

# **MAINVIEW<sup>®</sup>** **Total Object Manager** **User Guide**

**Version 1.1**

**November 21, 2003**



Copyright 2003 BMC Software, Inc. All rights reserved.

BMC Software, the BMC Software logos, and all other BMC Software product or service names are registered trademarks or trademarks of BMC Software, Inc. IBM and DB2 are registered trademarks of International Business Machines Corp. All other trademarks belong to their respective companies.

BMC Software considers information included in this documentation to be proprietary and confidential. Your use of this information is subject to the terms and conditions of the applicable End User License Agreement for the product and the proprietary and restricted rights notices included in this documentation.

## Restricted Rights Legend

U.S. Government Restricted Rights to Computer Software. UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure of any data and computer software by the U.S. Government is subject to restrictions, as applicable, set forth in FAR Section 52.227-14, DFARS 252.227-7013, DFARS 252.227-7014, DFARS 252.227-7015, and DFARS 252.227-7025, as amended from time to time. Contractor/Manufacturer is BMC Software, Inc., 2101 CityWest Blvd., Houston, TX 77042-2827, USA. Any contract notices should be sent to this address.

---

## Contacting BMC Software

You can access the BMC Software Web site at <http://www.bmc.com>. From this Web site, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

### United States and Canada

**Address** BMC Software, Inc.  
2101 CityWest Blvd.  
Houston TX 77042-2827

**Telephone** 713 918 8800 or  
800 841 2031

**Fax** 713 918 8000

### Outside United States and Canada

**Telephone** (01) 713 918 8800

**Fax** (01) 713 918 8000

---

---

## Customer Support

You can obtain technical support by using the Support page on the BMC Software Web site or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, please see “Before Contacting BMC Software.”

### Support Web Site

You can obtain technical support from BMC Software 24 hours a day, 7 days a week at [http://www.bmc.com/support\\_home](http://www.bmc.com/support_home). From this Web site, you can

- read overviews about support services and programs that BMC Software offers
- find the most current information about BMC Software products
- search a database for problems similar to yours and possible solutions
- order or download product documentation
- report a problem or ask a question
- subscribe to receive e-mail notices when new product versions are released
- find worldwide BMC Software support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

### Support by Telephone or E-mail

In the United States and Canada, if you need technical support and do not have access to the Web, call 800 537 1813. Outside the United States and Canada, please contact your local support center for assistance. To find telephone and e-mail contact information for the BMC Software support center that services your location, refer to the Contact Customer Support section of the Support page on the BMC Software Web site at [www.bmc.com/support\\_home](http://www.bmc.com/support_home).

### Before Contacting BMC Software

Before you contact BMC Software, have the following information available so that Customer Support can begin working on your problem immediately:

- product information
  - product name
  - product version (release number)
  - license number and password (trial or permanent)
- operating system and environment information
  - machine type
  - operating system type, version, and service pack or other maintenance level such as PUT or PTF
  - system hardware configuration
  - serial numbers
  - related software (database, application, and communication) including type, version, and service pack or maintenance level
- sequence of events leading to the problem
- commands and options that you used
- messages received (and the time and date that you received them)
  - product error messages
  - messages from the operating system, such as `file system full`
  - messages from related software



---

---

# Contents

<b>About This Book</b> .....	<b>xix</b>
<b>Chapter 1</b>	<b>Introduction to the MAINVIEW Total Object Manager (TOM)</b>
Overview .....	1-1
Benefits of TOM .....	1-3
TOM Requirements .....	1-3
TOM Terms and Concepts .....	1-4
Objects .....	1-4
Sets .....	1-5
Object Status and Dependencies .....	1-6
Scheduling and Calendar Dependencies .....	1-6
TOM-Plex .....	1-7
Single Image Mode .....	1-7
How TOM Works .....	1-8
How to Start TOM .....	1-8
How TOM Starts and Stops Objects .....	1-10
How TOM Recovers Failed Objects .....	1-12
How to Specify Start Command Limits .....	1-14
<b>Chapter 2</b>	<b>Implementing and Customizing TOM</b>
Preparing to Start and Use TOM .....	2-2
Starting the TOM Address Space .....	2-2
Running Compiled REXX Procedures under TSO .....	2-3
Start MAINVIEW AutoOPERATOR .....	2-3
Running Multiple TOM Systems within a Sysplex .....	2-4
Sharing Data Sets .....	2-4
Grouping Multiple TOM Systems in a Sysplex .....	2-4
Starting TOM for the First Time .....	2-6
Implementing Security for TOM Systems .....	2-7
Additional Information .....	2-7
Specifying Multiple BBPARM Members for TOM .....	2-8
<b>Chapter 3</b>	<b>Creating and Managing Objects</b>
Overview .....	3-2

