to-Program (PPI) Interface Dispatcher

The NetView program-to-program interface passes the request buffer to the PPI dispatcher task in the SA OS/390 application. The PPI dispatcher task (EVJTOPPI), a NetView subtask, receives the requests for an SA OS/390 action from the buffers from the EQQUX007 exit. Figure 9 on page 26 shows this flow.

OPC Focal Point System

OPC/ESA Controller

OPC Controller NetView

OPC Daily Plan

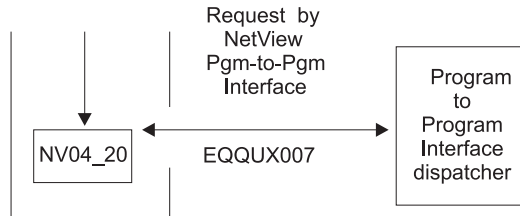


Figure 9. PPI Dispatcher

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 NetView. This module receives the action request buffer from the PPI dispatcher task. Figure 10 shows this process.

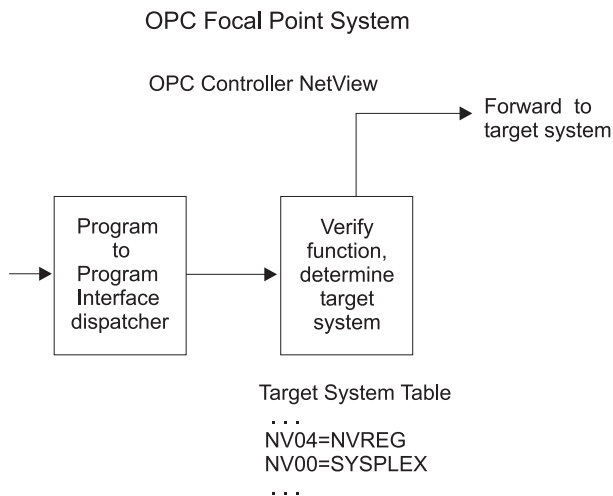


Figure 10. Verify Module

The verify module uses the NVxx index to obtain the destination NetView domain ID from the SA OS/390 policy database. If the relevant NVxx index specifies SYSPLEX, then all SA OS/390 systems in the local sysplex are queried for the status of the application associated with the job name of the request. The destination is determined to be the system which has the application in the most active state.

If the destination NetView and the requesting NetView are the same, OPC Automation logs the request buffer and invokes the request module. If the destination NetView and the requesting NetView are different, OPC Automation sends the request to the proper NetView domain by message forwarding.

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

If NVxx is associated with the sysplex on which the OPC controller is running (SYSPLEX keyword in the WORKSTATIONS DOMAIN entry) and OPC Automation does not find the job defined to any online SA OS/390 in the local sysplex, then OPC Automation issues a message, posts the operation status to E (ended-in-error, S998), and logs the results. To cater for the situation where all domains where the job runs are offline, the operation will be retried if a gateway connection to another SA OS/390 becomes active.

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

Request Module (EVJESPRQ)

The arrival of a request from the verify module drives the request module in the Tracker NetView. OPC Automation installs the request module on each system running an OPC/ESA Tracker. Figure 11 on page 28 shows the flow of this process.

The main function of the request module is to translate the OPC-generated request into a subsystem related command, or to schedule a user-defined function that is not related to a subsystem. The control flow of the module is shown in Figure 11 on page 28.

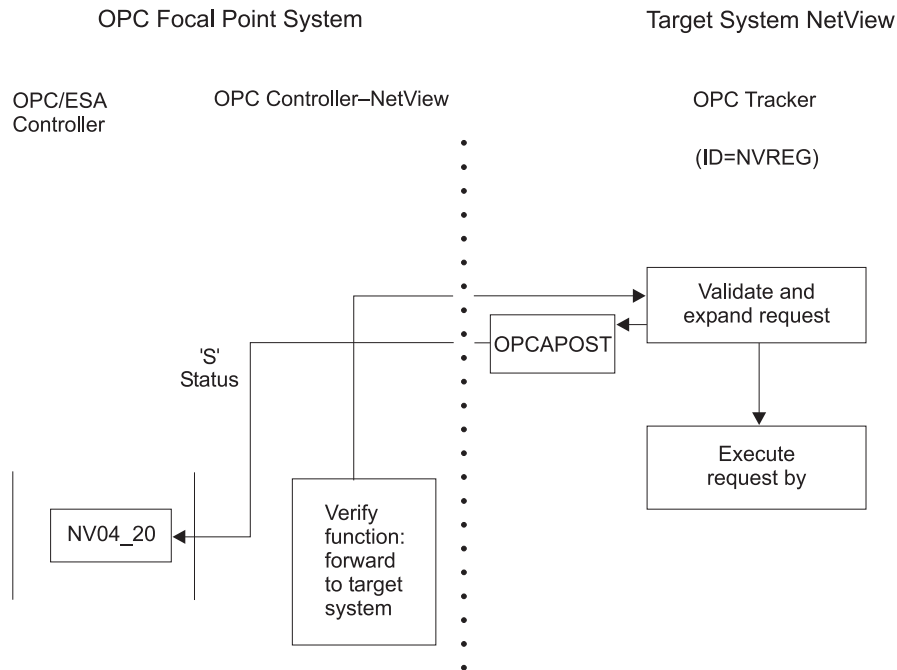


Figure 11. Request Module

If required, the request module uses definitions in the SA OS/390 policy database to create the command that initiates the function requested. The policy database contains entries (OPCA, OPCACMD keywords; see “Chapter 6. MESSAGES/USER DATA Entries and USER E-T Pairs for OPC Automation” on page 65) from which the command text and the parameter syntax for the actual request are obtained. If any of these entries are not found, the processing cannot continue. The OPCAPOST module posts an error to OPC, which logs the error and issues a WTO. Since this is a user-definition error, OPC attempts no automation recovery. The user must correct the definitions and reset the operations in error.

In Figure 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 “Chapter 8. Guidelines for User-Written Operations” on page 97.

For subsystem-related operations, the OPCAPOST command processor posts to OPC if the required entries are found in the policy database. OPC changes the status from R (ready) to S (started). OPC Automation then issues a timer request on the basis of the delay specified in the policy database (OPCA keyword, see “OPCA” on page 71), issues the command, and checks the return code. Then the request module terminates.

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

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

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

Status Change Module (EVJESPSC)

SA 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 request for the changed subsystem exists, and the new status is compliant with the expected status, OPC Automation cancels the timer. OPCPOST 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.

Target System NetView

(OPC Tracker–NetView)

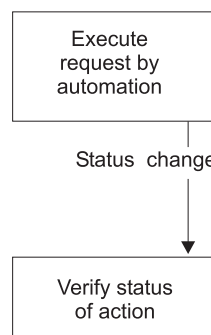


Figure 12. Status Change Module

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 Automation 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.

Target System NetView

(OPC Tracker–NetView)

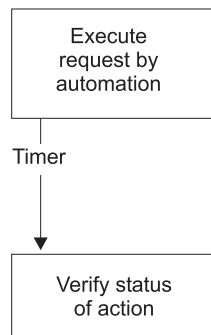


Figure 13. Timer Module

For subsystem related functions, the SA OS/390 status file provides the current status, and EVJESPTE compares this with the expected status. If a match is obtained, the OPCAPOST command processor posts a C (completed) status to OPC. If EVJESPTE determines a mismatch between the current and expected status, OPCAPOST posts an error to OPC for review by the OPC administrator. Figure 13 shows the flow of this process.

OPCAPOST Command Processor

The OPCAPOST command processor calls EQQUSINT, which passes the completion code to the OPC Tracker in this system. The OPC Tracker forwards the completion code to the system running the Controller through the mechanism used by OPC/ESA. Figure 14 shows the flow of this process.

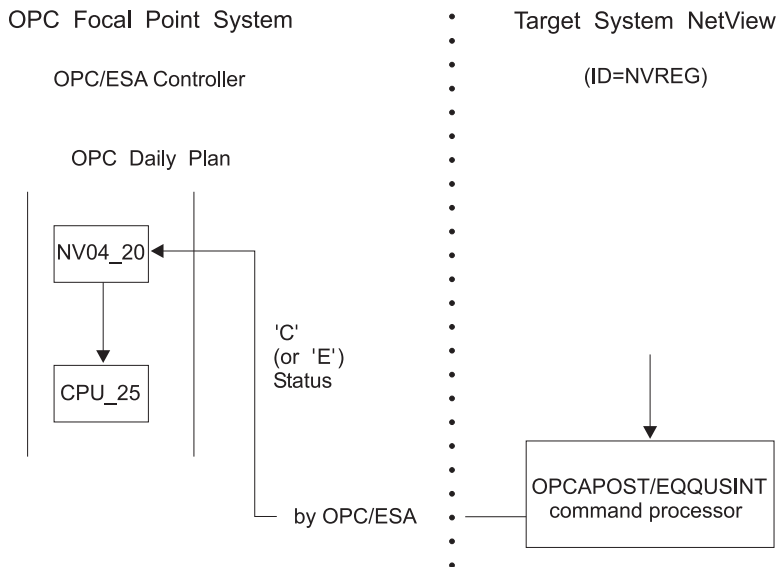


Figure 14. OPCAPOST Command Processor

Other functions use the OPCAPOST command. See "OPCAPOST" on page 92 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 on page 30. If the operation completes in error, OPC Automation sets an E status and a 4-character return code. The application does not continue processing until some intervention occurs. An operator or OPC Automation's recovery can sometimes provide this intervention.

Automated Operator Tasks

OPC is an SA OS/390-controlled subsystem. Normal definitions in the SA OS/390 policy database can describe OPC. In addition, SA OS/390 defines an automated operator task (called *automated function* by SA OS/390) 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 a proper sequence of processing is maintained, you must complete the actions in a single-thread fashion. In OPC, this is the responsibility of the user and is achieved through dependency control or critical resource specifications.

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

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

For the automated operator task 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.

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.

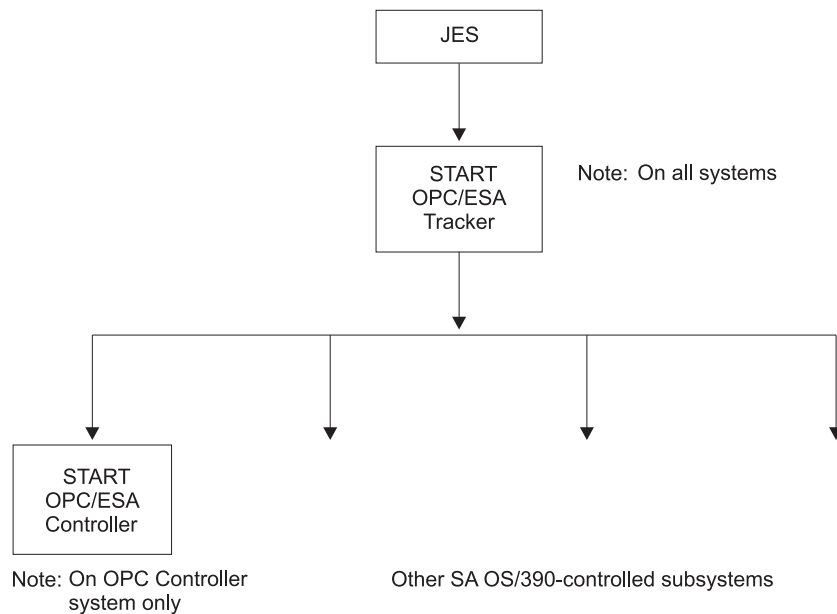


Figure 15. OPC/ESA Startup During IPL Process

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 has completed initializing its defined tasks, the startup phase of the OPC-controlled subsystems starts.

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

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

During OPC Automation startup, OPC Automation examines the SA OS/390 database for OPC Automation entries. If new entries are found, OPC Automation creates status file records and initializes them to a null value (never a used status). OPC Automation attempts no action for these subsystems until it receives a request from OPC. This allows coding entries into the policy database before defining the subsystems in the rest of SA 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 “EVJESPIN—Initialization” on page 136.

Request Handling in the OPC Controller System

In an OPC-controlled application, defining specific parameters for an operation generates a request. These parameters are defined for an operation on an NV xx workstation, where NV xx represents a NetView domain. When the daily planned execution of OPC makes this NV xx workstation ready, OPC Automation starts through the EQQUX007 OPC/ESA user exit. See Figure 16 on page 34 for an illustration of this flow.

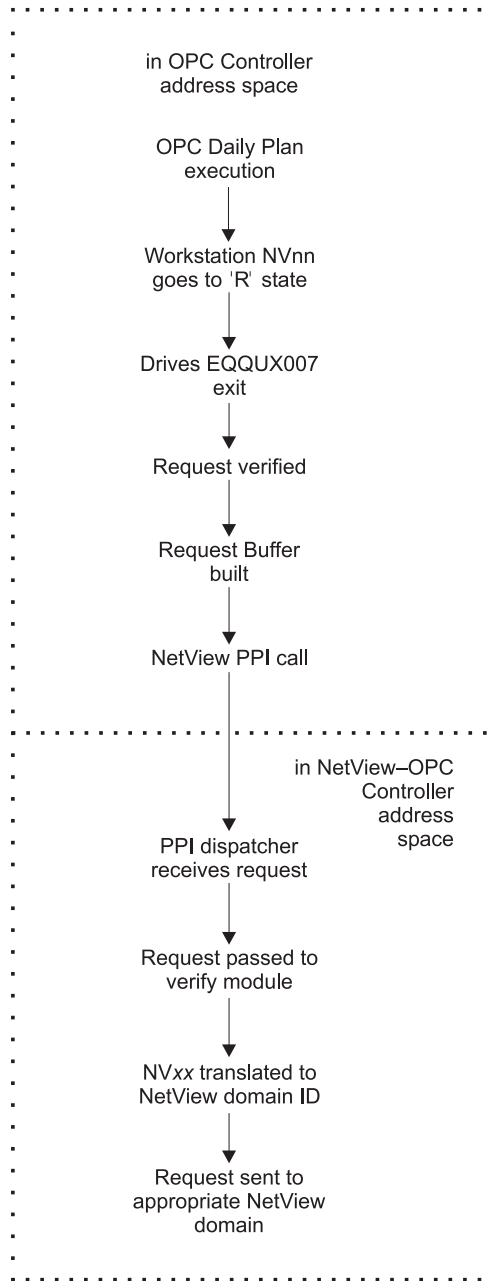


Figure 16. Request Handling in the OPC Controller Processor

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

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

If OPC Automation is unable to call the NetView interface module, it marks the operation with a status of E and an error code of UNTV. OPC Automation will tell OPC to reset operations which have ended with a UNTV error every time OPC or SA OS/390 is restarted, provided these errors occurred less than a user-specified time interval. This time interval is defined in the **Operation reset delay** field of the OPC SYSTEM DETAILS policy object; see *System Automation for OS/390 Defining Automation Policy*.

The verify module translates the NVxx identifier 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 the OPC Controller is on, OPC Automation transfers the request to the request module running under the NetView Tracker automated operator task.

Request Handling in the OPC Tracker System

Figure 17 describes the flow of OPC Automation for a request invoking a base function of the SA OS/390-defined subsystem.

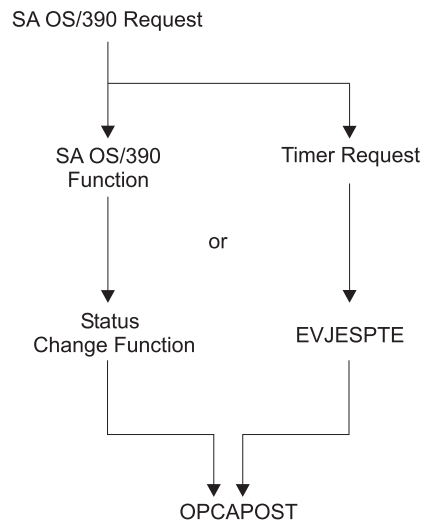


Figure 17. Request Flow for a Base SA OS/390 Function

The request module (EVJESPRQ), a part of the Tracker portion of OPC Automation, runs under the NetView Tracker automated operator task. This module issues the command associated with the request that is contained in the request buffer. The OPCACMD entry for each subsystem (see “OPC Requests and MESSAGES/USER DATA Keywords” on page 57) defines the actual command. This permits the coding of generic requests in the application descriptions in OPC. This request flow also allows customizing requests for each system and avoids changes in the target systems reflected in the focal-point system.

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

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

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

- RESET
- SYNC

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

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

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

Completion and Timer Flags

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

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

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

Operations Control

This section discusses the EVJESPIN module and obtaining information from OPC. "EVJESPIN—Initialization" on page 136 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/ESA. |
| SYNC | The OPC Automation status file record contains the status of the subsystem that was the result of the last successful request; this status is either UP or CTLDOWN. With the SYNCH option, EVJESPIN tries to synchronize the actual status of the subsystem with this information. When both are at variance, OPC Automation issues a SA OS/390 startup or shutdown command to change the status of the subsystem to match that in the status file record. The SYNC function does not post to OPC on completion of the SA OS/390 function. OPC's requests for the subsystem synchronize SA OS/390. |

If you have defined a new subsystem to SA OS/390 since last initializing NetView, you can create a single OPC Automation status file record. The (CREATE) option creates an OPC Automation status file record dynamically for the specified subsystem. OPC Automation initializes the record to a null condition, so that the subsystem can accept OPC Automation commands.

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

Obtaining Information from OPC

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

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

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

The host with the OPC Controller task provides the only availability to the OPC API interface. Because of this, all recovery is done only in the NetView on the processor running the OPC Controller task. Sometimes, when a user-written module utilizes this function, another NetView requires the information. OPC Automation maintains an entry in the policy database to allow OPC Automation to determine the domain ID of the OPC NetView to which the request is sent (CONTROLLER DETAILS policy object, OCS entry type). This entry has the domain ID of the NetView on the processor running the OPC Controller. In this manner, all the SA OS/390 applications can easily determine where to direct OPC Controller requests.

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

- “OPCALIST” on page 87,
- “OPCAMOD” on page 89,
- “OPCACAL” on page 83,
- “OPCACOMP” on page 86,
- “OPCAPOST” on page 92,
- “OPCSRST” on page 93.

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

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

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

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

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

When either of these two conditions occur, OPC Automation uses the OPC API function to obtain a list of all operations ended-in-error for the appropriate NVxx 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 exit. OPC Automation attempts no other recovery.

Part 2. Customizing OPC Automation

This part describes the steps that are necessary to customize and set up OPC Automation. Furthermore, it contains reference sections for OPC-specific MESSAGES/USER DATA keywords and for common routines which request information or perform tasks associated with OPC Automation.

Chapter 4. Customizing OPC Automation

This chapter explains how to customize NetView and SA OS/390 for OPC Automation.

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 through Release 3, program number 5687-OPC.
- Operations Planning and Control/Enterprise Systems Architecture (OPC/ESA) Version 1 Release 2 or higher, program number 5696-007

For OPC Automation sysplex-related functions (for example, 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
- SA OS/390 Version 2 Release 1
- The OPC Controller must be running in a sysplex environment

Installation Related Steps

Complete the steps listed below for installation.

Step 1: Updating MPFLSTxx

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

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

Step 2: Add Libraries to OPC and Recycle

Add your SINGMOD1 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 module EQQUX007. If you are using an existing exit 7, you can combine this exit with OPC Automation-supplied modules. See “Step 2: Integrate Existing Exit 7 with OPC Automation” on page 46 for details.

You must specify the CALL07(YES) parameter in the OPC/ESA initialization parameters.

Other initialization parameters must be specified in the OPC initialization member (EQPPARM) so that OPC will issue some of its messages to the MVS console. The DURATION, ERROROPER, LATEOPER, and OPCERROR messages are automated by OPC Automation. The RESCONT and QLIMEXCEED messages are useful for further customer automation.