Types of Objects . . . . .	3-2
How Objects Are Stored . . . . .	3-3
Defining an Object . . . . .	3-4
Defining a Model. . . . .	3-7
Inheritance . . . . .	3-7
How to Define a Model. . . . .	3-7
Specifying an Object That Inherits All Properties from a Model . . . . .	3-8
Specifying an Object That Inherits Some Properties from a Model . . . . .	3-9
Defining an Object in SUSPEND Mode . . . . .	3-10
Defining a Layer Object . . . . .	3-12
Viewing Set Associations . . . . .	3-13
Defining Valid Systems for the Object to Start On . . . . .	3-16
How TOM Progresses through the Valid Systems List . . . . .	3-16
Defining a Schedule . . . . .	3-19
Defining How to Start an Object. . . . .	3-23
Defining How to Stop the Object . . . . .	3-29
Defining Start or Stop Retry Commands . . . . .	3-32
Associating EXECs with an Object. . . . .	3-36
Defining Events for Object Startup or Shutdown . . . . .	3-43
Defining Events for Abnormal Object Termination . . . . .	3-46
Defining Alternate Systems. . . . .	3-49
Defining System Affinity . . . . .	3-51
High Affinity. . . . .	3-51
Med (Medium) Affinity . . . . .	3-52
Low Affinity . . . . .	3-53
Setting the Affinity . . . . .	3-53
Defining Object Dependencies . . . . .	3-55
Defining Reinstatement of an Object . . . . .	3-58
Verifying the Starting or Stopping of an Object . . . . .	3-61
Using Variables on the Started Task Definition Panel. . . . .	3-63
Where Variables Can Be Entered . . . . .	3-63
How Variables Are Resolved . . . . .	3-63
Example of Resolving Variables from Different Sources. . . . .	3-64
List of Reference Variables. . . . .	3-65

## Chapter 4

### Creating and Managing Sets

Understanding Sets . . . . .	4-1
When to Use Sets . . . . .	4-2
Adding Objects and Sets to a New Set . . . . .	4-3
Managing Sets. . . . .	4-9
Starting a Set. . . . .	4-10
Stopping a Set. . . . .	4-12
Resetting a Set . . . . .	4-14
Suspending or Activating a Set. . . . .	4-16

<b>Chapter 5</b>	<b>Creating Calendar Bases</b>	
	Overview of Calendar Bases . . . . .	5-1
	When to Use a Calendar Base . . . . .	5-2
	Calendar Terms and Concepts . . . . .	5-3
	Creating a New Calendar Base . . . . .	5-6
	Adding a Day Entry . . . . .	5-8
	Adding a Period Entry . . . . .	5-12
	Adding a Time Entry . . . . .	5-15
	Adding Items to a Calendar Base Set . . . . .	5-18
	Entering Calendar Dependencies for Object Definitions . . . . .	5-22
	How the Include logic Field Works . . . . .	5-24
	Calendar Dependency Panel . . . . .	5-25
	Examples of Entering Calendar Dependencies . . . . .	5-26
<b>Chapter 6</b>	<b>Using the Started Task Overview Panel to Manage Objects</b>	
	Understanding the Started Task Overview Panel . . . . .	6-2
	Viewing Object Information . . . . .	6-2
	What the Except Column Shows . . . . .	6-4
	Managing Objects . . . . .	6-5
	Starting an Object . . . . .	6-6
	Stopping an Object . . . . .	6-9
	Resetting an Object . . . . .	6-11
	Suspending or Activating an Object . . . . .	6-13
	Changing the Status of an Object . . . . .	6-16
	Managing Definition Databases . . . . .	6-18
<b>Chapter 7</b>	<b>Administrative Tasks</b>	
	How TOM Communicates with Other TOM Systems in a Sysplex . . . . .	7-2
	Overview of Cloning . . . . .	7-2
	Troubleshooting Cloning Failures . . . . .	7-4
	Backing Up the Registry . . . . .	7-5
	Overview of the Registry . . . . .	7-6
	Handling Registry Errors . . . . .	7-6
	Restoring a Registry from a Backup . . . . .	7-7
	Monitoring Registry Size . . . . .	7-8
	Maintaining Identical Registry Sizes . . . . .	7-9
	TOM System Shutdown Considerations . . . . .	7-10
	Difference between Stopping a TOM System and Stopping its Host System . . . . .	7-11
	Shutting Down a TOM System . . . . .	7-12
	Stopping Objects on a Specific TOM System . . . . .	7-13
	How TOM Stops Objects Defined with a Shutdown Mode of QUICK . . . . .	7-14
	Manually Shutting Down Remaining Objects . . . . .	7-15
	Stopping the TOM Address Space . . . . .	7-17
	Shutting Down a Set . . . . .	7-18
	Stopping Objects within a Set . . . . .	7-19
	Manually Shutting Down Remaining Objects . . . . .	7-21