You must specify the following in EQPPARM:

```
ALERTS WTO (DURATION
            ERROROPER
            LATEOPER
            RESCONT
            OPCERROR
            QLIMEXCEED)
```

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.

| Tivoli OPC V2 | | OPC/ESA | |
|---------------|--------|----------|--------|
| Message | Member | Message | Member |
| EQQW065I | EQQW06 | EQQW065I | EQQW06 |
| EQQW011I | EQQW01 | EQQW011I | EQQW01 |
| EQQN013I | EQQN01 | EQQN013I | EQQN01 |
| EQQZ086I | EQQZ08 | EQQZ086I | EQQZ08 |
| EQQE026I | EQQE02 | EQQE026I | EQQE02 |
| EQQE036I | EQQE03 | EQQE036I | EQQE03 |
| EQQZ128I | EQQZ12 | EQQFCC1I | EQQFCC |
| EQQZ201I | EQQZ20 | EQQPH00I | EQQPH0 |

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 Customization and Tuning* and *OPC/ESA Installation and Customization*.

NetView Related Steps

This chapter describes steps that must be performed in NetView.

Step 1: Add OPC/ESA Data Sets and Allocate EQQMLOG

Add JCL for OPC/ESA data sets used by NetView as shown:

```
/*
/* OPC/ESA MISC DATA SETS
/*
//EQQMLOG DD DSN=ING.V2R1M0.A0F01.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 data set 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.

Step 2: Check the NetView Message Automation Table for Conflicts

Make sure that no conflicts exist between OPC Automation-related messages and other messages in your NetView message automation table. There could be two possible areas of conflict:

- If you are currently using a customized NetView automation table and it contains messages prefixed with EQQ or CSY, these could be in conflict with OPC Automation message table entries in member EVJMOPCE. Document and resolve any conflicts.
- The EVJMCON1 member contains additional messages that may have conflicts with your NetView message automation table. These messages 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 message automation table. Refer to the *NetView Administration Reference* manual for additional information.

OPC Automation Initial Customization

This chapter describes the definitions that occur in NetView.

Step 1: Basic OPC Automation Common Policy Definitions

For each NetView domain, set up the OPC 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. Define the OPC Automation automation operators in the AUTO OPERATORS policy object (AOP entry type) according to the following table:

| Automated Function | Operator ID | Message Classes |
|--------------------|-------------|-----------------|
| OPCAOPR1 | AUTOPCP | CSY*,EQQ*,EVJ* |

| Automated Function | Operator ID | Message Classes |
|--------------------|-------------|-----------------|
| OPCAOPR2 | AUTPCE | CSY*,EQQ*,EVJ* |

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

3. Define the OPC/ESA controller, the OPC/ESA tracker, the OPC/ESA server (if required for a SYSPLEX), and the OPC/ESA data server (if required) in the SA OS/390 customization dialogs as applications of type OPC with the appropriate subtypes. See *System Automation for OS/390 Defining Automation Policy* for more information.

The recommendation for a sysplex is to have a separate tracker on each system in the sysplex. At least two controllers should be defined, each running on a different system in the sysplex. The server should therefore be defined to SA OS/390 as a child of the controller and started and stopped by the controller. Only one data server is required per JES Spool, so the data server should be defined to run on one system within the MAS. In the case of the server, tracker and data server, the appropriate ARM element name should be defined in the SA OS/390 customization dialogs. There are examples of these definitions provided in the samples.

4. Customize the WORKSTATION DOMAINS (ODM entry type) policy objects in the SA OS/390 policy database and connect them to systems in the WORKSTATION DOMAINS policy item as required. These policy objects map OPC workstations to NetView domain IDs.
5. Customize the CONTROLLER DETAILS (OCS entry type) policy objects in the SA OS/390 policy database and connect them to systems in the CONTROLLER DETAILS policy item as required. These objects specify the location of a controller and can be associated with a set of OPC special resources. See *System Automation for OS/390 Defining Automation Policy* for more information.
6. Customize the OPC SYSTEM DETAILS policy objects (OEN entry type) in the SA OS/390 policy database and connect them to systems in the OPC SYSTEM DETAILS policy item as required. These objects contain control information for OPC Automation. See *System Automation for OS/390 Defining Automation Policy* for more details.
7. If required, define OPC special resources as OPC SPECIAL RESOURCES policy objects (OSR entry type) in the SA OS/390 policy database and link them to CONTROLLER DETAILS objects. See *System Automation for OS/390 Defining Automation Policy* for more information.
8. You must define each subsystem you wish to automate from OPC to SA OS/390. In many cases, you will also have to define OPCA and OPCACMD entries in the MESSAGES/USER DATA policy item for these subsystems (see "Executing OPC Requests with OPC Automation" on page 56).

Step 2: Integrate Existing Exit 7 with OPC Automation

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

- EQQUX007
- UX007001
- UX007002
- UX007003

EQQUX007 is the exit driver program. It calls other modules in turn, as if OPC is calling each module directly. The driver searches for UX007001 through UX007010. UX007001 through UX007003 are supplied with OPC Automation. If you have an existing exit 7, rename your module from EQQUX007 to UX007004. The called routines are passed the same parameters the call to 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 UX007005. 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 3: Initializing the OPC Automation Status File

To initialize a status file from NetView for OPC, enter the command:

```
EVJESPIN CMD=INIT
```

Do not use this command until you have created the OPCA 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 entries in the policy database as necessary. Additional details and typical OPC definitions are shown in "Chapter 5. Setting Up OPC Automation" on page 49. 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 an NVxx workstation and operation text of START or STOP. This text matches a CMD attribute of the OPCACMD entry for RMF in the SA OS/390 policy database. For example, the text START, defined in an operation on NV06, sends the request to SA OS/390 for processing the INGREQ command coded in the OPCACMD entry for RMF:

```
CMD=(START,, 'INGREQ RMF REQ=START,TYPE=NORM,SOURCE=EXTERNAL')  
CMD=(STOP,, 'INGREQ RMF REQ=STOP,VERIFY=NO,RESTART=NO,SCOPE=ONLY,SOURCE=EXTERNAL')
```

When the NV06 operation with text STOP becomes ready, then the AOFS6 domain will execute the INGREQ command as specified. The CODE attributes for START and STOP of the OPCA entry for RMF 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 details. |

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.
- AOCTRACE ON may help with resolving problems.

Chapter 5. Setting Up OPC Automation

This chapter explains how to set up OPC Automation in OPC and in SA OS/390.

Defining Automated OPC Applications

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

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

Defining Information for OPC Automation in OPC

The information that is passed to OPC Automation is entered in standard OPC description fields. This information is used to route requests where they are verified and executed. When the request is completed, a status change for the operation is sent back to OPC. The minimum information that needs transferring to accomplish this is listed in Table 3.

Table 3. OPC Automation Items Defined in OPC

| Definition in OPC | Information item | Refer to |
|---------------------------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------------------|
| General reporting workstations that represent target NetView domains | OPC workstation ID representing the NetView domain ID | "Defining the Target NetView Domains" on page 49 |
| OPC-defined application making requests to OPC Automation | Application name | Application field in Figure 19 on page 51 |
| Target subsystem, such as RME, for which this function is executed | Job name | Job name field in Figure 19 on page 51 |
| Operations executed within a job | Operation number or numbers | No. field in Figure 19 on page 51 |
| Request and request parameters to be performed for the specific operation | A request, such as STOP or START, and optional parameters | Operation text field in Figure 20 on page 51 and Figure 21 on page 52 |

Defining the Target NetView Domains

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

NVxx

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

Figure 18 on page 50 shows a typical definition.

```

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

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

Work station      : NV00
Description       : NetView ADFS6 for test

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

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

Transport time    : 0:00
Duration          :

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

```

Figure 18. Sample NVxx Workstation Definition in OPC

Entries in the SA OS/390 policy database (WORKSTATION DOMAINS policy object, entry type ODM) translate NVxx to an actual NetView domain ID. The automation programmer defines these entries. In this manner, the scheduler defining the OPC applications does not need to know the NetView domain ID names, but rather works with the workstation representation of those names. This allows changes to the relationship of workstations to NetView domain IDs without modifying the OPC definitions.

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

Defining Applications for OPC Automation in OPC

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

- Application making the request
- Function requested

Application Making the Request: The application making the request is defined to OPC with operations specified on the NVxx workstation, as shown in Figure 19 on page 51.


```

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

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

Application          : FORCE          Test for maint appl

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

```

Figure 21. Request Using Optional Parameters

Displaying OPC Automation Requests in OPC

Since OPC Automation requests are stored as operation text, you can view them in OPC, as shown in Figure 22.

```

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

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

Application          : RMFBKUP          RMF Backup Processing

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

```

Figure 22. Browsing Operations Including OPC Automation Requests

The following list defines several of the fields that are shown on the panel in Figure 22.

- NV04** Represents the target NetView domain or the sysplex the request is sent to.
- RMFBKUP** The application submitting the request to OPC Automation.
- RMF** Target subsystem. This looks like a job to OPC, but is actually a subsystem.
- 005 and 010** Standard operation sequence numbers used by OPC.
- STOP and START** Requested function. There are no parameters in this example.

Example of an Application Making a Request

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

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

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

Figure 23 shows the initial panel in the application creation process.

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

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

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

APPLICATION TEXT  ==> RMF maintenance_____
                                   Descriptive text of application

Owner:
ID                ==> NSC02_____
TEXT             ==> _____
                                   Descriptive text of application owner

PRIORITY          ==> 5           A digit 1 to 9 , 1=low, 8=high, 9=urgent
VALID FROM       ==> 95/04/17    Date in the format YY/MM/DD
STATUS           ==> A           A - Active, P - Pending
AUTHORITY GROUP ID ==> _____ Authorization group ID
CALENDAR ID      ==> _____
                                   For calculation of work and free day
```

Figure 23. RMF Maintenance Application Primary Panel in OPC

In Figure 23, certain fields, such as calendar ID, are not used. However, OPC Automation does not preclude the use of normal application and operation functions.

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

In Figure 24 on page 54, three operations are defined. The first and third send requests to OPC Automation in the NetView domain that is associated with the NV00 workstation. The second is a batch job named RMFMAINT that performs the batch maintenance tasks.

```

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

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

Application          : MAINT          RMF maintenance

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

```

Figure 24. Operations in the MAINT Application

Selecting TEXT as a primary command allows entry of the operation text. OPC Automation uses the **Operation text** field to contain the request and up to two optional parameters for operations with the workstation defined for OPC Automation. Figure 25 shows the resulting operations text detail panel.

```

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

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

Application          : MAINT          RMF maintenance

Row Oper      Duration Job name  Operation text
cmd ws  no.   HH.MM
'''' NV00 005   0.01   RMF_____  STOP_____
'''' CPU1 010   0.11   RMFMAINT _____
'''' NV00 015   0.01   RMF_____  START_____
***** BOTTOM OF DATA *****

```

Figure 25. Operations Text Detail Panel

In the applications, OPC Automation defines OPC requests in a generic manner. Figure 25 shows the MAINT application with the first and last operations defined for the NetView workstation NV00. The requests that are forwarded to the NetView workstation NV00 are STOP and START. These requests are expanded by definitions in the policy database into commands; see “Executing OPC Requests with OPC Automation” on page 56.

Handling Time Dependencies

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

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

As an example, redefine the MAINT application shown in Figure 23 on page 53, and Figures 24 and 25 on page 54, with a timer dummy workstation (TIMR) as the first operation of the application. The panel in Figure 26 shows this new definition.

```

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

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

Application          : MAINT          RMF maintenance

Row  Oper      Duration Job name  Internal predecessors      More preds
cmd  ws   no.   HH.MM  _____  _____  _____  _____  _____  _____
'''' TIMR 005   0.01   _____  _____  _____  _____  _____  _____
'''' NV00 010   0.01   RMF_____  005_____  _____  _____  _____  _____
'''' CPU1 015   0.10   RMFMAINT  010_____  _____  _____  _____  _____
'''' NV00 020   0.01   RMF_____  015_____  _____  _____  _____  _____
***** BOTTOM OF DATA *****

```

Figure 26. Using Time as a Dependency

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

Changes to the Status of the Operation

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

If any definition problems are determined, OPC Automation updates the status to E with an error code of Uxxx. See *System Automation for OS/390 Messages and Codes*. OPC Automation takes no further action. The user is then responsible for correcting the error and restarting the application at the failed operation.

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

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

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

Extending the Daily Plan

OPC Automation does not call EQQUX007 for time-delay operations added at daily planning. To provide time-delay operations added at daily planning, you need to define an operation on a dummy workstation as a predecessor to the NVxx workstation. The operation on that workstation completes immediately after the daily plan is extended, and the operation on the NetView workstation is READY when all of its other dependencies are satisfied. See “Handling Time Dependencies” on page 54 for more information and an example of this technique.

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

Sending a Request to Optional Installation-Provided Functions

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

- Issuing a non-SA OS/390 command or request
- Sending a request through to OPC Automation to an installation extension

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

Executing OPC Requests with OPC Automation

Generally, you must use the OPC-specific OPCA, OPCACMD, and (optionally) OPCAPARM keywords to put an application defined to SA OS/390 under the control of OPC Automation. You must define these entries under the MESSAGES/USER DATA policy item of the respective application in the SA OS/390 policy database. See *System Automation for OS/390 Defining Automation Policy* for more information on the MESSAGES/USER DATA policy item, and “Chapter 6. MESSAGES/USER DATA Entries and USER E-T Pairs for OPC Automation” on page 65 for the OPC-specific keywords.

You can vary the amount of fine-tuning considerably. If you only want to start and stop subsystems known to SA OS/390 through OPC in a standard way, you need not even code any of these keywords. On the other hand, you can call user-written modules with OPCACMD that perform tasks not related to any SA OS/390 subsystem and that must inform OPC about the success of their execution independently of OPC Automation.

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

OPC Requests and MESSAGES/USER DATA Keywords

You put a request to OPC Automation by specifying the requested function and (optionally) one or two parameters in the **Operation text** field of the **Operations** panel for an OPC application (for details, see “Defining Applications for OPC Automation in OPC” on page 50). For example:

```

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

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

Application           : MAINT           Test for maint appl

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

```

Figure 27. OPC/ESA Operations Panel

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

OPCACMD Keyword

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

```

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

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

Subsystem  : RMF
Message ID : OPCACMD

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

Pass/Selection Automated Function/'*'
Command Text
START _____
INGREQ RMF REQ=START,TYPE=NORM,SOURCE=EXTERNAL_____

OPMSG _____
MSG ALL RMF MSG _____

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

```

Figure 28. Specifying the Command for a Request

These entries specify in the **Command Text** field the commands that are to be issued for RMF when START, respectively, OPMSG requests are put to OPC Automation. Thus, when the 005 request of Figure 27 on page 57 has been put to OPC Automation, OPC Automation will issue the command

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

OPCA Keyword

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

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

This is done through the OPCA keyword. The following panel continues the example of Figure 28.

```

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

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

Subsystem : RMF
Message ID : OPCA

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

Code 1      Code 2      Code 3      Value Returned
START _____          _____          _____          UP,2,RMFUTMER_____
OPMSG _____          _____          _____          UP,1,_____
STOP _____          _____          _____          CTLDOWN,2,_____
_____          _____          _____          _____
_____          _____          _____          _____

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

```

Figure 29. Specifying Expected Status and Time Interval

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

Flow of Control

By the entries of Figure 28 on page 58 and Figure 29, OPC Automation will execute the start request of Figure 27 on page 57 for RMF as follows:

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

Request Parameters and the &EHKVAR/ Variables

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

As an example, suppose you want to specify the startup type for a CICS application in an OPC start request. Then you enter it as a request parameter in the **Operation text** field (see the 010 request in Figure 27 on page 57) and pass the startup type to the command specified in the OPCACMD entry by incorporating the &EHKVAR1 variable in the command text. In the following example, this

command is not an SA OS/390 command, but a user-written CLIST named MYCLIST that uses the startup information to apply the desired startup method.

```

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

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

Subsystem  : CICS1H
Message ID : OPCACMD

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

Pass/Selection Automated Function/'*'
Command Text
START _____
MYCLIST CICS1H &EHKVAR1 _____

_____

_____

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

```

Figure 30. Specifying a Command that Requires Parameter Information

Then, when the request of the 010 operation in Figure 27 on page 57 is put to OPC Automation, MYCLIST will be called with the arguments CICS1 and COLD. A very simple version of MYCLIST could be as follows:

```

/* MYCLIST SAMPLE */
PARSE UPPER ARG CICSNAME STARTTYPE

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

```

If you use parameters, you must code an OPCA entry for every parameter (combination). For the example of Figure 30, the OPCA entry could look as follows:

```

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

Entry Type : Application                      PolicyDB Name : SCENARIO
Entry Name : CIGS1H                          Enterprise Name : TEST

Subsystem : CIGS1H
Message ID : OPCA

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

Code 1      Code 2      Code 3      Value Returned
START _____ COLD _____ _____ UP,10,CIGS1TMR _____
START _____ AUTO _____ _____ UP,5,CIGS1TMR _____
_____
_____
_____

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

```

Figure 31. Specifying Expected Status and Time Interval for Different Request Parameters

Request Types

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

Starting and Stopping Subsystems without OPC-Related Keywords

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

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

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

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

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

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

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

Note that if you want to add or change any parameter other than the TYPE parameter, you must specify your INGREQ command in an OPCACMD entry and

code the associated OPCA entry. For more information on INGREQ, see *System Automation for OS/390 Operator's Commands*.

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

Requests Using SA OS/390 Automation Functions

These are requests for which an OPCACMD entry is coded, and where the command specified in that entry changes the automation status of the subsystem to UP, RUNNING or CTLDOWN. This can be an SA OS/390 base command (for example, INGREQ), but also a user-written command that calls INGREQ. The name of the request must not begin with 'UX'. When the request contains parameters, these are stored in the &EHKVAR1 and &EHKVAR2 variables, respectively, and you can pass them to the command by incorporating these variables into the command text.

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

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

Subsystem Related Requests not Using SA OS/390 Automation Functions

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

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

For more information on this request type, see "User Functions Related to an SA OS/390-Defined Subsystem" on page 97.

Non-Subsystem Requests

These are requests for which an OPCACMD entry is coded, and where the command specified in that entry does not relate to a subsystem known to SA OS/390. The name of these requests must begin with 'UX'. The OPCACMD entry for a non-subsystem request must be coded as a USER E-T pair. With requests of this type, the complete request buffer will be stored in &EHKVAR1, and it is up to the user —written command to analyze this information. For the request buffer in general, see "Request Buffers and OPC Automation Log Entries" on page 15; for the format of the request buffer for 'UX' requests, see Table 6 on page 95.

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

For more information on this request type, see “Non-Subsystem Operations” on page 101.

Chapter 6. MESSAGES/USER DATA Entries and USER E-T Pairs for OPC Automation

As OPC Automation is integrated into SA OS/390, you must enter any information for OPC Automation in the policy database via the customization dialogs. In most cases the customization dialogs precisely determine the format in which this information must be entered. There are, however, some OPC application-specific automation parameters that must or can only be specified as entries in the MESSAGES/USER DATA policy items of the respective application, or as USER E-T pairs; for general information on the MESSAGES/USER DATA policy item and USER E-T pairs, see *System Automation for OS/390 Defining Automation Policy*. In these cases, the customization panels provide no information about the keywords and the format of their parameters.

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

Translating Format Descriptions

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

The first example is the OPCACMD keyword. With this entry, you specify the command that is executed in response to a certain OPC request (see "OPCACMD" on page 74 for more details). The format description of OPCACMD is as follows:

| Format |
|---------------------------------------------|
| <code>OPCACMD CMD=(request,,command)</code> |
| <code>.</code> |
| <code>.</code> |
| <code>[CMD=(request,,command)]</code> |

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

The translation of the format description for OPCACMD depends on whether or not the specified command is related to a subsystem that is known to SA OS/390. In the first case (which will also be treated first), the entry is defined in the MESSAGES/USER DATA item of the respective subsystem; for the second case, in which it must be entered within a USER E-T pair, see the end of this section on page 69.