Creating Multiple Definition Databases . . . . .	7-23
Reviewing Messages in the TOM Log . . . . .	7-26

**Chapter 8**

**Managing Objects across Sysplexes with Layering**

Overview of Layering . . . . .	8-2
Layering Terms and Concepts . . . . .	8-3
Layering Tasks. . . . .	8-5
Getting Started with Layering. . . . .	8-6
Entering Sysplex Names. . . . .	8-6
Entering System Names and Layer IDs for the Sysplex. . . . .	8-10
When to Create Layer Objects . . . . .	8-13
Layer Object Naming Conventions. . . . .	8-13
Layer Set Naming Conventions . . . . .	8-14
Creating a Layer Object . . . . .	8-15
Creating Layer Sets. . . . .	8-22
Adding Layer Objects and Layer Sets to a System Layer . . . . .	8-26
Adding System Layers to a Sysplex Layer. . . . .	8-30
Attaching a System Layer to a System within the Same Sysplex . . . . .	8-33
Attaching a System Layer to a System in a Remote Sysplex . . . . .	8-36
Attaching a Sysplex Layer to a Sysplex . . . . .	8-39
Example Scenarios . . . . .	8-42
How Object Names Are Resolved after an Attach . . . . .	8-42
Example of Creating a Layer Object for Multiple Systems . . . . .	8-44
Updating Objects and Sets Created from Layering. . . . .	8-47
Updating a Layer Object. . . . .	8-47
Updating a Layer Set . . . . .	8-47
Copying the Registry . . . . .	8-48
Using the IDCAMS REPRO Utility . . . . .	8-49

**Chapter 9**

**Using AOAnywhere TOM Functions and Console Commands**

AOAnywhere TOM Functions . . . . .	9-2
Coding Conventions . . . . .	9-2
Creating EXECs That Can Run on Multiple TOM Systems . . . . .	9-2
How to Use the TOMID and XCFGROUP Parameters . . . . .	9-3
Object and Set Naming Restrictions. . . . .	9-3
How to Use Stem Variables . . . . .	9-4
Return Codes. . . . .	9-4
Command Syntax for AOEXEC TOM Functions . . . . .	9-5
AOEXEC TOM Function List. . . . .	9-5
ACTIVATE Function . . . . .	9-6
BLOCK Function . . . . .	9-8
BOUNCE Function. . . . .	9-10
CHSTATUS Function . . . . .	9-12
GET Function . . . . .	9-14
LIST Function. . . . .	9-18
LOCK Function . . . . .	9-20
MOVE Function . . . . .	9-22
RESET Function . . . . .	9-25

---

SETPROPERTY Function . . . . .	9-27
SHUTSET Function . . . . .	9-29
SHUTSYS Function . . . . .	9-30
START Function . . . . .	9-31
STOP Function . . . . .	9-34
SUSPEND Function . . . . .	9-37
SYSINFO Function . . . . .	9-39
UNBLOCK Function . . . . .	9-41
UNLOCK Function . . . . .	9-43
TOM Console Commands . . . . .	9-45
Coding Conventions . . . . .	9-45
List of TOM Console Commands . . . . .	9-46

**Chapter 10**

**Converting the Continuous State Manager Version 6 Repository**

Migration Considerations . . . . .	10-2
Disabling CSM . . . . .	10-2
Using Test Mode . . . . .	10-2
Defining a Valid System, Definition Base Name, and Initial Control Setting . . . . .	10-3
Selecting a Name Pattern . . . . .	10-3
Definitions That Are Not Converted . . . . .	10-4
Accessing the Conversion Utility . . . . .	10-5
Exception Messages . . . . .	10-11
Additional Exceptions . . . . .	10-13

**Appendix A**

**TOM Statuses**

**Index**



---

# Figures

Figure 2-1	Accessing the AutoOPERATOR Primary Option Menu . . . . .	2-6
Figure 2-2	Selecting TOM from the Automation Menu . . . . .	2-7
Figure 3-1	TOM Primary Options Menu . . . . .	3-4
Figure 3-2	Defining an Object: Name, Description, Product, Last modified, STC name, Step name, and Type . . . . .	3-5
Figure 3-3	This is a Model only and Modeled from Fields . . . . .	3-8
Figure 3-4	Control Field . . . . .	3-10
Figure 3-5	Pattern Field . . . . .	3-12
Figure 3-6	Set associations Field . . . . .	3-13
Figure 3-7	First Started tasks sets Pop-Up Panel . . . . .	3-14
Figure 3-8	Second Started tasks sets Pop-Up Panel . . . . .	3-14
Figure 3-9	Members of Set Pop-Up Panel . . . . .	3-15
Figure 3-10	Valid Systems Field . . . . .	3-17
Figure 3-11	Valid Systems Pop-Up Panel . . . . .	3-17
Figure 3-12	Schedules Field . . . . .	3-19
Figure 3-13	Schedule Calendar Dependencies Pop-Up Panel . . . . .	3-20
Figure 3-14	Calendar Dependency Pop-Up Panel . . . . .	3-20
Figure 3-15	Fields for Defining Object Startup . . . . .	3-24
Figure 3-16	Fields for Defining How to Stop an Object . . . . .	3-29
Figure 3-17	Start retry commands and Stop retry commands Fields . . . . .	3-32
Figure 3-18	Start retry commands Pop-Up Panel—First Panel . . . . .	3-33
Figure 3-19	Start retry commands Pop-Up Panel—Second Panel . . . . .	3-33
Figure 3-20	Pre-start EXECs, Post-start EXECs, Pre-stop EXECs, and Post-stop EXECs Fields . . . . .	3-36
Figure 3-21	Pre-start EXECs Pop-Up Panel—First Panel . . . . .	3-37
Figure 3-22	Pre-start EXECs Pop-Up Panel—Second Panel . . . . .	3-37
Figure 3-23	Startup validation and Shutdown validation Fields . . . . .	3-43
Figure 3-24	Startup validation Pop-Up Panel—First Panel . . . . .	3-43
Figure 3-25	Startup validation Pop-Up Panel—Second Panel . . . . .	3-44
Figure 3-26	Pop-Up Panel for Startup Message Event Type . . . . .	3-44
Figure 3-27	Updated Startup validation Pop-Up Panel . . . . .	3-45
Figure 3-28	Abnormal termination validation Field . . . . .	3-46
Figure 3-29	Abnormal Termination Validation Pop-Up Panel . . . . .	3-46
Figure 3-30	ABEND validation Pop-Up Panel . . . . .	3-47

Figure 3-31	Pop-Up Panel for Message Event Type	3-47
Figure 3-32	Updated Abnormal Termination Validation Pop-Up Panel	3-48
Figure 3-33	Try alternate system upon failure Field	3-49
Figure 3-34	Affinity Field	3-53
Figure 3-35	Dependency Property value Field	3-55
Figure 3-36	Defining an Object: Dependency Pop-Up Panel	3-55
Figure 3-37	Reinstatement, Count, and Require confirmation Fields	3-59
Figure 3-38	Verify Start and Verify Stop Fields	3-61
Figure 4-1	TOM Primary Options Menu	4-3
Figure 4-2	Started Task Sets Panel	4-3
Figure 4-3	New Set Name Pop-Up Panel	4-4
Figure 4-4	Members of Set Pop-Up Pane	4-4
Figure 4-5	Objects Panel	4-5
Figure 4-6	Adding an Object to a Set	4-5
Figure 4-7	Defining a Set: Members of Set Pop-Up Panel with Object Selected	4-6
Figure 4-8	Defining a Set: Members of Set Pop-Up Panel with Object Selected	4-6
Figure 4-9	Sets Panel	4-7
Figure 4-10	Adding the Set to a Set	4-7
Figure 4-11	Defining a Set: Members of Set Pop-Up Panel (2)	4-8
Figure 4-12	Defining a Set: Members of Set Pop-Up Panel with Object Selected	4-8
Figure 4-13	Started Task Sets Panel	4-9
Figure 4-14	Starting Sets	4-10
Figure 4-15	Confirm Start Pop-Up Panel	4-10
Figure 4-16	Started Task Sets: Stopping Sets	4-12
Figure 4-17	Confirm Stop Pop-Up Panel	4-12
Figure 4-18	Resetting Sets	4-14
Figure 4-19	Confirm Reset Pop-Up Panel	4-14
Figure 4-20	Suspending Sets	4-16
Figure 4-21	Confirm Suspend Pop-Up Panel	4-16
Figure 4-22	Activating Sets	4-17
Figure 4-23	Confirm Activate Pop-Up Panel	4-17
Figure 5-1	TOM Primary Options Menu	5-6
Figure 5-2	Calendar Bases Panel	5-6
Figure 5-3	New Calendar Base Pop-Up Panel	5-7
Figure 5-4	Calendar Bases Panel: New Base Added	5-7
Figure 5-5	Adding Days to a Calendar Base	5-8
Figure 5-6	Base Calendar Entries Pop-Up Panel	5-9
Figure 5-7	Calendar Day Pop-Up Panel	5-9
Figure 5-8	Calendar Day Pop-Up Panel: Example	5-10
Figure 5-9	Base Calendar Entries Pop-Up Panel: New Day Entry Added	5-10
Figure 5-10	Adding Periods to a Calendar Base	5-12
Figure 5-11	Base Calendar Entries Pop-Up Panel	5-12
Figure 5-12	Calendar Period Pop-Up Panel	5-13
Figure 5-13	Calendar Period Entry Pop-Up Panel: Example	5-13
Figure 5-14	Base Calendar Entries Pop-Up Panel: New Period Entry Added	5-14