Accordingly, suppose first that a START request is specified in an OPC operation for the subsystem CICS1H, and that you want to start CICS1H with the INGREQ command (for the connection between the OPC request and the command, see "Chapter 5. Setting Up OPC Automation" on page 49). To specify an OPCACMD

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

```

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

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

Subsystem   : CICS1H

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

Action      Message ID              Cmd  Rep  Code  User
Description
CMD_____  OPCACMD_____
            Commands for OPC requests_____
_____
_____
_____
_____

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

```

Figure 32. Message Processing Panel of the Customization Dialogs 1

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

Rule

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

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

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

```

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

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

Subsystem   : CICS1H
Message ID  : OPCACMD

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

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

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

```

Figure 33. CMD Processing Panel of the Customization Dialogs

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

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

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

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

Notational Conventions for CMD, CODE, REP

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

For more information on the panel fields, see *System Automation for OS/390 Defining Automation Policy*.

The OPCA entry (see "OPCA" on page 71) supplies the second example. This entry defines the state that is the expected result of the command specified in the OPCACMD entry, and the time interval within which this state must have been reached. The format of the OPCA entry is as follows:

```

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

```

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

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

```

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

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

Subsystem : CICS1H

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

Action  Message ID              Cmd  Rep  Code  User
        Description
-----
OPCACMD  Commands for OPC requests
CODE    OPCA
        Expected status and time interval

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

```

Figure 34. Message Processing Panel of the Customization Dialogs 2

Now you must enter CODE in the **Action** column according to the rule stated on page 66. When you press ENTER, the **Code Processing** panel is called:

```

COMMANDS  HELP
-----
Code Processing                               Row 1 to 6 of 20
Command ==>                                SCROLL==> PAGE

Entry Type : Application                      PolicyDB Name : SCENARIO
Entry Name : CIGS1H                          Enterprise Name : TEST

Subsystem : CIGS1H
Message ID : OPCA

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

Code 1      Code 2      Code 3      Value Returned
START _____ UP,3,RMFUTMER _____
_____
_____
_____

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

```

Figure 35. Code Processing Panel of the Customization Dialogs

Here you must specify the values of the CODE attribute as displayed in Figure 35. The general format of the CODE attribute is:

CODE=([Code_1],[Code_2],[Code_3],Value_Returned)

For the OPCA keyword, **Code 1** must be the request. In the **Code 2** and **Code 3** fields, you can specify eventual parameters of the request. The **Value Returned** must specify the expected status, the time interval, and (optionally) a timer name, separated by commas.

Note: For other keywords, the fields can have a completely different function from those discussed above.

An asterisk (*) is admitted as a *trailing* wildcard character for the three **Code** fields; that is, you can specify simply * and ABC*, but not *ABC.

The preceding two examples illustrate how format descriptions are translated into MESSAGES/USER DATA entries. However, as already indicated, the OPCACMD entry must be coded in a USER E-T pair if the command to be specified is not related to a subsystem known to SA OS/390. The following figure provides an example:

```

COMMANDS  ACTIONS  HELP
-----
UET Keyword-Data Specification                               Row 3 from 3
Command ==>>>                                           SCROLL==>> PAGE

Entry Type : User E-T Pairs           PolicyDB Name  : SCENARIO
Entry Name  : NONSUBS                 Enterprise Name : TEST
UET Entry   : DUMMY                  UET Type      : OPCACMD

Action  Keyword/Data(partial)
-----  CMD
        (UXCKSPL,, 'EVJERUX1 &EHKVARI')
***** Bottom of data *****

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

```

Figure 36. OPCACMD in a USER E-T Pair

As you can see from Figure 36, you must enter CMD in the **Keyword** field and the value list in the **Data** field.

For the USER E-T pairs, the following translation conventions apply:

Notational Conventions for UET Keyword/Data Pairs

- The '=' sign must not be entered in the panels.
- Everything to the right of the '=' sign, including parentheses, commas, and single quotes, *must* be entered in the **Data** field.

OPC-Specific MESSAGES/USER DATA Keywords

The following keywords are specific for OPC Automation.

| Entry | Description |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| "OPCA" on page 71. | Use this ID to define the expected state of the subsystem and the time interval within which this state must have been reached.. |
| "OPCACMD" on page 74. | Use this ID to specify the command to be issued in response to an OPC request. |
| "OPCAPARM" on page 78. | Use this ID to specify modifications of eventual request parameters and a timer module for user-written commands. |

OPCA

Purpose

With the OPCA entry, you define the state that is the expected result of a request (with or without parameters), and the time interval within which this state must have been reached. The OPCA entry is defined in the MESSAGES/USER DATA policy item of the subsystem which is to be put under control of OPC.

Format

Format

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

Parameters

request

Request specified in the OPC operation text.

parm1

Parameter 1 as specified in the OPC operation text.

parm2

Parameter 2 as specified in the OPC operation text.

expstatus

Expected status of the subsystem at the completion of the request. *expstatus* stands for one of the following values: UP, RUNNING or CTLDOWN.

timerint

Timer interval in minutes. The maximum value permitted is 1439 (23 hours and 59 minutes).

Set a timer interval that is long enough for the operation to complete reasonably. If the operation does not complete in the interval specified, then an error is posted to OPC.

timerid

Timer ID — from 1 to 8 characters.

This must be a valid NetView timer ID with a value not equal to ALL or beginning with SYS, ING, or AOF. This field is optional.

Usage Notes

For every OPCA entry, there must be a corresponding OPCACMD entry.

Example 1

```

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

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

Subsystem   : RMF
Message ID  : OPCA

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

      Code 1          Code 2          Code 3          Value Returned
START _____    _____    _____    UP,3,RMFUTMER_____
STOP  _____    _____    _____    CTLDOWN,2,RMFDTMER_____
_____
_____

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

```

This panel shows the entries for two requests without parameters.

Example 2

```

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

Entry Type : Application          PolicyDB Name  : SCENARIO
Entry Name  : CICS1              Enterprise Name : TEST

Subsystem   : CICS1
Message ID  : OPCA

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

      Code 1          Code 2          Code 3          Value Returned
START _____    AUTO _____    _____    UP,5,CICS1TMR_____
START _____    COLD _____    _____    UP,10,CICS1TMR_____
_____
_____

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

```

This example shows two entries, one for an automatic start, and one for a cold start, of a subsystem called CICS1. The AUTO or COLD parameter is added to the START request in OPC to indicate which one of several startup procedures is to be used. Presumably the two operations will take differing amounts of time, so the timer intervals are different.

This OPCA code entry is used in conjunction with a user-written CLIST, specified in the OPCACMD entry; see "Example 2" on page 75 for details of that entry and the sample CLIST.

Example 3

```

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

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

Subsystem : TESTAPPL
Message ID : OPCA

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

Code 1      Code 2      Code 3      Value Returned
RECYCLE_____          _____          _____          UP,5,TESTTIMER_____
_____
_____
_____

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

```

This example shows a RECYCLE type of operation (bring the subsystem down and restart it immediately) being defined. See also “Example 4” on page 77 for the corresponding OPCACMD definition.

OPCACMD

OPCACMD

Purpose

With the OPCACMD entry, you define the command that is executed in response to a request (with or without parameters). Except for non-subsystem commands, there must be a corresponding OPCA entry for every OPCACMD entry.

Format

Format

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

Parameters

request

Request specified in the OPC operation text.

command

Actual command to be built.

Usage Notes

This entry is necessary for all request types except default START and STOP requests (for these default requests, see "Starting and Stopping Subsystems without OPC-Related Keywords" on page 61). The place where the OPCACMD entry is defined depends on the request type. When the request is related to a subsystem, it is defined in the MESSAGES/USER DATA policy item of the respective application. When the request is a so-called non-subsystem request (see "Non-Subsystem Operations" on page 101), the OPCACMD entry must be entered in a USER E-T PAIRS entry.

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

```

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

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

Subsystem   : RMF
Message ID  : OPCACMD

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

Pass/Selection Automated Function/'*'
Command Text
START_____
INGREQ RMF REQ=START,TYPE=NORM_____

STOP_____
INGREQ RMF REQ=STOP,VERIFY=NO,RESTART=NO,SCOPE=ONLY_____

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

```

Example 2

```

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

Entry Type : Application                PolicyDB Name  : SCENARIO
Entry Name  : CICS1                    Enterprise Name : TEST

Subsystem   : CICS1
Message ID  : OPCACMD

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

Pass/Selection Automated Function/'*'
Command Text
START_____
MYCLIST CICS1 &EHKVARI_____

_____

_____

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

```

This example assumes that AUTO or COLD is added as a parameter to the START request in OPC to indicate which one of several startup procedures is to be used. The command specified in the **Command Text** field is a user-written CLIST. This CLIST is passed the parameter value (AUTO or COLD) in the &EHKVARI variable.

A very simple version of the CLIST, which could be expanded in your environment to include other parameters and additional function, could be as follows:

```

/* MYCLIST SAMPLE */
PARSE UPPER ARG CICSNAME STARTTYPE

IF STARTTYPE='COLD' THEN

```

OPCACMD

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

The OPCACMD entry of this example must be supplemented by an OPCA entry as in “Example 2” on page 72.

Example 3

```
COMMANDS  ACTIONS  HELP  
-----  
                                UET Keyword-Data Specification          Row 3 from 3  
Command ==>>>                                SCROLL==>> PAGE  
  
Entry Type : User E-T Pairs          PolicyDB Name   : SCENARIO  
Entry Name  : NONSUBS                Enterprise Name : TEST  
UET Entry   : DUMMY                  UET Type       : OPCACMD  
  
Action      Keyword/Data(partial)  
-----  
            CMD  
            (UXCINITS,'MVS $TI20-30,C=P')  
***** Bottom of data *****  
  
F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=RFIND      F6=RCHANGE  
F7=UP        F8=DOWN        F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEVE
```

This example shows a command that is not related to a subsystem known to SA OS/390 (see “Non-Subsystem Operations” on page 101), and which, accordingly, must be defined as a USER E-T pair (see “Non-Subsystem Operations” on page 101). The jobname of the OPC request would be DUMMY.

OPC Automation recognizes such requests by the fact that the request name begins with 'UX'. In the example, OPC Automation simply issues the MVS command \$TI20-30,C=P, which tells JES to change initiators 20 to 30 so that they process jobs of class P.

Example 4

```

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

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

Subsystem   : TESTAPPL
Message ID  : OPCACMD

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

Pass/Selection Automated Function/ '*'
Command Text
RECYCLE_____
EVJESHUT TESTAPPL ALL_____

_____

_____

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

```

This example shows a RECYCLE type of operation (bring the subsystem down and restart it immediately) being defined. Note that the command to be issued is EVJESHUT, which will issue the

```
INGREQ TESTAPPL REQ=STOP,RESTART=YES,SCOPE=ALL,SOURCE=EXTERNAL
```

command in linemode and verify that it is accepted or else post the operation in error. For more information on EVJESHUT, see "EVJESHUT" on page 82; for the corresponding OPCA definition, see "Example 3" on page 73.

OPCAPARM

Purpose

The OPCAPARM entry supplies replacements for eventual request parameters and the name of a user-written timer module. The OPCAPARM entry is defined in the MESSAGES/USER DATA policy item of the subsystem which is to be put under control of OPC.

OPCAPARM is optional except for requests that relate to a subsystem defined to SA OS/390, but where the command specified in the OPCACMD entry is not an SA OS/390 command; in this case, you must specify a timer module.

Format

Format

```
OPCAPARM CODE=(request,param1,param2,'parm1value,param2value,timermod')
          .
          .
          .
          [CODE=(request,param1,param2,'parm1value,param2value,timermod')]
```

Parameters

request

Request specified in the OPC-operation definition.

parm1

Parameter 1 as specified in the OPC-operation text.

parm2

Parameter 2 as specified in the OPC-operation text.

parm1value

Substitution value used in the actual command.

parm2value

Substitution value used in the actual command.

timermod

Module called at the timer interval specified in the OPCA CODE entry for this subsystem. You must specify a timer module when the command specified in the OPCACMD entry relates to a subsystem defined to SA OS/390, but does not trigger a status change; see "User Functions Related to an SA OS/390-Defined Subsystem" on page 97.

Usage Notes

OPCAPARM is optional except for requests that relate to a subsystem defined to SA OS/390, but where the command specified in the OPCACMD entry is a user-supplied module that does not trigger a status change of the subsystem. In this case, you must specify a timer module. See "Implementing Completion of a Request" on page 98 for more details.

Example 1

```

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

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

Subsystem : RMF
Message ID : OPCAPARM

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

Code 1      Code 2      Code 3      Value Returned
START _____ , , _____
_____
_____
_____

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

```

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

Example 2

```

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

Entry Type : Application                      PolicyDB Name : SCENARIO
Entry Name : CICS1                           Enterprise Name : TEST

Subsystem : CICS1
Message ID : OPCAPARM

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

Code 1      Code 2      Code 3      Value Returned
START _____ , , _____
START _____ COLD _____ , , _____
_____
_____
_____

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

```

This example shows two entries, one for an automatic start, and one for a cold start, of a subsystem called CICS1. The AUTO or COLD parameter is added to the START request in OPC to indicate which one of several startup procedures is to be used.

OPCAPARM

Chapter 7. OPC Automation Common Routines and Data Areas

This chapter contains the common routines that are supplied by OPC Automation. Furthermore, it describes the data areas that are used to transfer requests from OPC to SA OS/390.

OPC Automation Common Routines

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

OPC Automation provides new routines to retrieve and update OPC Automation-unique information. These routines can also be used in user-written extensions of OPC Automation. The following routines are arranged alphabetically for easy reference.

Note

When using any of the OPC Automation commands that use the EQQYCOM (also called the PIF) interface, you should be careful to ensure serialization. This can be achieved by choosing the same auto operator for the command to run on. If you fail to do this you may receive an EQQZ038E message indicating that the OPC message log is not available, because the log data set (EQQMLOG, see "NetView Related Steps" on page 44) is required by EQQYCOM exclusively for each invocation.

EQQYCOM is used by the following commands: OPCACAL, OPCALIST, OPCAMOD.

EVJESHUT

EVJESHUT

Purpose

Use the EVJESHUT routine in a RECYCLE operation defined in the policy data base. A RECYCLE operation is one in which a subsystem which is currently UP is brought down and immediately restarted. EVJESHUT will issue the appropriate INGREQ command and verify that it is accepted by the automation platform. If it is not accepted, then EVJESHUT will post the operation in error to OPC. See "Example 4" on page 77 to see how to include EVJESHUT in an OPCACMD entry.

Format

Syntax

```
EVJESHUT subsys scope
```

Parameters

subsys

The name of a valid subsystem defined to SA OS/390.

scope

The scope of the INGREQ request. The valid values for *scope* are ALL, CHILDREN, and ONLY. See *System Automation for OS/390 Operator's Commands*.

Usage Notes

The **Status check on requests** field in the ENVIRON OPCAO item of the OPC SYSTEM DETAILS policy object determines for a system what OPC Automation is to do when a subsystem of this system is already in the status requested by OPC. This field affects EVJESHUT as follows: When the field is set to NO, and the subsystem is already in AUTODOWN, CTLDOWN, DOWN, or ENDED status, then a RECYCLE operation with EVJESHUT will be allowed to proceed and EVJESHUT will issue an INGREQ command to stop the requested subsystem. If the field is set to YES, then the subsystem must be in UP or RUNNING status for EVJESHUT to be issued; all other statuses will result in the operation terminating and an error posted to OPC.

OPCACAL

Purpose

The OPCACAL command retrieves OPC calendar status information. Use this command for your automation CLISTs. OPCACAL uses the EQQYCOM (also called the PIF) interface in OPC. For more information about this interface, see the *OPC/ESA Interfaces Guide*.

To show the use of this command, OPC Automation provides two sample CLISTs: EVJECCAL (CLIST) and EVJERCAL (REXX).

Format

Syntax

```
OPCACAL [SUBSYS=subsystem,CALENDAR=calname]
```

Parameters

SUBSYS=*subsystem*

OPC/ESA subsystem ID — 4 characters.

OPCA is the default subsystem name.

CALENDAR=*calname*

Calendar ID — 16 characters.

DEFAULT is the default calendar name, used if this parameter is not coded.

OPCACMD

OPCACMD

Purpose

The OPCACMD command is primarily used to retrieve OPC current plan data so that it can be viewed or modified by operators. The operator simply types in OPCACMD and fills in the panel (EVJKAC01) which is returned. But OPCACMD will also accept optional parameters on input. It could thus be assigned to a PF key, making it more user-friendly.

Format

Syntax

```
OPCACMD APPLID=application_id,OPNO=operation_number,  
        WSNAME=ws_name,STATUS=status,ERRCODE=errorcode,  
        PRIORITY=priority,OWNER=owner_name,GROUP=group_name,  
        JOBNAME=job_name
```

Parameters

application_id

Application name — 1 to 16 characters. May be generic.

operation_number

Operation number — 2 digits. Must be numeric or left unspecified.

ws_name

Workstation name — 4 characters. May be generic.

status

Occurrence status — 1 character. Must be valid or left unspecified.

Valid status:

| | |
|----------|-------------|
| A | Arriving |
| C | Completed |
| E | Error |
| I | Interrupted |
| R | Ready |
| S | Started |
| U | Undecided |
| W | Waiting |
| X | Reset |

errorcode

Error code — 4 characters. May be generic.

priority

Priority — 1 digit. Must be numeric or left unspecified. Acceptable values are from 1 (low) to 9 (high).

owner_name

OPC application owner name — up to 16 characters. May be generic.

group_name

OPC application group name — up to 8 characters. May be generic.

job_name

Job name — up to 8 characters. May be generic.

Usage Notes

Only as many parameters are required as necessary to identify the application(s) requested. Some parameters may be left unspecified (they will default). Some may be generic; that is, they may be only partial and end with an asterisk (*) to indicate a partial match. You may also use a percent sign (%) to substitute for a single character.

OPCACOMP

OPCACOMP

Purpose

The OPCACOMP command completes execution of a subsystem-related request by updating the OPC Automation status file and calling OPCAPOST (see “OPCAPOST” on page 92).

Format

Syntax

```
OPCACOMP subsys,sequence_number,status [,error_code]
```

Parameters

subsys

Subsystem or pseudo-subsystem to identify the request.

sequence_number

Sequence number assigned to this request by the EVJESPVY module.

status

Operation status reflected to OPC Valid statuses:

C Complete

E Error

error_code

Error code — 4-character value

Takes the form *anmn*, where *a* is alphabetic and *nmn* are numerics.

Do not specify the values *Uxxx* and *Sxxx*; reserve them for OPC Automation.

If the status is error, the error code is returned to OPC.

Usage Notes

Call this routine in user-written functions that are related to a subsystem defined to SA OS/390 whenever the standard modules for completing a request (EVJESPSC and EVJESPTE) cannot be used. See “Implementing Completion of a Request” on page 98.

Example

```
OPCACOMP RMF,842,E,R028
```

This example shows setting the operation requested for RMF in an error status with an error code of R028.

OPCALIST

Purpose

The OPCALIST command retrieves OPC data. Use this command in your own automation CLISTS. The module creates a CPOPCOM call to OPC to retrieve the data. OPCALIST uses the EQQYCOM (also called the PIF) interface in OPC. For more information about this interface, see the *OPC/ESA Interfaces Guide*.

Format

Syntax

```
OPCALIST SUBSYS=subsystem,ADID=id,IA=yymmddhhmm,
        PRIORITY=nnnn,ERRCODE=cccc,STATUS=s,OPNO=nnnn,
        JOBNAME=name,WSNAME=name,
        GROUP=groupname,OWNER=name
```

Parameters

SUBSYS=*subsystem*

OPC subsystem ID — 4 characters.

ADID=*id*

Application description ID — up to 16 characters.

IA=*yymmddhhmm*

Input arrival date *yymmdd* and time *hhmm*.

PRIORITY=*nnnn*

Priority — 4 digits.

ERRCODE=*cccc*

Error code — 4 characters.

STATUS=*s*

Occurrence status. Valid statuses:

| | |
|----------|-------------|
| R | Ready |
| S | Started |
| C | Completed |
| E | Error |
| I | Interrupted |

Refer to the OPC documentation for more information.

OPNO=*nnnn*

Operation number — 4 digits.

JOBNAME=*name*

Job name — up to 8 characters.