Figure 5-15	Adding a Time Entry to a Calendar Base	5-15
Figure 5-16	Base Calendar Entries Pop-Up Panel	5-15
Figure 5-17	Calendar Time Pop-Up Panel	5-16
Figure 5-18	Calendar Time Pop-Up Panel: Example	5-16
Figure 5-19	Base Calendar Entries Pop-Up Panel: New Time Entry Added	5-17
Figure 5-20	Adding a Set Entry to a Calendar Base	5-18
Figure 5-21	Base Calendar Entries Pop-Up Panel	5-18
Figure 5-22	Calendar Set Pop-Up Panel	5-19
Figure 5-23	Calendar Set Pop-Up Panel: Example	5-19
Figure 5-24	Base Calendar Entries Pop-Up Panel: New Set Entry Added	5-19
Figure 5-25	Base Calendar Entries Pop-Up Panel: Edit New Set Entry	5-20
Figure 5-26	Base Calendar Entries Pop-Up Panel: Edit New Set Entry (2)	5-20
Figure 5-27	Base Calendar Entries Pop-Up Panel: Edit New Set Entry (3)	5-21
Figure 5-28	SCHEDULE Calendar Dependencies Panel	5-23
Figure 5-29	Calendar Dependency Panel	5-23
Figure 5-30	Calendar Dependency Panel	5-25
Figure 5-31	Entering an Exclude Calendar Dependency	5-27
Figure 5-32	Entering an Include Calendar Dependency	5-28
Figure 5-33	Entering an Include Calendar Dependency: Using OR Logic	5-29
Figure 5-34	Entering an Include Calendar Dependency: Using OR Logic (2)	5-30
Figure 5-35	SCHEDULE Calendar Dependencies Panel with OR Logic Specified	5-31
Figure 5-36	Entering an Include Calendar Dependency: Using AND Logic	5-32
Figure 5-37	Entering an Include Calendar Dependency: Using AND Logic (2)	5-33
Figure 5-38	SCHEDULE Calendar Dependencies Panel with AND Logic Specified	5-34
Figure 5-39	Entering an Exclude Calendar Dependency: Stopping an Object	5-35
Figure 5-40	Entering an Exclude Calendar Dependency for a Period of Time	5-36
Figure 5-41	Entering an Exclude Calendar Dependency for a Specific Time	5-37
Figure 5-42	Schedule Calendar Dependencies	5-38
Figure 6-1	Displaying the Started Task Overview Panel	6-3
Figure 6-2	Started Task Overview Panel	6-3
Figure 6-3	Started Task Overview: Line Commands	6-5
Figure 6-4	Starting an Object	6-6
Figure 6-5	Confirm Start Pop-Up Panel	6-6
Figure 6-6	Stopping an Object	6-9
Figure 6-7	Confirm Stop Pop-Up Panel	6-9
Figure 6-8	Resetting an Object	6-11
Figure 6-9	Confirm Reset Pop-Up Panel	6-11
Figure 6-10	Started Task Overview: Suspending an Object	6-14
Figure 6-11	Confirm Suspend Pop-Up Panel	6-14
Figure 6-12	Activating an Object	6-15
Figure 6-13	Confirm Activate Pop-Up Panel	6-15
Figure 6-14	Changing the Status of an Object	6-17
Figure 6-15	Confirm Change Status Pop-Up Panel	6-17
Figure 6-16	Definition Base Field	6-18

Figure 6-17	Definition databases Panel . . . . .	6-19
Figure 6-18	Changing Definition Databases Panel: Activate . . . . .	6-20
Figure 6-19	Changing Definition Databases Panel: Switch To . . . . .	6-21
Figure 7-1	TOM Administration Options Menu: Shutting Down a TOM System . . . . .	7-13
Figure 7-2	TOM Managed Systems Panel: Shut Down a TOM System . . . . .	7-13
Figure 7-3	Please confirm shutsys Pop-Up Panel . . . . .	7-14
Figure 7-4	Verifying That Objects Are Stopped . . . . .	7-16
Figure 7-5	Started Task Sets Panel . . . . .	7-19
Figure 7-6	Confirm Shutset Pop-up Panel . . . . .	7-19
Figure 7-7	Verifying That Objects Are Stopped . . . . .	7-22
Figure 7-8	TOM Primary Options Menu . . . . .	7-23
Figure 7-9	TOM Administration Options Menu . . . . .	7-24
Figure 7-10	Definition Databases Panel . . . . .	7-24
Figure 7-11	New database name Pop-Up Panel . . . . .	7-24
Figure 7-12	Definition Databases Panel: New Database Added . . . . .	7-25
Figure 7-13	TOM Primary Options Menu . . . . .	7-26
Figure 7-14	TOM Log . . . . .	7-27
Figure 8-1	TOM Primary Options Menu . . . . .	8-7
Figure 8-2	Administration Options Menu . . . . .	8-7
Figure 8-3	Layer Management Panel . . . . .	8-8
Figure 8-4	Sysplex Matrix Panel . . . . .	8-8
Figure 8-5	Add Sysplex Definition Pop-up Panel . . . . .	8-9
Figure 8-6	Sysplex Matrix Panel (New Sysplex Added) . . . . .	8-9
Figure 8-7	Entering System Names . . . . .	8-11
Figure 8-8	System Definitions Panel . . . . .	8-11
Figure 8-9	Add Systems Definition Panel . . . . .	8-12
Figure 8-10	System Definitions Panel . . . . .	8-12
Figure 8-11	Creating a Layer Object . . . . .	8-15
Figure 8-12	Started Task Overview Panel . . . . .	8-16
Figure 8-13	Defining a Layer Object . . . . .	8-16
Figure 8-14	Defining a Layer Object . . . . .	8-17
Figure 8-15	Defining a Layer Object: Valid Systems Pop-Up Panel . . . . .	8-19
Figure 8-16	System Definitions Panel . . . . .	8-20
Figure 8-17	Defining a Layer Object: Valid Systems Pop-Up Panel (2) . . . . .	8-20
Figure 8-18	Dependency Pop-Up Panel . . . . .	8-21
Figure 8-19	Creating a Layer Set . . . . .	8-23
Figure 8-20	Started Task Sets Panel . . . . .	8-23
Figure 8-21	Pop-Up Panel for Defining a Layer Set . . . . .	8-23
Figure 8-22	Defining a Layer Set . . . . .	8-24
Figure 8-23	Members of Set Pop-Up Panel . . . . .	8-24
Figure 8-24	Objects Pop-Up Panel . . . . .	8-25
Figure 8-25	Members of Set Pop-Up Panel with Object Selected . . . . .	8-25
Figure 8-26	Adding Layer Objects and Sets to a System Layer . . . . .	8-26
Figure 8-27	System Layers Panel . . . . .	8-27
Figure 8-28	Add System Layer Pop-Up Panel . . . . .	8-27
Figure 8-29	System Layers Panel: New System Layer Added . . . . .	8-28
Figure 8-30	System Layer Objects Panel . . . . .	8-28

---

Figure 8-31	System Layer Sets Panel .....	8-29
Figure 8-32	Adding System Layers to a Sysplex Layer .....	8-30
Figure 8-33	Sysplex Layers Panel: Adding System Layers .....	8-30
Figure 8-34	Add Sysplex Layer Pop-Up Panel .....	8-31
Figure 8-35	Sysplex Layers Panel: New Sysplex Layer Added .....	8-31
Figure 8-36	Sysplex Layer Definitions Panel .....	8-32
Figure 8-37	Attaching a System Layer to a Local System .....	8-33
Figure 8-38	Sysplex Matrix Panel: Attaching a System Layer to a Local System .....	8-33
Figure 8-39	Sysplex Matrix Panel: Attaching a System Layer to a Local System .....	8-34
Figure 8-40	System Definitions Panel: Attaching a System Layer to a Local System .....	8-34
Figure 8-41	Attach System Layer Pop-Up Panel .....	8-35
Figure 8-42	Attaching a System Layer to a Remote System .....	8-36
Figure 8-43	Sysplex Matrix Panel: Attaching a System Layer to a Remote System .....	8-36
Figure 8-44	Sysplex Matrix Panel: Attaching a System Layer to a Remote System .....	8-37
Figure 8-45	System Definitions Panel .....	8-37
Figure 8-46	Attach System Layer Pop-Up Panel .....	8-38
Figure 8-47	Attaching a Sysplex Layer to a Remote Sysplex .....	8-39
Figure 8-48	Sysplex Matrix Panel: Attaching a Sysplex Layer to a Remote Sysplex .....	8-39
Figure 8-49	Sysplex Matrix Panel: Attaching a Sysplex Layer to a Remote Sysplex .....	8-40
Figure 8-50	Attach Sysplex Layer Pop-Up Panel .....	8-40
Figure 10-1	Accessing the Conversion Utility .....	10-5
Figure 10-2	Administration Options Menu .....	10-5
Figure 10-3	CSM v6 to TOM Conversion Panel .....	10-6
Figure 10-4	CSM v6 to TOM Conversion Panel (Second Panel) .....	10-7
Figure 10-5	CSM v6 to TOM Conversion Panel (Third Panel) .....	10-8
Figure 10-6	CSM v6 to TOM Conversion Panel (Fourth Panel) .....	10-8
Figure 10-7	Conversion Log Panel .....	10-10