WSNAME=*name*

Workstation name — 4 characters.

GROUP=*groupname*

OPC/ESA application group name — up to 8 characters.

OWNER=*name*

OPC/ESA application owner name — up to 16 characters.

Usage Notes

Only as many parameters are required as necessary to identify the application(s) requested. Some parameters may be left unspecified (they will default). Some may

OPCALIST

be generic; that is they may be only partial and end with an asterisk (*) to indicate a partial match. You may also use a percent sign (%) to substitute for a single character.

The response to the OPCALIST is made up of three messages. The following example below shows a typical response where:

EVJ410I

This is the message header, showing row titles. This is always present.

EVJ411I

This is the detail message. If there are no entries matching the selection criteria this message is not produced.

EVJ412I

This is the end of request message.

Response details are:

ADID

Application Description Id — up to 16 characters

JOBNAME

Job Name — up to 8 characters

WS

Workstation Name — up to 4 characters

OPNO

Operation Number — up to 4 numbers

S Status (See “Parameters” on page 87 for valid statuses)

ERRC

Error Code (set to none for no error)

IA Input Arrival — date (yymmdd) and time (hhmm)

OPTTEXT

Descriptive Text — up to 24 characters

Example

| | ADID | JOBNAME | WS | OPNO | S | ERRC | IA | OPTTEXT |
|---------|--------|---------|------|------|---|------|------------|----------------|
| EVJ410I | | | | | | | | |
| EVJ411I | MAINT2 | RMF | NV05 | 0010 | C | NONE | 9707190615 | STOP |
| EVJ411I | MAINT2 | RMF | NV05 | 0015 | C | NONE | 9707190615 | START |
| EVJ411I | MAINT2 | RMF | NV05 | 0010 | C | NONE | 9707200615 | STOP |
| EVJ411I | MAINT2 | RMF | NV05 | 0015 | C | NONE | 9707200615 | START |
| EVJ412I | | | | | | | | END OF REQUEST |

OPCAMOD

Purpose

The OPCAMOD command modifies OPC data. This command is used in the OPCACMD CLIST and could be used in your own automation CLISTS. The module creates a CPOPCOM or CPOCCOM call to OPC to perform occurrence or operation changes. OPCAMOD uses the EQQYCOM (also called the PIF) interface in OPC. For more information about this interface, see the *OPC/ESA Interfaces Guide*.

Format

Syntax

```
OPCAMOD SUBSYS=subsystem, ADID=id, IA=yymmddhhmm,
IANEW=yymmddhhmm, DEADLINE=yymmddhhmm, PRIORITY=nnnn,
ERRCODE=cccc, OPNO=nnnn, STATUS=s,
JOBNAME=name, WSNAME=name, DESC=text,
EDUR=hhmm, PSUSE=nnnn, RIUSE=nnnn, R2USE=nnnn,
JCLASS=c, AEC=Y|N, ASUB=Y|N, AJR=Y|N, TIMEDEP=Y|N,
CLATE=Y|N, HRC=value, FORM=value, OPIA=yymmddhhmm,
OPDL=yymmddhhmm, RERUT=Y|N, USERDATA=userdata,
RESTA=Y|N, DEADWTO=Y|N
```

Parameters

SUBSYS=*subsystem*

OPC/A subsystem ID — 4 characters (default OPCA).

ADID=*id*

Application description ID — up to 16 characters (required).

IA=*yymmddhhmm*

Input arrival date *yymmdd* and time *hhmm* (required).

IANEW=*yymmddhhmm*

New input arrival date and time.

DEADLINE=*yymmddhhmm*

Deadline date and time.

PRIORITY=*nnnn*

Priority — 4 digits.

ERRCODE=*cccc*

Error code — 4 characters.

STATUS=*s*

Occurrence status. Valid statuses:

| | |
|----------|-------------|
| R | Ready |
| S | Started |
| C | Complete |
| E | Error |
| I | Interrupted |

Refer to the OPC documentation for more information.

JOBNAME=*name*

Job name — up to 8 characters.

WSNAME=*name*

Workstation name — 4 characters.

OPCAMOD

DESC=*text*

Descriptive text — 24 characters.

EDUR=*hhmm*

Estimated duration (hours and minutes).

PSUSE=*nnnn*

Number of parallel servers required — 4 digits.

R1USE=*nnnn*

Amount of resource 1 required — 4 digits.

R2USE=*nnnn*

Amount of resource 2 required — 4 digits.

JCLASS=*c*

MVS job class — 1 character.

AEC=Y/N

Y — Perform automatic error completion. N — Do not perform automatic error completion.

ASUB=Y/N

Y — Perform automatic job submission. N — Do not perform automatic job submission.

AJR=Y/N

Y — Perform automatic job hold and release. N — Do not perform automatic job hold and release.

TIMEDEP=Y/N

Y — Time dependent job. N — Not a time dependent job.

CLATE=Y/N

Y — Cancel if time job and late. N — Do not cancel if time job and late.

HRC

Highest successful return code.

FORM=*value*

Form number — 8 characters.

OPIA=*yymmddhhmm*

Operation input arrival date and time.

OPDL=*yymmddhhmm*

Operation deadline date and time.

RERUT=Y/N

Y — Reroutable operation. N — Not a reroutable operation.

USERDATA=*userdata*

User data — up to 16 characters.

RESTA=Y/N

Y — Restartable operation. N — Not a restartable operation.

DEADWTO=Y/N

Y — Issue WTO if deadline missed. N — Do not issue WTO if deadline missed.

Usage Notes

OPCAMOD may be used to set the status of operations occurring on general workstations which are not automatically reporting (such as CPUs). OPCAPOST will set the status of operations which occur on automatically reporting general workstations (a NetView, for example).

Format

Syntax

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

Parameters

SUBSYS=*subsystem*

OPC/ESA subsystem ID — 4 characters (default OPCA).

ADID=*id*

Application description ID — up to 16 characters (required).

IA=*yymmddhhmm*

Input arrival date *yymmdd* and time *hhmm* (required).

IANEW=*yymmddhhmm*

New input arrival date and time.

DEADLINE=*yymmddhhmm*

Deadline date and time.

PRIORITY=*nnnn*

Priority — 4 digits.

ERRCODE=*cccc*

Error code — 4 characters.

STATUS=*s*

Occurrence status. Valid statuses:

| | |
|----------|-----------|
| W | Waiting |
| C | Completed |

Example

```
OPCAMOD SUBSYS=OPCA,ADID=TEST,IA=9403100900,OPNO=0010,
STATUS=W
```

This example will set the status of operation number 0010 in application test, to waiting.

Example

```
OPCAMOD SUBSYS=OPCA,ADID=TEST,IA=9403100900,STATUS=W
```

This example will set the occurrence status of application TEST to WAITING. All operations in application TEST will be set to WAITING.

OPCAPOST

Purpose

OPCAPOST posts the status of an OPC Automation operation back to OPC. Because OPCAPOST uses the EQQUSINT interface, it can only change the status of operations on automatic reporting workstations. For more information on this interface, see *OPC/ESA Installation and Customization*.

Format

Syntax

```
OPCAPOST ADNAME=adname, WSNAME=www, OPNUM=nn,
        TYPE={S|C|I|E|X}, ERRCODE=xxxx
```

Parameters

ADNAME=*adname*

Application name — 1 to 16 characters.

WSNAME=*www*

Workstation name — 1 to 4 characters.

OPNUM=*nn*

Operations number — 2 digits.

TYPE={S|C|I|E|X}

Type of call — 1 character. Acceptable event types:

| | |
|---|-------------|
| S | Started |
| C | Complete |
| I | Interrupted |
| E | Error |
| X | Reset |

ERRCODE=*xxxx*

Error code — 4 characters.

Note: This parameter is only valid with TYPE=E.

Usage Notes

OPCAPOST can be used to set the status of an operation which occurs on an automatically reporting general workstation (a NetView, for example) only. Due to restrictions in the OPC interface, OPCAPOST cannot set the status of operations on general workstations which are not automatically reporting (such as CPUs). OPCAMOD is available to set the status of such operations if this is required.

OPC Automation sets a return code on completion of the execution of the OPCAPOST command processor, as follows:

| | |
|---|--------------------|
| 0 | Successful command |
| 4 | Parameter error |
| 8 | OPCAPOST failed. |

OPCSRST

Purpose

The OPCSRST command lets you manipulate the availability of OPC special resources. It is similar to the SRSTAT operator command, as presented in “Chapter 10. OPC Automation Operator Commands” on page 117.

Format

Syntax

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

Parameters

subsys

Subsystem ID — 4 characters.

sname

Special resource name — up to 44 characters.

AVAIL=Y/N

Availability indicator.

Usage Notes

OPCSRST uses the following return codes:

- 0 Accepted by OPC — Special Resource-event issued
- 4 Request failed — Parameter error detected by this program
- 8 Request failed — Rejected by OPC — No Special Resource-event issued.

Internal failures:

- 1011 SRNAME keyword not supplied
- 1010 AVAIL keyword not supplied
- 1020 DSILOD failed — Unable to load EQQUSINS
- 1030 DSILCS failed — Unable to obtain SWB
- 1031 DSIPRS failed — Unable to determine size of PDB
- 1032 DSIGET failed — Unable to obtain storage
- 1033 DSIPRS failed — Unable to do parse

Example

```
OPCSRST SUBSYS=OPCT,SRNAME='IMSCNTL.IMS01A.RUNNING',AVAIL=Y
```

In this example, the OPCSRST command is run when the IMS control region IMS01A becomes available. The special resource becoming available makes it possible for work that depends on this control region’s execution to run. When IMS01A becomes available, a number of applications are added to the current plan.

The variable used for *subsys*, OPCT, is the name of the tracker subsystem. Only in this command is the tracker subsystem name required.

Data Areas

This section shows the following:

- Requestor ID block (&EHKVAR9)
- Request buffer

Requestor ID Block (&EHKVAR9)

OPC Automation sets the task global variable (&EHKVAR9) in the request module and passes it to the user module, as follows:

Name of Subsystem, Sequence #, Module Name, Domain ID

Table 4 shows the lengths and values of the variables.

Table 4. Lengths and Values of Task Global Variable (EHKVAR9)

| Variable | Name of Subsystem | Sequence # | Module Name | Domain ID |
|---------------------|-------------------------------------|-------------------------------|-------------------|------------------------------|
| Length (characters) | 8 | 4 | 8 | 5 |
| Values | SA OS/390 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 4.

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 5 shows the request buffer layout for standard subsystem operations:

Table 5. 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 | NVnn | 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 6 shows the request buffer layout for nonsubsystem, user extension (UXaaaaaa) operations:

Table 6. Request Buffer Layout for Nonsubsystem, User Extension (UXaaaaaa) 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 | NVnn | WSNAME |
| delimiter | 1 | F | blank | constant |
| Operation no | 3 | V | 1 – 255 | OPNO |
| delimiter | 1 | F | blank | constant |
| Subsystem name | 8 | V | variable | JOBNAME |
| delimiter | 1 | F | blank | constant |
| Request | 24 | V | variable | TXTOP |

Any parameter with a variable length is left-adjusted and all trailing blanks are ignored.

Chapter 8. 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 an action for a subsystem known to SA OS/390.
- 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.

User Functions Related to an SA OS/390-Defined Subsystem

OPC Automation provides support for stopping and starting 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 SA OS/390 policy database. However, since you may not need to use this command under certain conditions, OPC can initiate a user-supplied module to perform the command. You can split the startup sequence with the system commands, so OPC executes them separately from the subsystem startup commands. If this is the case, define each command sequence to OPC as an operation. Using scheduling parameters, such as specific types of days, you can include or exclude certain operations.

For example, consider a subsystem which normally runs on a specific processor. On weekends, you use this processor for testing purposes and move the application to another, perhaps smaller, processor within the same complex. On the days that the application needs moving, you need several VTAM[®] VARY commands to start the VTAM application statements.

In OPC, you can define an extra operation or application which runs on the first free day of each period, and another which runs on the first working day of each period. OPC calls the CLIST containing the VTAM commands. This allows the issuing of the appropriate VARY commands when needed before you start the application subsystem on the correct processor.

Triggering a user-written CLIST provides another example. This determines if all users of a specific application are logged off before issuing the commands to take down the subsystem.

Flow of Control

In a situation where a non-SA OS/390 command needs issuing, specify the user CLIST or command processor in the OPCACMD entry of the subsystem. In

response to the request, instead of issuing an SA OS/390 command, OPC Automation passes control to this user CLIST or command processor. All information available is made accessible to the user-supplied module. If the user-supplied module does not trigger a status change of the subsystem and returns control to OPC Automation synchronously, you are responsible for completing the operation. This should be done by calling OPCACOMP once the results of these commands are analyzed. The OPCACOMP module ensures that actions are accomplished in the correct sequence, does some housekeeping, updates the SA OS/390 status file, and calls the OPCAPOST command processor to return the specified completion code to OPC; for more details see “Implementing Completion of a Request”.

Figure 37 shows the flow including the user responsibilities.

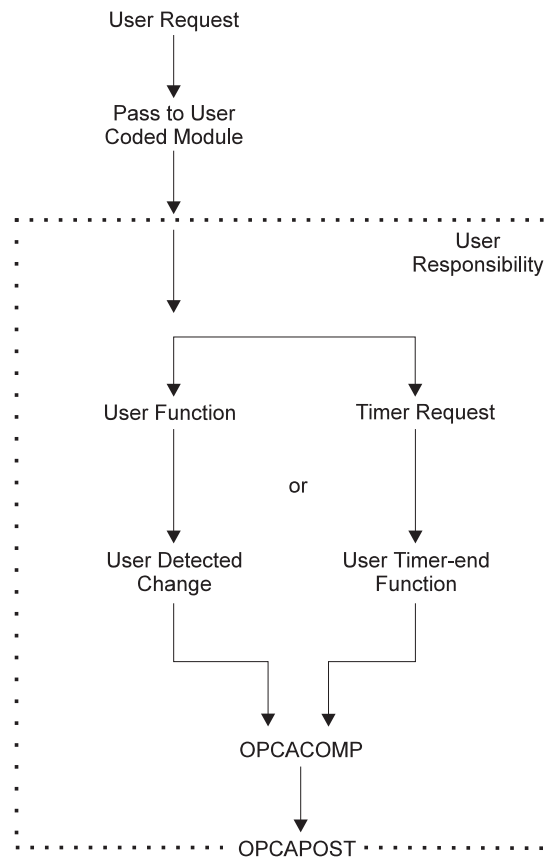


Figure 37. Request Flow for a Subsystem-Related User Function

To simplify implementation, you may plan to only use the timer function or only the detection of the completion of the command. If you use only the event-driven method, then consider what happens if the anticipated event fails to occur.

Implementing Completion of a Request

The general mechanism for executing requests for subsystems that have an OPCA entry coded in their MESSAGES/USER DATA policy item is as follows:

1. The request (with one or two optional parameters) and the job name of the OPC operation are passed to OPC Automation.
2. OPC Automation identifies the SA OS/390 definition of the subsystem through the job name of the OPC operation.

3. OPC Automation retrieves the expected result, the timer interval, and possibly a timer name, for the respective request/parameter combination from the OPCA entry of the subsystem.
4. OPC Automation then checks if an OPCAPARM entry is present and if that entry contains a timer module name.
5. If a timer module is specified, OPC Automation sets the timer with this timer module. Otherwise, it uses a standard timer module (EVJESPTE).
6. OPC Automation issues the command that is specified in the OPCACMD entry.

After the command has been issued, two things remain to be done:

- If the command was executed successfully, the status file record for this subsystem must be updated.
- The (positive or negative) result of the request must be posted back to OPC.

How this is done, depends on the type of command specified in the OPCACMD entry.

Using OPC Automation Standard Modules

If the command specified in the OPCACMD entry triggers a status change of the subsystem to RUNNING, UP or CTLDDOWN (no matter whether it is a user-written command or an SA OS/390 standard command), you can leave it to OPC Automation to perform these two tasks. The modules responsible for this are the status change module (EVJESPSC) and the standard timer module (EVJESPTE), already mentioned in step 5 above; two flags, a completion flag and a timer flag, ensure that both modules are not active at the same time. The two standard modules operate as follows:

1. The *status change module* is called whenever the status of the subsystem changes. It first checks the timer flag. If that flag is set, EVJESPSC terminates at once. If the flag is not set, EVJESPSC checks if the status change is the result of an OPC request, and if the new status is identical to the expected result as specified in the OPCA entry. When both conditions are satisfied, the status change module assumes that the request was successfully executed and
 - a. purges the timer defined in the OPCA entry,
 - b. sets the completion flag in the status file record for this subsystem,
 - c. actualizes further fields of the status file record, and
 - d. posts the result of the request back to OPC.
2. The *timer module* (EVJESPTE) is called after the timer set in step 5 has expired (except when you have specified your own timer module in the OPCAPARM entry). It first checks the completion flag. If that flag is set, EVJESPTE will terminate at once. If the completion flag is not set, EVJESPTE sets the timer flag and compares the current state of the subsystem with the expected result of the OPCA entry. If both are compliant, the timer module assumes that the request was executed successfully; it updates the status file accordingly and posts a positive result back to OPC. If they are not compliant, it only posts the failure of the request back to OPC.

Programming your own Completion Routines

If you specify a command in the OPCACMD entry that does not change the status of the subsystem to UP, RUNNING or CTLDDOWN, then you cannot use the standard modules for completing the request. In this case, you must perform the update of the status file and the posting of the result to OPC. You can do that in the command module (specified in the OPCACMD entry) or in a user-written timer module (specified in the OPCAPARM entry) or in both. The user-written timer module is called by OPC Automation in the following format:

*MODULE_NAME subsystem_name expected_result OPC_application_ID
OPC_workstation_ID request_sequence_number*

To simplify completion of a user-written module, OPC Automation provides the following facilities.

The OPCACOMP Command: This command, which is described in more detail in “OPCACOMP” on page 86, updates the status file and posts the result of the request back to OPC.

In particular, OPCACOMP first checks if the timer flag is set. If so, it will terminate at once. If not, it will

1. set the completion flag in the status file record of the subsystem for which it is called,
2. actualize further fields of the status file record, and
3. post the result of the request back to OPC.

The main difference between OPCACOMP and the standard modules (EVJESPSC and EVJESTPE) is that OPCACOMP does not check if the current status of the subsystem is in agreement with the expected result. Rather, it requires the (positive or negative) result of the request as one of its input parameters, and usually simply forwards this result to OPC. Thus, a user-written module must itself decide whether or not the request was executed successfully. It can then pass that information to OPCACOMP in order that the request be completed in an orderly manner.

One of the input parameters for OPCACOMP is the sequence number of the current request (see “OPCACOMP” on page 86). OPC Automation provides this and other information in some task global variables. Note, however, *that it will do this only when you have specified a timer module in the OPCAPARM entry.* You can specify a user-written timer module or the EVJESPTE standard module in the OPCAPARM entry. The following section describes the information contained in the global variables.

The &EHKVAR7, &EHKVAR8, and &EHKVAR9 Variables: When you supply a timer check module in the OPCAPARM entry (third value of the **Value Returned** field) OPC Automation sets some task global variables as follows:

&EHKVAR7 This variable contains the expected status, the timer interval, and the timer id as specified in the OPCA entry. The values are separated by commas.

&EHKVAR8 This variable contains the string ‘OPC’.

&EHKVAR9 This value contains the subsystem name, the sequence number, the name of the timer check module and the domain, separated by commas. This is also known as the Requestor ID block; see “Requestor ID Block (&EHKVAR9)” on page 94.

Do not modify the information in the task global variables. OPC uses information in &EHKVAR7 if the timer is purged. The SA OS/390 problem determination uses information in &EHKVAR8. In order to call OPCACOMP, the sequence number of the current request must be known; this number is stored in &EHKVAR9.

Non-Subsystem Operations

Operations of this type, containing requests named UXxxxxxx, allow you to perform commands that are independent of a specific subsystem. Figure 38 shows the flow for these types of operations.