---

# Tables

Table 1-1	Where to Find Important Terms and Concepts	1-4
Table 3-1	Name, Description, Product, Last modified, STC name, Step name, and Type Fields	3-6
Table 3-2	This is a Model only and Modeled from Fields	3-9
Table 3-3	Control Field	3-11
Table 3-4	Set associations Field	3-15
Table 3-5	Valid Systems Field	3-18
Table 3-6	Schedules Field	3-22
Table 3-7	Defining an Object: Startup Fields	3-26
Table 3-8	Defining an Object: Stop command type, Stop command, Stop time out, and Stop retry commands Fields	3-31
Table 3-9	Defining an Object: Start retry and Stop retry commands Fields	3-35
Table 3-10	Defining an Object: Pre-start EXECs, Post-start EXECs, Pre-stop EXECs, and Post-stop EXECs Fields	3-39
Table 3-11	Startup validation and Shutdown validation Fields	3-45
Table 3-12	Abnormal termination validation Field	3-48
Table 3-13	Try alternate system upon failure Field	3-50
Table 3-14	Affinity Field	3-54
Table 3-15	Dependency Property value Fields	3-56
Table 3-16	Reinstatement, Count, and Require confirmation Fields	3-60
Table 3-17	Verify Start and Verify Stop Fields	3-62
Table 3-18	Example of Resolving Variables	3-65
Table 3-19	List of Reference Variables	3-65
Table 4-1	Starting a Set: Options	4-11
Table 4-2	Stopping a Set: Options	4-13
Table 4-3	Resetting a Set: Options	4-15
Table 5-1	Calendar Terms and Concepts	5-3
Table 5-2	Fields for Entering a Calendar Dependency	5-25
Table 5-3	Entry Descriptions for Figure 5-31	5-27
Table 5-4	Entry Descriptions for Figure 5-32	5-28
Table 5-5	Entry Descriptions for Figure 5-33	5-29
Table 5-6	Entry Descriptions for Figure 5-34	5-30
Table 5-7	Entry Descriptions for Figure 5-36	5-33

---

Table 5-8	Entry Descriptions for Figure 5-37	5-33
Table 5-9	Entry Descriptions for Figure 5-39	5-35
Table 5-10	Entry Descriptions for Figure 5-40	5-37
Table 5-11	Entry Descriptions for Figure 5-41	5-38
Table 6-1	Starting an Object: Options	6-7
Table 6-2	Stopping an Object: Options	6-10
Table 6-3	Resetting an Object: Options	6-12
Table 8-1	Layering Terms and Concepts	8-3
Table 8-2	Overview to Layering Tasks	8-5
Table 8-3	How Object Names Resolve after an Attach	8-42
Table 9-1	List of AOEXEC TOM Functions	9-5
Table 9-2	Return Codes for the AOEXEC TOM ACTIVATE Function	9-7
Table 9-3	Return Codes for the AOEXEC TOM BLOCK Function	9-9
Table 9-4	Return Codes for the AOEXEC TOM BOUNCE Function	9-11
Table 9-5	Return Codes for the AOEXEC TOM CHSTATUS Function	9-13
Table 9-6	REXX Stem Variables and Sample Returned Values: Object	9-15
Table 9-7	REXX Stem Variables and Sample Values: Set	9-16
Table 9-8	Return Codes for the AOEXEC TOM GET Function	9-17
Table 9-9	Return Codes for the AOEXEC TOM LIST Function	9-19
Table 9-10	Return Codes for the AOEXEC TOM LOCK Function	9-21
Table 9-11	Return Codes for the AOEXEC TOM MOVE Function	9-23
Table 9-12	Return Codes for the AOEXEC TOM RESET Function	9-26
Table 9-13	Return Codes for the AOEXEC TOM SETPROPERTY Function	9-28
Table 9-14	Return Codes for the AOEXEC TOM SHUTSET Function	9-29
Table 9-15	Return Codes for the AOEXEC TOM SHUTSYS Function	9-30
Table 9-16	Return Codes for the AOEXEC TOM START Function	9-32
Table 9-17	Return Codes for the AOEXEC TOM STOP Function	9-35
Table 9-18	Return Codes for the AOEXEC TOM SUSPEND Function	9-38
Table 9-19	Return Codes for the AOEXEC TOM SYSINFO Function	9-40
Table 9-20	Return Codes for the AOEXEC TOM UNBLOCK Function	9-42
Table 9-21	Return Codes for the AOEXEC TOM UNLOCK Function	9-44
Table 9-22	List of TOM Console Commands	9-47
Table 9-23	List of Additional TOM Console Commands	9-50
Table 10-1	Group Exception Messages	10-11
Table 10-2	Object Exception Messages	10-11
Table A-1	TOM @STATUS Values	A-1

---

---

# About This Book

This book contains an introduction to the MAINVIEW Total Object Manager (TOM) product component and describes how to use its applications.