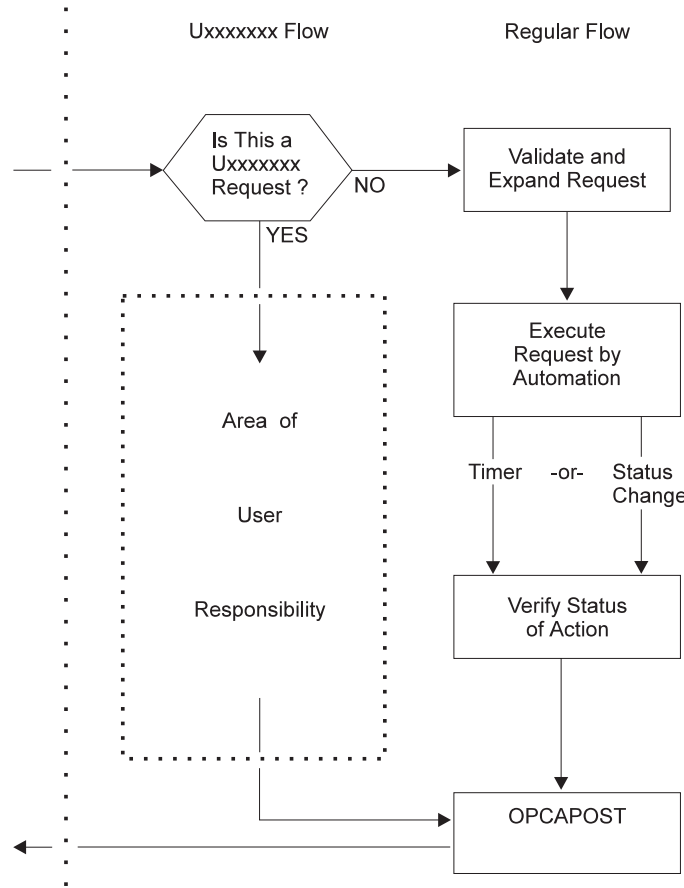


Figure 38. User Exit UXxxxxxx Flow

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 39 on page 102 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.

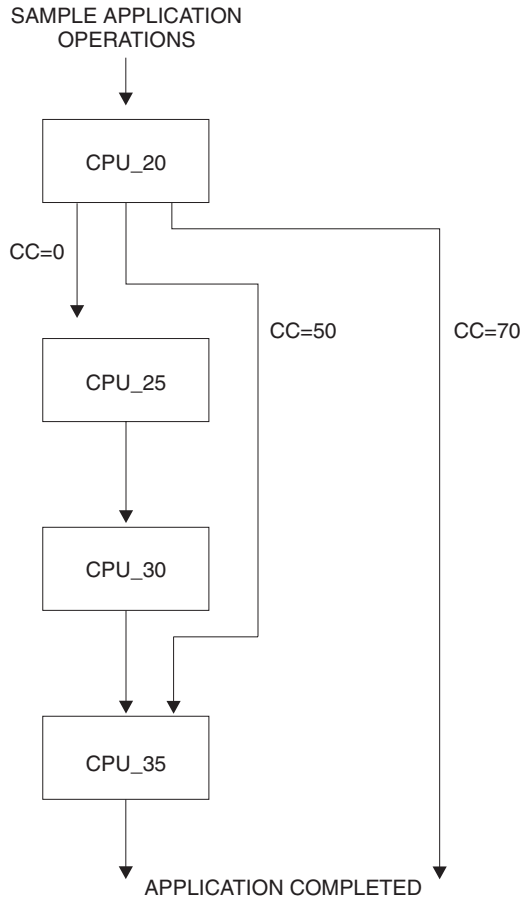


Figure 39. Condition Code Driven Application Flow

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 (UXxxxxxx) request. This request would pass to the specified NetView to a user-written task. This task could then use the OPCAMOD command to do a modify current plan to OPC for the application in question on the basis of the condition code received as part of the user exit request.

Flow of Control

When the name of a request starts with UX, OPC Automation assumes that the request is not related to a subsystem known to SA OS/390. As before, it expects to find an OPCACMD entry within a policy object that is identified through the **Job name** field of the OPC operation. But there are two differences as compared to subsystem-related requests:

- The only keyword that is needed is OPCACMD. OPCA and OPCAPARM are ignored. The CMD attributes of the OPCACMD entry should have the following format:

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

For &EHKVAR1, see "Parameters Passed to a User Exit" on page 103.

- The policy object identified through the **Job name** field of the OPC operation should be a USER E-T pair (see "OPCACMD" on page 74).

For non-subsystem requests, OPC Automation immediately tries to issue the command specified in the OPCACMD entry. After issuing the command, the request module of OPC Automation terminates. It is up to the user function to determine whether or not the request was executed successfully. The user function should then call OPCAPOST (see “OPCAPOST” on page 92) with the corresponding completion code. This returns the control of the application processing to OPC. The samples contain a code template for a non-subsystem command (EVJERUX1).

Parameters Passed to a User Exit

When the request name begins with UX OPC Automation stores the complete request buffer in the &EHKVAR1 task global variable. This variable must be forwarded to the command as an input parameter, as indicated in the format description above.

In contrast to a subsystem operation, the request buffer for a non-subsystem operation contains the entire request in one field. For the request buffer in general, see “Request Buffers and OPC Automation Log Entries” on page 15; the format of the request buffer for ‘UX’ requests is described in Table 6 on page 95.

Interaction with CICS Automation

The following example shows how to use the CEMTPPI command of CICS Automation to open and close CICS files. The CEMTPPI command allows you to perform CEMT commands on any CICS subsystem. If CICS Automation is not installed, then you can perform a similar function using the MVS MODIFY command from a NetView CLIST. First, you need these requests:

UXCICSOP Requests CICS to open a file.

UXCICSCL Requests CICS to close a file.

The example selects the CLIST names of CICSOPEN and CICSCLAS. Using these names, the format of the CMD attributes of the OPCACMD entry (see “OPCACMD” on page 74) is as follows:

```

COMMANDS  ACTIONS  HELP
-----
          UET Keyword-Data Specification          Row 3 from 3
Command ==>          SCROLL==> PAGE

Entry Type : User E-T Pairs          PolicyDB Name : SCENARIO
Entry Name : NONSUBS                 Enterprise Name : TEST
UET Entry  : DUMMY                   UET Type      : OPCACMD

Action  Keyword/Data(partial)
-----  -----
      CMD
      (UXCICSOP,, 'CICSOPEN &EHKVAR1')

      CMD
      (UXCICSCL,, 'CICSCLOS &EHKVAR1')
***** Bottom of data *****

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

```

Figure 40. OPCACMD Entry for Interaction with CICS

Figure 41 shows the definition of the operation text and other fields in OPC.

```

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

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

Application          : PAYMAINT          Payroll Master Update

Row  Oper      Duration Job name  Operation text
cmd ws  no.   HH.MM
''' NV04 015   0.01   DUMMY__  UXCICSCL CICS01 PAYROLL__

```

Figure 41. Defining Sample CICS Application in OPC

The example uses the CICS subsystem name and the file name as parameters to the request. These parameters are optional and flexible. Thus, the CICS name could also be passed through the **Job name** field. The REXX code for CICSOPEN and CICSCLOS is supplied in the samples as EVJRUX2 and EVJERUX3.

Interaction with IMS Automation

The following example shows how to use the IMSCMD of IMS Automation to start and stop databases in IMS. The IMSCMD command allows you to perform IMS MTO commands on any IMS in the system. Other IMS commands could be imbedded into IMSCMD and incorporated in NetView CLISTs you write yourself, using similar logic to that shown in the EVJERUX4 and EVJERUX5 CLISTs supplied with the samples. If IMS Automation is not installed, then you can

perform similar function by replying to the outstanding reply ID of the IMS you wish to communicate with from a NetView CLIST you write yourself. First, you will need these requests:

UXIMSSDB Requests to start a database.

UXIMSPDB Requests to stop a database.

The example selects the CLIST names EVJERUX4 and EVJERUX5. Using these names, the format of the CMD attributes of the OPCACMD entry (see "OPCACMD" on page 74) is as follows:

```

COMMANDS  ACTIONS  HELP
-----
Command ==>          UET Keyword-Data Specification          Row 3 from 3
                        SCROLL==> PAGE

Entry Type : User E-T Pairs          PolicyDB Name : SCENARIO
Entry Name : NONSUBS                 Enterprise Name : TEST
UET Entry  : DUMMY                    UET Type      : OPCACMD

Action  Keyword/Data(partial)
-----  -----
      CMD
      (UXIMSSDB, 'EVJERUX4 &EHKVAR1')

      CMD
      (UXIMSPDB, 'EVJERUX5 &EHKVAR1')
***** Bottom of data *****

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

```

Figure 42. OPCACMD Entry for Interaction with IMS

Figure 43 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 list
enter the GRAPH command to view the list graphically.

Application          : CUSTMAINT          Customer DB update

Row  Oper      Duration Job name  Operation text
cmd  ws  no.  HH.MM
'''  NV01 020   0.02   DUMMY__  UXIMSSDB IMS05Z_____

```

Figure 43. Defining Sample IMS Application in OPC

The parameters of the request are the IMS subsystem name and the database name. The REXX code of EVJRUX4 and EVJERUX5 is supplied in the samples.

Part 3. Using OPC Automation

This part describes the tasks of an OPC Automation operator, explains the commands available to the operator, and discusses some resynchronization and recovery scenarios.

Chapter 9. Operator's Guide

This chapter describes OPC Automation typically carried out by the operations staff.

How the Operator Uses OPC Automation

The OPC Automation operator performs two basic tasks:

- Monitoring the health of the complex using the Status Display Facility
- Responding to alerts or other error conditions that are not resolved by OPC, SA OS/390, or OPC Automation.

OPC Automation and SA OS/390 work by capturing messages and taking actions based on message contents. Some of those actions are coded by the programmer when OPC Automation and SA OS/390 are installed and implemented.

Messages fall into the following two categories:

- Problems encountered by OPC
- OPC-controlled applications

To ensure that an operator is made aware of the event, OPC Automation uses the Status Display Facility to show the status of the systems, subsystems, and applications.

Monitoring the Health of the System

OPC Automation provides enhancements to the SA OS/390 Status Display Facility so that production-control environment information is shown with operational environment information. This gives you a comprehensive view of your installation.

These enhancements include the following:

- Tape mounts
- Batch job monitoring
- TSO user monitoring
- Critical message alerting
- OPC ended-in-error alerting

To use the Status Display Facility, type **SDF** and then press ENTER. See "SDF—Status Display Facility Panel" on page 119 for details on using the Status Display Facility.

Notes:

1. The panel examples shown in this section are the default panels provided with OPC Automation. Your system programmer may have changed these panels during implementation. This is an option that OPC Automation provides so that you can tailor your panels to your specific needs.
2. The Status Display Facility requires that the terminal from which the command is entered support 3x79 terminal extended attributes.
3. The definitions of color and highlighting depend on how SA OS/390 and OPC Automation are implemented at your location.
4. To fully understand the Status Display Facility, refer to the appropriate section in the *System Automation for OS/390 Customizing and Programming*.

Problems in an OPC-Defined Application

If an application under OPC control ends in error, OPC issues a message which NetView intercepts. OPC Automation uses the message to update the Status Display Facility panel. The subsystem name turns the appropriate color, probably red, which is the default.

Note: When the problem is in OPC (the Controller, the Tracker, or both), the subsystem fields change color.

You start problem determination from the OPC Automation Monitor panel (for more details on what the fields in this panel mean, see “OPC Monitor Panel” on page 120):

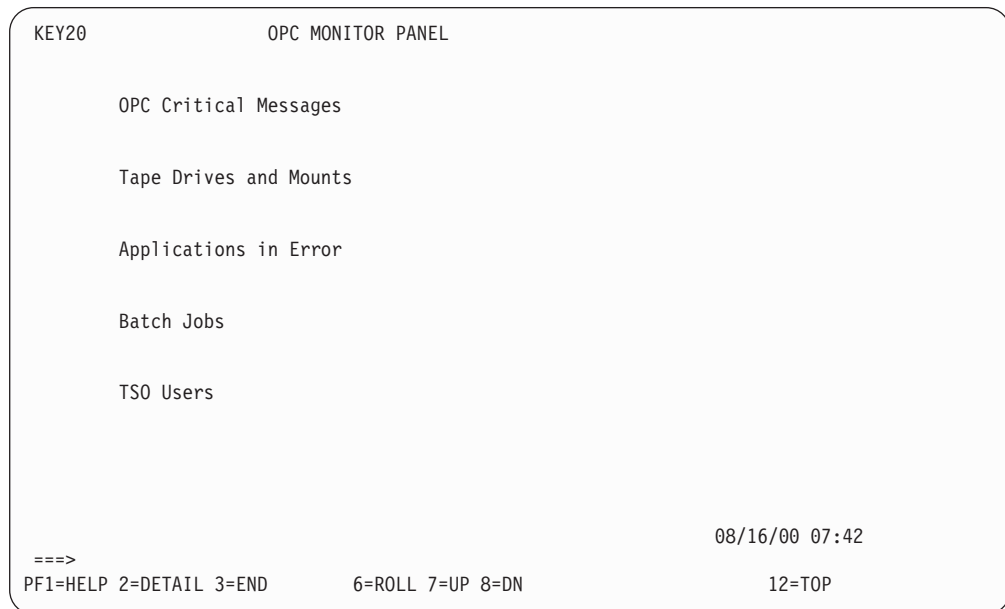


Figure 44. OPC Automation Monitor Panel

For this scenario, the **Applications in Error** field has changed to an alert-condition color. This means that an error has occurred in an OPC-defined application. Place the cursor under **Applications in Error**. After you press PF8, OPC Automation displays the **Applications in Error** panel.

The **Applications in Error** panel, as shown in Figure 45 on page 111, lists the OPC application or applications that initiated the error. See “OPC Automation: Applications in Error Panel” on page 121 for more discussion of this panel.

```

KEY20PC1
Applications in Error

RMFDLY_RMFMAINT
APPL2_25

08/17/00 08:21
===>
PF1=HELP 2=DETAIL 3=END      6=ROLL 7=UP 8=DN      10=LF 11=RT 12=TOP

```

Figure 45. OPC Automation: Applications in Error Panel

Now that you know which application has failed, you are ready to determine your actions. Type **OPCACMD** on the command line of any NetView panel. After you press ENTER, OPC Automation displays the **SA/OPC – Display or Modify OPC Data** panel (EVJKAC01), as shown in Figure 46.

```

EVJKAC01      SA/OPC - Display or Modify OPC Data
Date: 08/17/00
Time: 08:25:34

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=Ro11

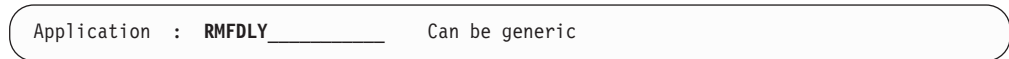
```

Figure 46. SA/OPC – Display or Modify OPC Data Panel

In cases where the reported problem concerns an OPC-defined application, you have the option of logging on to OPC or using the OPCACMD command to access panels that allow you to browse or change the application information from OPC

Automation. Using OPCACMD has the advantage of allowing you to remain in NetView and to jump between the OPC information and other NetView components.

You can enter the search criteria for the application directly onto this panel. Since RMFDLY_RMFMMAINT is the failing application and job, type **RMFDLY** in the application field, as shown in Figure 47.



Application : **RMFDLY**_____ Can be generic

Figure 47. Entering MAINT in the Application Field

Note: You can enter other parameters onto the OPCACMD panel, such as job name and operation number. As in this example, the application does not need to be in error to be accessible by the OPCACMD module.

After you press ENTER, OPC Automation displays the **SA/OPC – Display or Modify OPC Data** panel (EVJKAC03), as shown in Figure 48.

```

EVJKAC03      SA/OPC - Display or Modify OPC Data
                                                    Date: 08/17/00
                                                    Time: 08:26:11
CMD: C change  B browse
      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  0005262230 STOP      N   N
b   RMFDLY            RMFMAINT CPU1 0002 E  0005262230      N   N
_   RMFDLY            RMF      NV11 0003 W  0005262230 START     N   N
_   RMFDLY            RMF      NV11 0001 A  0005302200 STOP     N   N
_   RMFDLY            RMFMAINT CPU1 0002 W  0005302200      N   N
_   RMFDLY            RMF      NV11 0003 W  0005302200 START     N   N
_   RMFDLY            RMF      NV11 0001 A  0006062200 STOP     N   N
_   RMFDLY            RMFMAINT CPU1 0002 W  0006062200      N   N
_   RMFDLY            RMF      NV11 0003 W  0006062200 START     N   N

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

Figure 48. Display or Modify OPC Data Panel

Figure 48 presents the operations in the requested application. The panel displays detailed information about a specific operation, enabling you to review the information or to make changes. The operation with the RMFMAINT batch job shows that the status is ended in error (E).

To obtain additional information on the type of error, type **B** in the CMD field, as shown in Figure 49.

```

B RMFDLY      RMFMAINT      CPU1 0002 E      08/17/00 08:26:51
  
```

Figure 49. Entering B (Browse) in the CMD Field

After you press ENTER, OPC Automation displays the **SA/OPC – OPC Occurrence Data** panel (EVJKAC02).

```

EVJKAC02          SA/OPC - OPC Occurrence Data
                                                    Date: 08/17/00
                                                    Time: 08:28:33

Subsystem   : OPCC           Deadline  : 00/08/25 23:20
Application : RMFDLY        OPIA date : / /      OPIA time :   :
Opno       : 0002          Edur     : 0005      PS reqd  : 0001
Inp arrival : 00/08/25 22:20 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 50. SA/OPC – OPC Occurrence Data Panel

Note: From the panel in Figure 50, you can determine that the problem is an SB37 abend. Assume that the conditions causing this abend are fixed and that the recovery is to rerun the RMFMAINT job.

Press PF3 to return to the previous panel. This time type **C** in the CMD field, as shown Figure 51.

```

C RMFDLY      RMFMAINT      CPU1 0002 E      08/17/00 08:30:07

```

Figure 51. Entering C (Change) in the CMD Field

The resulting panel is similar to the one in Figure 48 on page 113, except that you can overwrite the fields with new information since this panel is in change mode rather than browse mode. Recovery consists of restarting the application at this operation. To do this, you overwrite the **Status** field with an R, indicating a rerun request, and press ENTER. OPC Automation then examines the changes and sends them to OPC through the OPC API. The bottom left of the panel is updated with a message indicating that OPC has accepted the request or that there is an error.

Although the above scenario is a simple one, you can handle more complex situations without logging on to OPC to perform the actions. You can only use this interface to fix existing problems. You cannot use it to define new applications or workstations. If these functions are needed, you are required to log on to OPC directly, from TSO/ISPF.

Note: The OPCACMD command and this panel are described in detail in “OPCACMD—Interacting Dynamically with OPC” on page 129.

Chapter 10. OPC Automation Operator Commands

The following commands are used with OPC Automation:

Table 7. OPC Automation Commands

| Command | Description |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPCA | Displays the OPC Automation Main Menu (EVJT0000) that lists the most commonly used commands and provides access to the tutorials. See “OPC Automation Main Menu and Tutorials” on page 118. |
| SDF | Accesses the Status Display Facility. This is actually an SA OS/390 command. However, sample Status Display Facility panels are supplied with OPC Automation. This includes the following changed or additional panels: <ul style="list-style-type: none">• Status Display Facility panel• OPC Automation: TSO Users Status panel• OPC Automation: Batch Job Display panel See “SDF—Status Display Facility Panel” on page 119. |
| OPCACMD | Displays the OPC Automation: Display or Modify OPC data (EVJKAC01) panel that is used to interact dynamically with OPC. See “OPCACMD—Interacting Dynamically with OPC” on page 129. |
| OPCAQRY | Lists pending NetView operations. See “OPCAQRY—Display Status of Operations” on page 131. |
| OPCAPOST | Posts an operation in OPC from SA OS/390. See “OPCAPOST—Posting an OPC Operation from SA OS/390” on page 134. |
| SRSTAT | Determining status of OPC special resources. See “SRSTAT—Determining OPC Special Resource Status” on page 135. |
| EVJESPIN | Normally used only during initialization by SA OS/390. The operator can use this command manually to create a new status file record or resynchronize an existing one. The actions available with this command are INIT, RESET, SYNC, and CREATE. See “EVJESPIN—Initialization” on page 136. |

OPC Automation Main Menu and Tutorials

Type **OPCA** on the command line. After you press ENTER, OPC Automation displays the **SA/OPC – Main Menu**, as shown in Figure 52.

| EVJT0000 | | SA/OPC - MAIN MENU | |
|----------|----------|--------------------------------|----------|
| TYPE | COMMAND | DESCRIPTION | TUTORIAL |
| | | SA/OPC Automation | 1 |
| P | SDF | Display facility additions | 2 |
| P | OPCAQRY | Display status of operations | 3 |
| L | OPCAPOST | Manually post status to OPC | 4 |
| P | OPCACMD | Display and modify OPC data | 5 |
| L | SRSTAT | Update special resource status | 6 |

Note: Commands of type L (linemode) do not have an input panel and must be invoked with all required parameters specified.

Enter a Tutorial Number or Command ==>
PF1= Help PF2= End PF3= Return
PF6= Roll

Figure 52. SA/OPC – Main Menu

To obtain information on using this panel, enter the tutorial number adjacent to the command.

SDF—Status Display Facility Panel

To use the Status Display Facility, type **SDF** and then press ENTER. You will see a display similar to the **SA OS/390 Support Systems** Status Display Facility panel in Figure 53..

| SYSTEM | | | | | | SA OS/390 - SUPPORT SYSTEMS | | | | | |
|--------|------------|----------|----------|----------|-------------|-----------------------------|--|--|--|--|--|
| System | Subsystems | WTORs | Gateways | Products | System | | | | | | |
| KEY1 | IMS712M1 | IM631C4 | IPUFMI | C I D O | P V M B T U | | | | | | |
| KEY2 | CICSK4D | NETATST2 | IPSFNO | C I D O | P V M B T U | | | | | | |
| KEY3 | | | IPSF00 | C I D O | P V M B T U | | | | | | |
| KEY4 | | | | C I D O | P V M B T U | | | | | | |

08/16/00 07:37

===>

1=HELP 2=DETAIL 3=RETURN 6=ROLL 8=NEXT SCR 10=LEFT 11=RIGHT 12=TOP

Figure 53. SA OS/390 Support Systems—Status Display Facility Panel

Use the SA OS/390 Support Systems panel, shown in Figure 53, as follows:

- Tab to a system name and press PF8 to display the next level of detail.
- Press PF2 to display the detail screen with the highest level of priority for that system.
- Tab to the **O** in the same row as the system and press PF8. You will go to the OPC Automation Monitor Panel, as seen in Figure 54 on page 120.

OPC Monitor Panel

If you press PF8 to get a “down” panel from the “O” field on the initial Status Display Facility panel, you will see the **OPC MONITOR PANEL**. Figure 54 shows this panel.

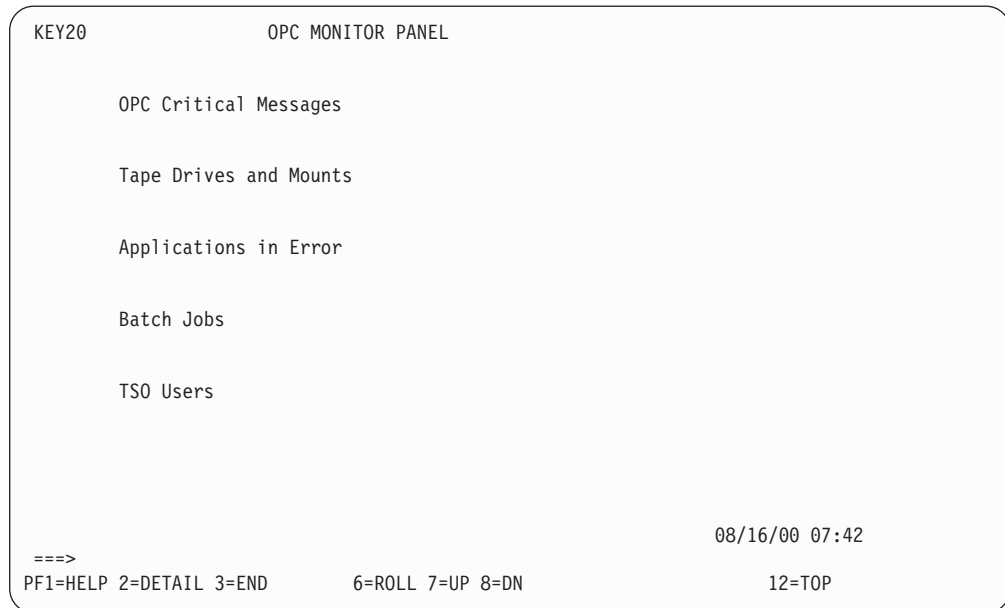


Figure 54. OPC Monitor Panel

The following fields and functions are added with OPC Automation:

- | | |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPC Critical Messages | Critical messages. Place the cursor on this field and press PF8 to display the screen described in “Critical Messages Panel” on page 122. |
| Tape Drives and Mounts | Tape mount request and tape units varied online. Place the cursor on this field and press PF8 to display the screen described in “Tapes Panel” on page 124. |
| Applications in Error | OPC operations ended-in-error alerts. Place the cursor on this field and press PF8 to display the screen described in “OPC Automation: Applications in Error Panel” on page 121. |
| Batch Jobs | Batch jobs executing. Place the cursor on this field and press PF8 to display the screen described in “Batch Display Panel” on page 128. |
| TSO Users | TSO users logged on. Place the cursor on this field and press PF8 to display the screen described in “TSO Users Status Panel” on page 126. |

OPC Automation: Applications in Error Panel

If you select **Applications in Error** from the **OPC MONITOR PANEL** shown in Figure 54 on page 120, OPC Automation displays the **Applications in Error** panel, as shown in Figure 55.

```
KEY20PC1
          Applications in Error

RMFDLY_RMFMMAINT
APPL2_25

                                08/17/00 08:21
===>
PF1=HELP 2=DETAIL 3=END      6=ROLL 7=UP 8=DN      10=LF 11=RT 12=TOP
```

Figure 55. Applications in Error Panel

This panel presents you with all the OPC applications that have ended in error, regardless of whether they have operations which use OPC Automation. If the operation that ended in error is a batch job, the name of the job is provided after the underscore. In the first example above, the operation ended in error is a job with a name of RMFMMAINT in an application with the name of RMFDLY. In the second example, operation 25 of the application APPL2 has ended in error. Because a numeric name is displayed, this operation is not a batch job.

Critical Messages Panel

If you select **OPC Critical Messages** from the **OPC MONITOR PANEL** shown in Figure 54 on page 120, OPC Automation displays the **CRITICAL MESSAGES** panel, as shown in Figure 56.

Note: These message must have been selected by your systems programmer for display.

```
KEY20CM1                CRITICAL MESSAGES
NAME      MESSAGE TEXT
IEF450I   JOB CICS06CB - IEF450I CUCS06CB CICS06CB - ABEND=S22 U0000 REASON=0
IEF450I   JOB VERA - IEF450I VERA STEPISPF USERPROC - ABEND=S622 U0000 REASON=0

                                08/17/00 09:21
===>
PF1= HELP  2= DETAIL  3=END      6=ROLL  7=UP      10=LF  11=RT  12=TOP
```

Figure 56. Critical Messages Panel

Press PF2 to list the critical messages in the system. OPC Automation displays the Critical Message—Detail Status Display, as shown in Figure 57 on page 123.

```
----- DETAIL STATUS DISPLAY -----
                                     1 OF 1

COMPONENT: IEE450I                SYSTEM : KEY1
COLOR   : GREEN                   PRIORITY : 503
DATE    : 08/16/00                TIME    : 16:09:59
REPORTER : GATAOF06              NODE    : AOF06
DUPLICATE COUNT: 1

JOB CICS06CB - IEF450I CICS06CB CICS06CB - ABEND=S222 U0000 REASON=0000

===>
PF3=RET 4=FPI 6=ROLL 7=UP 8=DN 9=AST 10=DEL 11=BOT 12=TOP
```

Figure 57. Critical Message—Detail Status Display

Your system administrator selects the messages on the Critical Message—Detail Status Display and determines whether they are critical. They remain in the system until the automation NetView is shut down, or you manually delete them using PF10.

Tapes Panel

If you select **Tape Drives and Mounts** from the **OPC MONITOR PANEL** shown in Figure 54 on page 120, then OPC Automation displays the **Tape Unit Display** panel, as shown in Figure 58.

```
KEY20TP1          Tape Unit Display
TAPE UNITS ONLINE
  350_3480   351_3400   352_3480   253_3480

TAPE MOUNTS
  350_KMX994   351_EXP723

                                08/17/00 09:21
===>
PF1= HELP  2= DETAIL  3=END      6=ROLL  7=UP      10=LF  11=RT  12=TOP
```

Figure 58. Tape Unit Display Panel

TAPE UNITS ONLINE lists the online tape units and their model numbers. TAPE MOUNTS displays outstanding tape mount requests. To use this panel, move the cursor to the specific tape mount and press PF2 to display detail information.

For example, if you request 350_KMX994, the **DETAIL STATUS DISPLAY** is displayed as shown in Figure 59.

```
----- DETAIL STATUS DISPLAY -----
                                     1 OF 1

COMPONENT: 350_KMX994          SYSTEM : KEY1
COLOR   : PINK                 PRIORITY : 430
DATE    : 08/16/00            TIME     : 16:45:10
REPORTER : AUTO1              NODE     : A0FS6
REFERENCE VALUE : 350_KMX994

IEF223A 16:45 VOLUME KMX994 REQUIRED ON DEVICE 350

===>
PF3=RET 4=FPI 6=ROLL 7=UP 8=DN 9=AST 10=DEL 11=BOT 12=TOP
```

Figure 59. Tape User—Detail Status Display

Note: Mount requests that are not satisfied in two minutes turn to reverse video red.

TSO Users Status Panel

If you select **TSO Users** from the OPC Automation Monitor panel shown in Figure 54 on page 120, then OPC Automation displays the **TSO Users Status** panel shown in Figure 60.

The panel in Figure 60 shows all currently logged-on TSO users. As users log off,

```
KEY20TS1          TSO Users Status

      OPER1          NISTMPL
      NISTMPS        JILL
      DAVE           JAMES
      NISTMPK        SDL

                                08/17/00 09:45
===>
PF3=RET  4=FPI  6=ROLL  7=UP  8=DN  9=AST 10=DEL 11=BOT 12=TOP
```

Figure 60. TSO Users Status Panel

they are deleted from the screen. To use this panel, move the cursor to the specific user and press PF2 to display detail information.

For example, if you request NISTMPL, OPC Automation displays the TSO User—Detail Status Display, as shown in Figure 61.

```
----- DETAIL STATUS DISPLAY -----
                                     1 OF 10

COMPONENT: NISTMPL          SYSTEM  : KEY1
COLOR   : GREEN            PRIORITY : 550
DATE    : 08/17/00        TIME    : 08:22:06
REPORTER : AUTO1          NODE    : A0FS6
REFERENCE VALUE : NISTMPL

IEF125I 08:22 TSO USER NISTMPL LOGGED ON 08/17/00 AT 08:22

==>
PF3=RET 4=FPI 6=ROLL 7=UP 8=DN 9=AST 10=DEL 11=BOT 12=TOP
```

Figure 61. TSO User—Detail Status Display

Batch Display Panel

If you select Batch Jobs from the OPC Automation Monitor panel shown in Figure 54 on page 120, then OPC Automation displays the **Batch Job Display** panel as shown in Figure 62.

```
KEY20BT1      Batch Job Display
              SLRCOLL
              PLI
              BLDFPI
              COMPRESS
              COLEY7
              CICSXA1
              APPLYPTF
              TAPEDMP
              EREP08
              NSC08B
              REMMAINT
              08/17/00 10:25
              ==>
PF3=RET 4=FPI 6=ROLL 7=UP 8=DN 9=AST 10=DEL 11=BOT 12=TOP
```

Figure 62. Batch Job Display Panel

The panel in Figure 62 shows all batch jobs active in the selected system. As jobs complete they are deleted from the screen. To use this panel, move the cursor to the specific job and press PF2 to display detail information. For example, requesting PLI displays the Batch Jobs—Detail Status Display shown in Figure 63.

```
----- DETAIL STATUS DISPLAY -----
                                           1 OF 11
COMPONENT: PLI                          SYSTEM : KEY1
COLOR   : GREEN                          PRIORITY : 550
DATE    : 08/17/00                        TIME     : 10:11:01
REPORTER : AUTJES                          NODE     : A0FS6
REFERENCE VALUE : PLI
IEF403I 10:11 : SUBSYSTEM ENTRY JOB PLI STARTED ON 08/17/00 AT 10:11.
              ==>
PF3=RET 4=FPI 6=ROLL 7=UP 8=DN 9=AST 10=DEL 11=BOT 12=TOP
```

Figure 63. Batch Jobs—Detail Status Display

OPCACMD—Interacting Dynamically with OPC

The primary use of the OPC SA OS/390 interface is to recover operations which could not proceed because of communications disruptions and to recover OPC-detected errors which require physical intervention. A NetView operator can use this interface for checking the status and progress of an application without logging on to OPC.

To access the OPCACMD panels, type the following on any NetView command line:

OPCACMD

After you press ENTER, OPC Automation displays the SA/OPC – Display or Modify OPC Data panel (EVJKAC01), as shown in Figure 64.

```
EVJKAC01          SA/OPC - Display or Modify OPC Data
                                                    Date: 08/17/00
                                                    Time: 08:25:13

Specify search criteria and press ENTER

  Subsystem      : OPCC                From ACF-file
  Application    : day*_____        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=Ro11
```

Figure 64. SA/OPC - Display or Modify OPC Data Panel

For a detailed example of using OPCACMD, see “Problems in an OPC-Defined Application” on page 110.

After you press ENTER, OPC Automation displays the Display or Modify OPC Data panel (EVJKAC03), as shown in Figure 65. This panel shows the operation summary.

```

EVJKAC03          SA/OPC - Display or Modify OPC Data
                                                    Date: 08/17/00
                                                    Time: 08:27:03
CMD: C change  B browse
      H hold   R release N no-op  U unno-op
CMD  Application      Jobname  Ws  Opno St  Inp. Arr  Description  H   N
-----
-   DAYRMF            RMF      NV11 0001 C  0007162200 STOP      N   N
-   DAYRMF            RMFMaint CPU1 0002 E  0007162200      N   N
B   DAYRMF            RMF      NV11 0003 W  0007162200 START      N   N

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

```

Figure 65. SA/OPC - Display or Modify OPC Data Panel (Operation Summary)

After you type **B** (browse) in the CMD field, as shown in Figure 65, OPC Automation displays the SA/OPC – OPC Occurrence Data panel, as shown in Figure 66. This panel shows operation details.

```

EVJKAC02          SA/OPC - OPC Occurrence Data
                                                    Date: 08/17/00
                                                    Time: 08:28:21

Subsystem : OPCC          Deadline : 00/08/16 23:30
Application : DAYRMF      OPIA date : / /      OPIA time : :
Opno : 0003              Edur : 0005          PS reqd : 0000
Inp arrival : 00/08/16 22:30 Job Class :          R1 reqd : 0000
Jobname : RMF            Auto Sub : Y          R2 reqd : 0000
Wsname : NV11           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 : _____  DeadWTO : N          Cat mgt : N
Status : W               Manual hold : N       NOP : N
                          Description : START_____
                          User data : _____

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

```

Figure 66. Operation Detail Panels

OPCAQRY—Display Status of Operations

The OPCAQRY command displays the status of OPC Automation operations.

To use this command, type the following on any NetView command line:

OPCAQRY

After you press ENTER, OPC Automation displays the **SA/OPC Automation Operation Status Display** panel, as shown in Figure 67.

```
EVJKCGAA      SA/OPC - Automation Operation Status Display
Valid Actions: B Browse  D Delete  R Reset
Date: 08/17/00
Time: 11:43:00

Act Job      Application  Request   Date      Time      Status
-  CX06AA
-  DBSYS1 TEST2      STOP     05/24/00  08:44    No request
-  RMF      RMFMAINT   START    05/29/00  14:16    Complete
-  SUBSYS1
-  ***** ***** No request

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

Figure 67. SA/OPC - Operation Status Display Panel

The panel in Figure 67 shows the status of requests from OPC Automation in NetView. It also provides a convenient place to delete unused records or to reset an operation in the event of problems.

The fields shown on the panel in Figure 67 on page 131 are defined as follows:

- Act** Action field, used for browsing, deleting, or resetting the status file record. See "Selecting Actions" for more discussion on this topic.
- Job Name** Job name from OPC, typically used to represent a subsystem.
- Application** OPC application requesting the operation.
- Last requested action** Action specified in the OPC operation description text.
- Date** Date the request was received.
- Time** Time the request was received.
- Status** Status of the request in NetView, either complete, incomplete, timeout, or no request. A status of timeout indicates that the operation is marked in error because it did not complete within the time limit set by the system programmer in the opca code entry. A status of incomplete indicates that the operation did not achieve the expected status set by the system programmer in the same entry. Complete and no request are considered normal statuses.

Selecting Actions

Select one of the following valid actions for any operation listed on the OPC Automation Operation Status Display panel, as shown in Figure 67 on page 131.

- B** Browse. Selecting the browse action allows the user to examine a record on OPC Automation Operation Status panel as shown in Figure 68 on page 133.
- D** Delete. This action deletes the record from OPC Automation status file.

A confirmation pop-up will appear, as follows:

```
..... DELETE CONFIRMATION .....  
:                                     :  
:                                     :  
:          RECORD ID... TESTSYS      :  
:                                     :  
:   - ENTER 1 TO DELETE RECORD,      :  
:   - PRESS F3 OR F12 TO KEEP RECORD. :  
:                                     :  
:.....:                               :
```

- R** Reset. This clears flags in the record for reuse by performing an EVJESPIN CMD=RESET. This is useful for recovering operations that did not process normally. See "EVJESPIN—Initialization" on page 136 for more details.

After you type **B** (browse) in the Act field of the OPC Automation Operation Status Display panel shown in Figure 67 on page 131 and press ENTER, OPC Automation displays the OPC Automation Operation Status Detail panel, as shown in Figure 68.

```
EVJKCGAA          SA/OPC - Operation Status Display          Date: 08/17/00
Status file record display for CX06AA                        Time: 14:44:00

EHK170I OPCA RECORD DISPLAY FOR: CX06AA
EHK171I ID= CX06AA          , TYPE= OPCA, OPID= RICK
EHK172I LAST COMPLETED STATUS=  , LAST SEQUENCE NUMBER= 0000
EHK173I TIMER FLAG=  , COMPLETION FLAG=
EHK174I CURRENT SEQUENCE NUMBER= 0000, CHECK MODULE=
EHK175I EXPECTED STATUS=  , TIMER INTERVAL=  ,TIMER ID=
EHK176I ADNAME=  , WSNAME=
EHK177I OPNUM=  , JOBNAME=  , DATE= 08/17/00 ,TIME= 14:33
EHK178I REQUEST=  , PARM1=  , PARM2=
EHK002I END

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

Figure 68. OPC Automation: Operation Status Detail Panel

OPCAPOST—Posting an OPC Operation from SA OS/390

This command is used by SA OS/390 to inform OPC of status changes. This is accomplished by the OPCAPOST command processor, which is normally used internally in OPC Automation. Although you can issue OPCAPOST as an operator command, operators should use OPCACMD, if possible. OPCACMD provides a full-screen interface to OPC and dynamically acknowledges the action, rather than OPCAPOST.

If you determine that you must use the OPCAPOST command, refer to “OPCAPOST” on page 92.

SRSTAT—Determining OPC Special Resource Status

This command lets you manipulate the status of OPC special resources. Status is returned via messages. The format of this command is:

SRSTAT *sname*,**SUBSYS**=*subsys*,**AVAIL**=Y|N

sname

Special resource name — up to 44 characters.

subsys

Subsystem ID — 4 characters.

AVAIL=Y/N

Availability indicator.

Example:

```
SRSTAT EOD.CICSPRD1.TRANS,SUBSYS=OPCT,AVAIL=Y
```

In the above example, end-of-day transactions are required to complete before production work can begin. SRSTAT is executed when the transactions are complete. The special resource name EOD.CICSPRD1.TRANS is used to trigger OPC/ESA applications that are able to run when the transactions are finished. A number of applications are added to the current plan.

The variable used for *subsys*, OPCT, is the name of the tracker subsystem. Only in this command is the tracker subsystem name required.

EVJESPIN—Initialization

OPC Automation uses this command during initialization when SA OS/390 is started. An operator can also use it to create a new OPC Automation status file record or to resynchronize an existing one.

Four command actions are available. The first, INIT, acts on all OPC Automation controlled subsystems, while the other three are available for the specified subsystem only.

EVJESPIN *CMD=action,SUBSYSTEM=subsystem*

CMD=action The command to execute where *action* is one of the following:

INIT Forces an equivalent action to that taken during normal initialization. Do not specify a subsystem parameter for this parameter. To properly understand the action of this command parameter and for a complete overview of the initialization process, refer to the initialization topic in “Initialization Module (EVJESPIN)” on page 33.

RESET

Resets the timer and comp flags to a null value, and unlocks a specific subsystem after a user error is detected and corrected. By resetting the timer and comp flags, OPC Automation again accepts requests from OPC.

SYNC Checks the last completed status field in OPC Automation and ensures that the specified subsystem status agrees whether the last completed status is UP or CTLDOWN. OPC Automation resets the timer and comp flags to null. This marks the action as completed. Use SYNC when manual action is necessary to stop or start a subsystem.

CREATE

Creates a new OPC Automation status file record from the control file definition for a specified subsystem. Use this function to refresh the control file dynamically with new OPC Automation definitions, when NetView is not recycled.

subsystem The subsystem initialized. Do not specify a subsystem with INIT.

Chapter 11. Resynchronization and Recovery Considerations

OPC Automation combines the capabilities of two different subsystems, OPC and SA OS/390, which may reside over several systems. As a result, a failure or a scheduled interruption of services with one of the subsystems, processors, or telecommunications facilities may occur and prevent OPC Automation from processing operations. Further complications arise by shifting work load across multiple system images, either for scheduled work-load balancing or as part of a recovery situation.

In such a case, a loss of synchronization can occur between the OPC schedule and the OPC Automation components in NetView. When this happens, you may need a manual process to examine the OPC schedule, to ensure that OPC Automation in NetView is performing actions as required and, in some cases, to resynchronize OPC and OPC Automation.

During a loss of contact or a failure with either OPC or NetView, OPC Automation's facilities invoke and restore the environment as it was before the failure so that event scheduling can pick up where it left off. This results in a satisfactory resolution and manual resynchronization is not required. See "Automated Recovery Functions" on page 141 for additional information.

Generally, the longer the outage, the more likely that resynchronization is required. This depends on the number of scheduled events that are not processed. A long outage during the day may have a smaller synchronization impact than a shorter outage during a period when many online facilities are started or shut down.

Examples and Scenarios

This section describes possible scenarios for resynchronization and recovery.

Loss of Contact Between OPC and OPC Automation

Under most situations, once OPC Automation re-establishes connectivity, its automatic recovery schedules requests for execution that it could not execute prior to connectivity. In some situations you may need to intervene manually, such as when the request is no longer valid. The following sections discuss several reasons for manual intervention.

Taking Action Manually on the Target System

The scheduler acts on the ended-in-error notification, requesting that an operator with access to the target system perform the required operation manually. The operation requested by OPC completes, and the scheduler manually updates it to a completed status, enabling OPC to continue processing.

You should issue `EJESPIN CMD=SYNC`.

Taking Action Too Late

The remaining processing window is too small to allow an operation to occur. For example, an online system may require a certain amount of time to initialize. If this amount of time is close to the scheduled shutdown time, you probably should override the request and complete the operation manually.

You should issue `EJESPIN CMD=RESET`.

Queuing Several Actions for a Specific Target Subsystem

Rather than not having enough time for a system initialization as in the previous example, the outage may have lasted long enough for a specific subsystem to receive several queued operations that frequently conflict. For example, an online task may have a start request with an ended-in-error status because of connectivity problems. This same task may also have a stop request already due for scheduling. If you allow automatic recovery for this application, the subsystem would start, but an immediate shutdown would follow.

Complete these operations manually. You should issue `EJESPIN CMD=RESET`.

Backup on a Different Processor

If you perform a backup using a different processor, pay special attention to ensure that you properly restore the work load on the new system. Depending on the backup structure, you need to follow one of several different procedures discussed in the next sections.

Full Takeover onto a Standby System at the Same Site

This is the simplest type of backup. It becomes the same as a single-system recovery if the data is also available. OPC Automation uses the information in the status file to restore the various subsystems to the prebackup status.

Full Takeover onto an Standby System at a Different Site

If the status file is not available, restructure the environment manually. Examine pertinent applications that control this specific `NVnn` workstation. The last completed `NVnn` operation requires manual triggering. You can achieve this with the `OPCACMD` function, allowing the NetView operator direct access to the relevant applications. Although, at times, you may find it necessary to perform

specific operations manually, in most cases, resetting an operation with EVJESPIN or restarting an application produces the desired effect.

Takeover onto a Working System

In most situations, the takeover is onto a working system and is restricted to certain critical applications. The previously discussed considerations as to whether the system is at the same site or at another site apply to this situation. Since not all applications are restarted on the backup system, several new considerations become important. Certain applications need cancelling before you can achieve a restoration of services. Some situations can result in duplications, such as with subsystems like TSO, which are frequently found on every MVS system. Although normally this is controlled by SA OS/390, OPC Automation can control this. Here, duplicate applications need cancelling for the backup period.

During the backup period, use one of these two methods to run OPC Automation:

- Add the new work load to the resident NetView by changing the *NVnn* workstation entry in the platform control file to point to the same NetView domain ID.
- Start another copy of NetView with the NetView domain ID used in the failing system.

The consideration of which method to use becomes important once you restore the original configuration.

With the single NetView solution method, you need to resolve the subsystems manually since their original identity is lost. With the extra NetView solution method, you can stop the subsystems controlled by the extra NetView by shutting it down. This simplifies the restoration process, requiring almost no manual intervention.

In both cases, restoring the environment follows similar procedures used to backup a host onto a standby backup system.

Long Term Outage

You must manually intervene when the outage duration is more than a single scheduling cycle. This type of recovery is confusing since many applications are shown as late in the OPC plan. Carefully review these applications since some of them still need scheduling, while others need to be cancelled. For applications that need scheduling, certain operations involving the online portion need cancelling or holding. To ensure success, this type of recovery needs precise planning and monitoring. Otherwise, you can use the scenarios previously outlined.

Example Using Doubly-Defined NetView Domain IDs

The following example shows the WORKSTATION DOMAINS entry for a 4-processor environment:

```

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

Entry Type : Workstation domainID  PolicyDB Name  : SCENARIO
Entry Name  : SAMPLE_01            Enterprise Name : TEST

Subsystem  : OPCA_DOMAINID
Message ID : DOMAINID

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

Code 1      Code 2      Code 3      Value Returned
NV00 _____ _____ _____ NVTOR
NV01 _____ _____ _____ XBAOF
NV02 _____ _____ _____ AOFT5
NV06 _____ _____ _____ AOFS6
_____
_____

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

```

Figure 69. Mapping of NVxx Workstations to Domain IDs

Here, NV00 maps to the NVTOR NetView domain ID, and the NV01 workstation maps to the XBAOF NetView domain.

Under normal circumstances, each NetView domain ID represents a processor with its MVS operating system and a unique NVxx general automatic reporting workstation. For situations such as testing, backup, or work load management, this relationship needs no maintenance.

In the previous example, both NV00 and NV06 OPC-defined workstations represent their own specific NetView domain, NVTOR and AOFS6, respectively.

Assume that the system represented by NVTOR has failed, and you make the decision to shift the work load to the AOFS6 system. You can accomplish this by changing the domain ID in the first CODE statement from NVTOR to AOFS6. This would imply that the AOFS6 domain is associated with two OPC workstations, namely NV00 and NV06.

If you accomplish this change without altering the OPC definitions, you must reload the SA OS/390 control file. The scheduler or operator needs to ensure that the OPC-defined applications that are running in the failed system are restarted on the backup system. Because the SA OS/390 status records are on the failed system, the scheduler manually recovers the failed environment. Once resynchronization completes, any new scheduled event originally intended for the NetView domain ID NVTOR automatically is scheduled for AOFS6. After you resolve the problem on the NVTOR system, perform the previous scenario in reverse order to restore the system to its original configuration.

When double definitions of this type are used, exercise caution to avoid creating conflicting requests for specific subsystems. For example, if RMF exists in the ADFS6 domain, OPC can then schedule a shutdown request on NV00 and a start request on NV06.

Automated Recovery Functions

Only a small portion of OPC Automation resides in the OPC address space in OPC user exits. These exits communicate to the rest of OPC Automation which resides in the NetView address space. A loss of contact results if a NetView address space becomes unavailable or if OPC Automation code in NetView is unavailable. Also, a communication failure can prevent a request from reaching its ultimate destination.

OPC Automation automated recovery determines which operations are affected by a specific loss of communications. It also determines when the connectivity and availability of a given target NetView is corrected and the NV nn operation is reset to the ready state. This redrives the EQQUX007 exit, allowing it to re-create the original request.

OPC Actions in a Loss of Contact Situation

The EQQUX007 exit or an intermediary NetView with an ended-in-error status and a return code of Sxxx reports a connectivity loss to OPC. OPC does not schedule any dependent operations and shows an error on the operations ended-in-error Status Display Facility panel. OPC takes no further action until the connectivity is restored and OPC Automation automatic recovery is invoked.

The operator or scheduler can manually override the ended-in-error status, thus allowing the application to continue or cancelling it.

OPC Automation Actions in a Loss of Contact Situation

If loss of connectivity to NetView is detected in the OPC Automation portion of the EQQUX007 exit (using the EQQUSINT function directly), then OPC Automation posts an ended-in-error status with a UNTV return code. OPC Automation uses this mechanism because the EQQUX007 exit cannot directly modify operation status.

If the request is received in NetView, but OPC Automation cannot propagate the request to the appropriate target system, OPC Automation uses the OPCAPOST function to post the operation as ended-in-error with an S999 return code.

Appendix. 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 145.

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 147.

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

Syntax

```
DFUPDT type,resource,component,ref_value,info,text
```

Parameters

type

Type used to get Status Display Facility parameters from the control file.

resource

Status Display Facility resource name.

component

Status Display Facility component name.

ref_value

Reference value used for the Status Display Facility entry. Used as a way to group related or duplicate entries. If not supplied, then use *resource*.

info

Information text that appears on the panel for this entry. If not supplied, then use *resource*.

text

Message or user text that appears in the detail panel for this entry. If not specified, then use *resource*.

DFCOPY

Purpose

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

Format

Syntax

```
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.

Example

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

Syntax

```
DFCRIT message text DFCRIT TYPE=t, message
text
```

Parameters

t A 1-character value corresponding to an SDF CRITMSG type entry in the control file. A, E, I, and W are supplied with OPC Automation. Other values may be specified, provided an SDF CRITMSG*t* corresponds to that message.

message text

Message text added to the Status Display Facility critical message display panel.

```
IF MSGID='IOS001I' & TEXT=MESSAGE
  THEN EXEC(CMD('DFCRIT 'MESSAGE) ROUTE(ALL *));
```

Usage Notes

If the **TYPE=** parameter is not specified, *t* is set to the last character of the message ID, and the search is made. If no CRITMSG*t* entry is found, the CRITMSG value will be used.

Example

If you wish to see certain application messages in blue reverse video, add:

```
SDF CRITMSGU,CO=B,PR=500,HL=R
```

to your control file and call DFCRIT from the message table as follows:

```
IF MSGID='NORMAL' & 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

Syntax

```
EVJEAB11 sendernode,component,ref_value,data
```

Parameters

The input parameters for EVJEAB11 are those for SA OS/390 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 Customizing and Programming* 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 of 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 message 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.

Glossary of Terms

The intent of this glossary is to define terms as TME 10 OPC uses them. However, where applicable, terms are taken from the *IBM Dictionary of Computing*, New York; McGraw-Hill, 1994. These terms are marked by an asterisk (*). Unless otherwise noted, the definitions below apply equally well to OPC/ESA and TME 10 OPC.

A

actual duration. At a workstation, the actual time in hours and minutes it takes to process an operation from start to finish.

APAR. Authorized program analysis report. A report of a problem caused by a suspected defect in a current unaltered release of a program.

all workstations closed. A user defined interval during which *all* OPC's workstations are not available for running applications under OPC's control.

Note: All the workstations could be either shut down or simply not available to OPC.

application. (1) A group of related operations performed together to satisfy a specific end user task. (2) A measurable and controllable unit of work that completes a specific user task such as the running of payroll or financial statements. The smallest entity that an application can be broken down into is an operation. Generally, several related operations make up an application.

application description. A database description of an application.

application ID. The name of an application. Examples: Y1976, Payroll.

arrival (A). Status of an operation that indicates it is waiting for the input to arrive before processing.

authority. The ability to access a protected resource.

authority group. A name used to generate a RACF resource name for authority checking.

automatic events. Events recognized by or triggered by an executing program. Automatic events are usually generated by OPC job tracking programs but may also be created by a user-defined program.

automatic reporting workstation. A workstation that reports events (the starting and stopping of operations) in real time to OPC, such as a processor or printer.

automatic job recovery. An OPC function which allows you to specify, in advance, alternative recovery strategies for applications or operations ended in error.

availability. * The degree to which a system (and in OPC, an application) or resource is ready when needed to process data.

B

batch loader. An OPC batch program you can use to create and update information in the application description and operator instruction databases.

bracketed DBCS. A MIXED format field consisting of a DBCS part only, that is, DBCS characters enclosed by a shift-out/shift-in control character pair.

browse. An ISPF/PDF dialog function that manages data for display only. This function lets the user view but not change data.

C

CP. Current plan.

calendar. The data that defines the operation department's processing schedule in days and periods.

capacity. The actual number of parallel servers and workstation resources available during a specified open time interval.

capacity ceiling. The maximum number of operations a workstation can handle simultaneously.

case code. A code in the automatic job recovery function that represents a group of abend codes or return codes. Any code in the JOBCODE and STEPCODE parameters is considered a potential case code if defined as such in the case code macro.

closed workstation. A workstation that is unavailable to process work for a specific time, day, or period.

command. * A request from a terminal for the performance of an operation or the execution of a particular program. A character string from a source external to a system that represents a request for system action.

complete. Status of an operation indicating that it has finished processing.

completion code. An OPC system code indicating how the processing of an operation ended at a workstation.

complex of processors. A JES2 multi-access spool system or a JES3 system with more than one processor.

computer workstation. A workstation that performs MVS processing and usually reports status to OPC automatically. A processor when used as a workstation. It can refer to single processors or multiprocessor complexes serving a single job queue (for example JES2 or JES3 systems).

controller. The portion of TME 10 OPC or OPC/ESA that runs on the controlling processor and contains the tasks that manage OPC databases and plans.

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

current plan. A minute by minute schedule of each operation of an application. It reflects the current state of the operating environment showing the status of work completed and work still to be done.

current schedule. The database that contains the current plan information.

cyclic interval. The number of days in a cyclic period.

cyclic period. A period with a specific origin date and set frequency. A cyclic period can be broken down into two types:

- Those that include work and free days
- Those that include only work days.

Cyclic periods must always represent a fixed time period in days. For example, week (7 days).

D

daily plan. A set of plans that shows work that the operations department does on a particular day or shift. A list by day and application of all operations to be performed within the operations department.

default calendar. (1) A calendar that you have defined for OPC to use when you do not specify a calendar in an application description. (2) A calendar that OPC uses if you have neither specified a calendar in an application description, nor defined your own default calendar.

deadline. See deadline date and deadline time.

deadline date. The latest date by which an occurrence must be complete.

deadline time. The latest time by which an occurrence must be complete.

defined. An open day status which indicates that specific open time intervals exist for a workstation on a particular day.

dependency. A relationship between two operations where the first operation must successfully finish before the second operation can begin.

dialog. The user's online interface with OPC.

displacement. A number specifying 'Number of Days from Period Start' or 'Number of Days from Period End'. Sometimes called offset. See offset.

duration. The time an operation is active at a workstation.

E

edit. An ISPF/PDF dialog function that is used for editing text, collecting data, and modifying data.

end user. A person who uses the services of the data processing center.

ended in error (E). The OPC reporting status for an operation that has ended in error at a workstation.

error code. The system completion code or program return code for automatic reporting workstations. The code entered by the workstation operator for manually reporting workstations.

exclusive. The state of a special resource indicating that it is fully used by one operation and cannot be used simultaneously by other operations.

exclusive resource. A workstation resource that is solely used by one operation and cannot be shared with other operations.

expected arrival time. The time when an operation is expected to arrive at a workstation. It may be calculated by daily planning or specified in the long-term plan.

extend current period. An OPC function that allows the user to extend the current plan up to a maximum of 504 hours (21 days) from the current end date.

external dependency. A relationship between two occurrences where an operation in the first occurrence must successfully finish before an operation in the second occurrence can begin processing. See dependency.

external predecessor. The name given to the operation in the first occurrence of an external dependency that must finish before its external successor can begin processing.

external successor. The name given to the operation, in the second occurrence of an external dependency, that cannot begin until its external predecessor completes.

F

free day. A nonworking day.

free day rule. A rule that determines how OPC will treat free days when the application run day falls on a free day. The rule is as follows:

Excluded: Free days excluded; only work days are taken into account.

Included: Free days included; all days are taken into account, as follows:

- (1) Run before the free day.
- (2) Run after the free day.
- (3) Run on the free day.
- (4) Do not run on the free day.

G

general workstation. A workstation where activities, usually manual, and other than printing and processing, are carried out. Manual activities might be data entry or job setup. A general workstation reporting to OPC is usually manual, but can be automatic.

generic search argument. A portion of a key containing a generic search character which in OPC is an asterisk (*) or percent sign (%). The asterisk represents any string of characters and the percent sign any single character. Use with any portion of a key to search the database for items to be displayed as part of a listing. Examples: %ABC, A*C, A*.

H

host processor. * A processor that controls all or part of a user application network. * In a network, the processing unit in which the access method for the network resides.

highest return code. A numeric value from 0 to 4095. If this return code is exceeded during a job's processing, the job will be reported as ended in error.

I

incident log. An optional function available under the job completion checker.

input arrival. The user-defined date and time an operation or an application becomes ready for processing.

internal dependency. A relationship between two operations within an occurrence where the first operation must successfully finish before the second operation can begin.

internal predecessor. The name given to the operation of an internal dependency that must finish before its internal successor can begin processing.

internal successor. The name given to the operation of an internal dependency that cannot begin until its internal predecessor completes processing.

ISPF. Interactive System Productivity Facility.

interrupted (I). An OPC reporting status for an operation indicating that the operation has been interrupted while processing.

J

job. * A set of data that completely defines a unit of work for a computer. A job usually includes all necessary computer programs, linkages, files, and instructions to the operating system. In OPC, an operation performed at a CPU workstation.

job completion checker (JCC). An optional function of OPC that provides an extended checking capability of the results from CPU operations.

job control language (JCL). * A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

JES. Job Entry Subsystem.

job entry subsystem (JES). * A system facility for spooling, job queuing, and managing I/O.

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

job submission. An OPC process that presents jobs to MVS for running on an OPC defined workstation at a time specified in the daily plan.

JS. The JCL repository data set.

K

keyword. * A symbol that identifies a parameter. * A part of a command operand that consists of a specific character string (such as DSNAME=).

keyword parameter. * A parameter that consists of a keyword, followed by one or more values.

L

LTP. Long-term plan.

last operation. (1) An operation in an occurrence that has no internal successor. (2) The terminating node in a network.

latest start. The latest start day and time (calculated by OPC) for an operation that will allow all occurrences to meet their deadline.

layout ID. A unique name that identifies a specific ready list layout.

limit for feedback. See feedback limit.

local. * Synonym for channel-attached.

local processor. * In a complex of processors under JES3, a processor that executes users' jobs and that can assume global functions in the event of failure of the global processor. In OPC, a processor in the same installation that communicates with the controlling OPC processor through shared DASD communication.

long-term plan. A high-level schedule of processing activities for the forthcoming weeks and months. The scope of a long-term plan can be from one day to four years.

The long-term planning function produces a list of application occurrences identified by name, date, and run time for a specified planning period.

M

manual reporting workstation. A type of workstation reporting where events, once they have taken place, are manually reported to OPC. This type of reporting requires that some action be taken by a workstation operator. Manual reporting is usually performed from a list of ready operations.

mass updating. A function of the application description dialog where a large update to the application database can be requested.

modify current plan. An OPC dialog function used to dynamically change the contents of the current schedule to respond to changes in the operation environment. Examples of special events that would cause alteration of the current schedule are: a rerun, a deadline change, or the arrival of an unplanned application.

most critical application occurrences. Those unfinished applications that have a latest start time that is less than or equal to the current time.

N

node. * In a network, a point where one or more functional units interconnect transmission lines.

noncyclic period. A period that has a varying frequency for which you must define each origin date. Examples: month, payroll period, and quarterly.

nonreporting. A reporting attribute of a workstation which indicates that information is not fed back to OPC.

O

OPC/ESA. Operations Planning and Control/Enterprise Systems Architecture

occurrence. Each instance of an application in the long-term plan and current plan is called an occurrence.

An application occurrence is one attempt to process that application. Occurrences are distinguished from one another by run date, input arrival time, and application ID. For example, one application that runs four times a day is said to have four occurrences a day.

offset. A maximum of 12 positive and 12 negative values in the ranges 1 to 999 and -1 to -999 that indicate on which days of a calendar period an application shall run. See displacement.

OPC host. The processor where OPC updates the current plan database.

OPC local processor. A processor that connects to the OPC host or remote processor through shared event data sets.

open time interval. The time interval during which a workstation is active and can process work.

operation. An operation is a unit of work that is part of an occurrence and is processed at a workstation.

operation waiting for arrival. The status of an operation that indicates that the necessary input has not arrived at a workstation so that the operation can begin processing. This status is applicable only for operations without predecessors.

operation status. The status of an operation at a workstation.

An operation's status can be one of the following:

A Waiting for input to arrive.

R Ready for processing. All predecessors are complete.

***** Ready for processing. There is a nonreporting predecessor. All predecessors are complete but

one or more predecessors were executed at a nonreporting workstation.

- S** Started.
- I** Interrupted operation.
- C** Complete.
- E** Operation ended in error.
- W** Waiting for predecessor to complete.
- U** Undecided. The status is not known.

operator. * (ISO) A symbol that represents the action to be performed in a mathematical operation. * In the description of a process, that which indicates the action to be performed on operands. * A person who operates a machine.

option. A selection item on a menu panel in the OPC dialog.

origin date. The date on which a period (cyclic or noncyclic) starts.

P

panel. * A particular arrangement of presentation windows used to show information to the user. OPC uses only fixed-format panels.

parallel operations. Operations at workstations that are not dependent on one another and therefore can be performed simultaneously.

parallel server. The function that processes operations at a workstation, especially when there is more than one such function. See server.

parameter. * (ISO) A variable that is given a constant value for a specified application and that may denote the application. * A name in a procedure that is used to refer to an argument passed to that procedure.

pending application description. An application description which is incomplete and not ready for use in planning or scheduling.

period. A business processing cycle. A time period defined in the OPC calendar. They are used to describe when, and how often, applications are to run.

period name. A name of a period. Examples are week, month, quarter and fiscal period end.

period type. Periods are of two types: cyclic or noncyclic.

PDE. program development facility.

predecessor. An operation of an internal or external dependency that must finish successfully before its successor operation can begin.

printout routing. The ddname of the daily planning printout data set.

print workstation. A workstation that prints output and usually reports status to OPC automatically.

priority. A digit from 1 to 9 (where 1 = low, 8 = high, and 9 = urgent) that determines how OPC schedules applications to run. A number from 1 (low priority) to 9 (high priority) which establishes the importance of an application relative to other applications.

processor. * (ISO) In a computer, a functional unit that interprets and executes instructions. * A functional unit or part of another unit (such as a terminal or a processing unit) that interprets and executes instructions.

program interface. An OPC interface that allows a user-written program to issue various types of requests to the OPC subsystem.

Q

QCP. Query current plan.

R

RACF. Resource Access Control Facility.

read authority. A type of access authority that allows a user to read the contents of a data set, file, or storage area, but not to change it.

ready (R). The status of an operation indicating that predecessor operations are complete and that the operation is ready for processing.

ready list. A display list of all the operations ready to be processed at a workstation. Ready lists are the means by which workstation operators manually report on the progress of work.

recovery. See automatic job recovery.

remote processor. A processor connected to the OPC host processor by a VTAM network.

remote job tracking. The function of tracking jobs on remote processors connected by VTAM links to an OPC controlling processor. This function enables a central site to control the submitting, scheduling, and tracking of jobs at remote sites.

replan current period. An OPC function that recalculates planned start times for all occurrences to reflect the actual situation.

reporting attribute. A code that specifies how a workstation will report events to OPC.

rerun. An OPC function where an application or part of an application that ended in error can be run again.

rescale factor. A value from 0 to 100 used to reduce the new duration value by a given percentage amount.

return code. An error code issued by OPC for automatic reporting workstations.

row command. A dialog command used to manipulate data in a table.

run cycle period. A time frame defining the effective period and run days of a calendar period.

run day. The date on which an application is to run. It is expressed as a number relative to the start or the end of a run cycle period.

S

SAF. System Authorization Facility.

search argument. A value that is used to search the database for an item that is to be part of a displayed listing.

selection criteria. Search arguments entered on a list criteria panel in the dialog that limit the contents of a listing.

server. A program or device set up for a workstation to perform a service for that particular type of workstation. For example, an initiator is a server for a computer workstation. A printer is a server for a print workstation.

service functions. Functions of OPC that let the user deal with exceptional conditions such as investigating problems, preparing APAR tapes, and testing OPC during implementation.

shared DASD. Direct access storage device that can be accessed from more than one processor.

shared resource. A special or workstation resource that can be used simultaneously by more than one operation while the operation is processed at a workstation.

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 workstation.

status. The current state of an operation or an occurrence.

started (S). An OPC reporting status of an operation or an application indicating that an operation or an occurrence is started.

submit/release data set. A data set shared between the OPC host and a local OPC processor that is used to send job stream data and job release commands from the host to the local processor.

subresources. A set of resource names and rules for the construction of resource names. OPC uses these names when checking a user's authority to access individual OPC records.

subsystem. * A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system.

successor. An operation in an internal or external dependency that cannot begin until its predecessor completes processing.

sysout class. * An indicator used in data definition statements to signify that a data set is to be written on a system output unit. It applies only to print workstations.

T

temporary operator instructions. Operator instructions that have a specific time limit during which they are valid. They will be displayed to the workstation operator only during that time period.

TME 10 OPC. TME 10 Operations Planning and Control

tracker. The portion of TME 10 OPC or OPC/ESA that runs on every system in your complex. It acts as the communication link between the MVS system that it runs on and the controller.

tracking event log. A log of job tracking events and updates to the current schedule.

transport time. The time allotted for transporting materials from the workstation where the preceding operation took place, to the workstation where the current operation is to occur.

TSO. Time Sharing Option.

time zone support. A feature of OPC that allows applications to be planned and run with respect to the

local time of the processor that runs the application. Some networks may have processors in different time zones. The controlling processor will make allowance for differences in time during planning activities, for example the input arrival time of predecessor applications, to make sure that interacting activities are correctly coordinated.

turnover. A subfunction of job tracking that is activated when job tracking creates an updated version of the current schedule.

U

undecided (U). An OPC reporting status for an operation or an application indicating that the status is not known.

update authority. Access authority given to a user by RACF to use the ISPF/PDF edit functions of the OPC dialog. Access authority to modify a master file or data set with the current information.

V

validity period. The time interval defined by an origin date and an end date within which a run cycle or an application description is valid.

versions. Applications with the same ID but different validity dates.

VSAM. Virtual Sequential Access Method.

VTAM. Virtual Telecommunication Access Method.

W

waiting (W). An OPC reporting status (for an application) indicating that it is waiting for a predecessor operation to complete.

waiting list. A list of submitted jobs that are waiting to be processed.

work day end time. The time at which OPC will consider a work day to have ended when that work day immediately precedes a free day. For example, if you specify Saturday to be a free day, you could specify 08.00 hours Saturday morning as the end of Friday's work day. OPC can then plan work to be done from 00.00 to 08.00 Saturday morning, as if that time was actually part of Friday.

workstation. A unit, place, or group that performs a specific data processing function. A logical place where work occurs in an operations department.

OPC requires that you define the following characteristics for each workstation: the type of work it does, the quantity of work it can handle at any

particular time, and the times it is active. The activity that occurs at each workstation is called an operation.

workstation description database. An OPC database containing descriptions of the workstations in the operations department.

workstation resources. Limited resources defined for each workstation that an operation requires a certain amount of to process work.

workstation type. Each workstation can be one of three types: computer, print, or general.

work day. A day on which applications can normally be scheduled to start.

Index

Special Characters

&EHKVAR1 59, 62, 103
&EHKVAR2 59
&EHKVAR7 100
&EHKVAR8 100
&EHKVAR9 94, 100
\$HASP373 144

A

ACF 4
actual state 4
alerts 13
application groups 8
 BASIC 9
 MOVE 9
 nesting of 9
 SERVER 9
applications 3
 OPC-defined 50, 53
 policy items
 MESSAGES/USER DATA 10, 57, 65
 recovery 20
automation
 control file 4
 goal-driven 4
 operators 31

B

backup 138

C

CNMCNETV 25
commands
 EVJESHUT 82
 EVJESPIN 136
 OPCACAL 83
 OPCACMD 84, 129
 OPCACOMP 15, 86, 100
 OPCALIST 87
 OPCAMOD 89
 OPCAPOST 15, 92, 134
 OPCAQRY 131
 OPCSRST 93
 SRSTAT 135
completion flag 36, 37, 99
connectivity loss 141
customization dialogs 3

D

daily plan
 extension 56
data areas 93
data distribution
 across multiple systems 19
dependencies
 start 4

dependencies (*continued*)
 stop 4
desired state 4
DFBTCH 144
DFCOPY 146
DFCRIT 143, 147
DFDEL 143
DFTAPK 143
DFTAPM 143
DFTAPO 143
DFTSOR 144
DFTSOU 144
DFUPDT 143, 145

E

entry 3
entry type 3
EQQDUMP 45
EQQMLIB 44
EQQMLOG 45
EQQPARM 44
EQQUX007 exit 24, 46
error codes 31, 86
 Sxxx 15, 55
 S998 27, 38
 S999 27, 33, 38
 Uxxx 15, 55
 U003 27
 UNTV 25, 33, 35, 38
events 7
EVJEAB11 148
EVJECCAL 83
EVJERCAL 83
EVJESHUT 82
EVJESPIN 136
EVJESPIN (initialization module) 33, 36, 47
EVJESPRQ (request module) 15, 27, 35
EVJESPPSC (status change module) 29, 36, 99
EVJESPTTE (timer module) 29, 36, 99
EVJESPPVY (verify module) 14, 26
EVJTOPPI 25

F

flags
 completion 36, 37, 99
 timer 36, 37, 99

G

generic routines
 ISSUECMD 10
goal 4
goal-driven automation 4

I

IEC501A 143
IEC502E 143

IEC701D 143
IEC705I 143
IEF125I 144
IEF126I 144
IEF233A 143
IEF234E 143
IEF251I 143
IEF404I 144
IEF450I 144
IEF453I 144
initialization
 EVJESPIN 33
 OPC 23
 OPC Automation 23
 OPC components 31
 OPC-controlled subsystems 32
 SA OS/390 31
initialization module (EVJESPIN) 36, 47
installation 43
interception
 OPC alerts 13
ISSUECMD 10

L

log entries 16
long term outage 139
loss of contract 138
LPAR (logically partitioned mode)
 preparing 19

M

MAT 9
MESSAGES/USER DATA keywords
 format descriptions 65
 notational conventions 67
 translation rule 66
OPCA 58, 60, 71
OPCACMD 15, 57, 74, 102
OPCAPARM 78, 100
MESSAGES/USER DATA policy item 65
 attributes 65
 CMD 66
 CODE 68
MPFLISTxx 43

N

NetView domain
 represented by OPC workstation 49
NetView message automation table 9, 45
NNT link 38
notational conventions
 for CMD, REP, CODE attributes 67
 for USER E-T PAIRS 70

O

online databases
 cycling individually 18

- online services
 - hours of availability 17
- OPC
 - API (application program interface) 37
 - application 53
 - Exit 7 46
 - initialization with OPC
 - Automation 12
 - operation 53
- OPC Automation
 - backup 138
 - connectivity loss 141
 - long term outage 139
 - loss of contact 138
 - OPCAPOST 30
 - outage 139, 141
 - request module (EVJESPRQ) 15, 27
 - status change module (EVJESPSC) 29
 - timer module (EVJESPTTE) 29
 - verify module (EVJESPVY) 14, 26
- OPC Automation status file
 - initialization 47
 - subsystem records 32, 37, 99
- OPC controller
 - definition in SA OS/390 46
- OPC data server
 - definition in SA OS/390 46
- OPC data sets 44
- OPC-defined applications 50
- OPC initialization
 - EQQPARM 44
- OPC messages
 - SEQQMSG0 44
- OPC monitor panel 120
- OPC operation 12, 53
 - Job name** field 57
 - Operation text** field 57
 - states 55
- OPC request 14, 51, 54
 - buffer 16, 95
 - default START/STOP 61
 - displaying 52
 - naming conventions 61, 62, 101
 - non-subsystem 62, 101
 - parameters 59
 - subsystem-related 98
 - requiring user programming 62, 99
 - using standard functions 62, 99
 - types 61
- OPC SYSTEMS DETAILS 82
- OPC tracker
 - definition in SA OS/390 46
- OPC workstation 12, 24
 - representing NetView domain 49
 - naming conventions 49
 - time dependency 54
- OPCA 71
- OPCACAL 83
- OPCACMD (command) 37, 84, 129
- OPCACMD (MESSAGES/USER DATA keyword) 57, 74
- OPCACOMP 15, 86, 94, 100
- OPCALIST 87
- OPCAMOD 89
- OPCAPARM 78, 100

- OPCAPOST 15, 27, 29, 30, 36, 92, 134
- OPCAQRY 131
- OPCSRST 93
- operation (*see* OPC operation) 12
- outage 139, 141

P

- panels
 - Applications in Error 110, 121
 - Batch Job Display 128
 - Critical Messages 122
 - OPC Monitor panel 110
 - SA OS/390 – Support Systems 119
 - SA/OPC – Automation Operation Status Display 131
 - SA/OPC – Display or Modify OPC Data 111, 129
 - SA/OPC – Main Menu 118
 - SA/OPC – OPC Occurrence Data 114, 130
 - SA/OPC – Operation Status Display 133
 - Tape Unit Display 124
 - TSO Users Status 126
- persistence of requests 6
- policy database 3
- policy item 3
- policy object 3
- policy objects
 - AUTO OPERATORS 45
 - CONTROLLER DETAILS 46
 - OPC SPECIAL RESOURCES 46
 - OPC SYSTEM DETAILS 46
 - OPC SYSTEMS DETAILS 82
 - USER E-T PAIRS 69, 74, 76
 - WORKSTATION DOMAINS 46, 50
- PPI (*see* program-to-program interface) 13
- priority of requests 4
- problem determination 48
- problem resolution 110
- program-to-program interface 14, 25
 - dispatcher 25
 - EVJTOPPI 25

R

- recovery 20, 137
 - automated 38
- REQCOMP 94
- request module (EVJESPRQ) 15, 35
- requestor ID block 94
- requests (SA OS/390) 4
 - conflicting 6
 - persistence 6
 - priority 4, 7, 8
 - propagation 4
- resources 3
 - names, format of 3
- resynchronization 137

S

- SDF (*see* Status Display Facility) 109
- service periods 8
- service windows 8

- service windows 8
- special resources 16
- SRSTAT 135
- start dependencies 4
- state
 - actual 4
 - desired 4
- status
 - of OPC operation 24, 55
 - of SA OS/390 subsystem 62
- status change module (EVJESPSC) 36, 99
- Status Display Facility 109
 - terminal requirements 109
- Status Display Facility enhancements
 - \$HASP373 144
 - abend 144
 - capture
 - batch job start and stop 144
 - TSO logons and logoffs 144
 - coding reference 143
 - critical message processor 143
 - DFBTCH 144
 - DFCOPY 146
 - DFCRIT 143, 147
 - DFDELT 143
 - DFTAPK 143
 - DFTAPM 143
 - DFTAPO 143
 - DFTSOR 144
 - DFTSOU 144
 - DFUPDT 143, 145
 - EVJEAB11 148
 - IEC501A 143
 - IEC502E 143
 - IEC701D 143
 - IEC705I 143
 - IEF125I 144
 - IEF126I 144
 - IEF233A 143
 - IEF234E 143
 - IEF251I 143
 - IEF404I 144
 - IEF450I 144
 - IEF453I 144
 - insert display data 145
 - job
 - ended 144
 - failed 144
 - started 144
 - process critical messages 147
 - Status Display Facility delete 143
 - synchronize data in a distributed environment 148
 - synchronize SDF components 146
 - tape
 - check 143
 - message processor 143
 - online 143
 - TMS001 143
 - TMS002 143
 - TSO refresh 144
 - TSO user
 - abend 144
 - logged off 144
 - logged on 144
 - update CLIST 143

- status file (*see* OPC Automation status file) 32
- stop dependencies 4
- subsystems 3
- systems
 - policy items
 - CONTROLLER DETAILS 46
 - OPC SYSTEM DETAILS 46
 - WORKSTATION DOMAINS 46

T

- time dependencies 54
- timer flag 36, 37, 99
- timer module
 - standard module (EVJESPTE) 29, 99
 - user-defined 62
 - format of call 99
- timer module (EVJESPTE) 36
- TMS001 143
- TMS002 143
- triggers 7
 - events 7
 - shutdown conditions 7
 - startup conditions 7

U

- USER E-T PAIRS 69, 74, 76
 - format descriptions
 - notational conventions 70

V

- variables
 - &EHKVAR1 59, 62, 103
 - &EHKVAR2 59
 - &EHKVAR7 100
 - &EHKVAR8 100
 - &EHKVAR9 100
- verify module (EVJESPVY) 14, 26
- votes 5

W

- workstation (*see* OPC workstation) 12
- WORKSTATION DOMAINS 50

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

System Automation for OS/390
OPC Automation
Programmer's Reference
and Operator's Guide
Version 2 Release 1

Publication No. SC33-7046-00

Overall, how satisfied are you with the information in this book?

| | Very Satisfied | Satisfied | Neutral | Dissatisfied | Very Dissatisfied |
|----------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| Overall satisfaction | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

How satisfied are you that the information in this book is:

| | Very Satisfied | Satisfied | Neutral | Dissatisfied | Very Dissatisfied |
|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| Accurate | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Complete | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to find | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to understand | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Well organized | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Applicable to your tasks | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

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.



Fold and Tape

Please do not staple

Fold and Tape



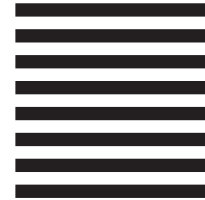
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Entwicklung GmbH
Department 3248
Schönaicher Strasse 220
D-71032 Böblingen
Federal Republic of Germany



Fold and Tape

Please do not staple

Fold and Tape



Program Number: 5685-151



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

SC33-7046-00