## Who Should Read This Book

This book is intended for the system administrators or other personnel who would be implementing automation for managing Started Tasks (STCs) on their enterprise systems.

## How This Book Is Organized

This book is organized as follows. This book also contains an index.

Chapter/Appendix	Description
Chapter 1, "Introduction to the MAINVIEW Total Object Manager (TOM)"	introduces important terms, concepts, and processes that are used and referred to throughout the book
Chapter 2, "Implementing and Customizing TOM"	describes what TOM requires prior to starting TOM and other implementation information
Chapter 3, "Creating and Managing Objects"	describes how to use the Stated Task Definition panel to define objects that TOM will manage
Chapter 4, "Creating and Managing Sets"	describes sets, how to group objects into sets, and how to manage sets
Chapter 5, "Creating Calendar Bases"	provides a description and examples about how to create new entries in a Calendar Base that can be used when defining schedules for objects
Chapter 6, "Using the Started Task Overview Panel to Manage Objects"	describes how to use the Started Task Overview panel to view and manage objects

<b>Chapter/Appendix</b>	<b>Description</b>
Chapter 7, "Administrative Tasks"	describes how TOM communicates with other TOM systems in a TOM-plex, shutting down a system or set, and other administrative tasks
Chapter 8, "Managing Objects across Sysplexes with Layering"	describes how to use the Layering option to create Layer objects and sets which can then be used to create normal objects and sets on other systems and sysplexes
Chapter 9, "Using AOAnywhere TOM Functions and Console Commands"	describes the syntax for the AOAnywhere TOM functions that allow you to manipulate TOM objects from EXECs
Chapter 10, "Converting the Continuous State Manager Version 6 Repository"	describes the CSM to TOM migration tool that is provided with TOM for customers who want to try to migrate their CSM version 6 and higher definition databases to TOM object repositories
Appendix A, "TOM Statuses"	contains a table listing all the possible TOM statuses an object can have and a description of the status

---

## Related Documentation

BMC Software products are supported by several types of documentation:

- online and printed books
- online Help
- release notes and other notices

In addition to this book and the Help, you can find useful information in the publications listed in the following table. As “Online and Printed Books” explains, these publications are available on request from BMC Software.

Category	Document	Description
Installation and other related documents	<i>OS/390 and z/OS Installer Guide</i>	provides instructions for installing and maintaining CPO- or SMP-packaged BMC Software products.
	<i>MAINVIEW Installation Requirements Guide</i>	explains how to download the MAINVIEW for OS/390 product tape components, receive, apply, and accept product libraries, and access AutoCustomization
	<i>MAINVIEW Common Customization Guide</i>	describes how to customize the product manually; includes common steps for all MAINVIEW products
	<i>Using MAINVIEW</i>	describes how to use the common MAINVIEW interface.
	<i>Implementing Security for MAINVIEW Products</i>	provides information about securing specific views and services
	<i>MAINVIEW Administration Guide</i>	describes how to manage and maintain the operating environment for MAINVIEW products at your site.

## Online and Printed Books

The books that accompany BMC Software products are available in online and printed formats. Online books are formatted as Portable Document Format (PDF) files. Some online books are also formatted as HTML files.

### To Access Online Books

To view any online book that BMC Software offers, visit the Customer Support page of the BMC Software Web site at [http://www.bmc.com/support\\_home](http://www.bmc.com/support_home). You can also access PDF books from the documentation compact disc (CD) that accompanies your product.

---

Use the free Acrobat Reader from Adobe Systems to view, print, or copy PDF files. In some cases, installing the Acrobat Reader and downloading the online books is an optional part of the product-installation process. For information about downloading the free reader from the Web, go to the Adobe Systems site at <http://www.adobe.com>.

### **To Request Additional Printed Books**

BMC Software provides some printed books with your product order. To request additional books, go to [http://www.bmc.com/support\\_home](http://www.bmc.com/support_home).

## **Online Help**

MAINVIEW Total Object Manager includes online Help. In the MAINVIEW Total Object Manager ISPF interface, access Help by pressing **PF1** from any ISPF panel.

## **Release Notes and Other Notices**

Printed release notes accompany each BMC Software product. Release notes provide current information such as

- updates to the installation instructions
- last-minute product information

In addition, BMC Software sometimes provides updated product information between releases (in the form of a flash or a technical bulletin, for example). The latest versions of the release notes and other notices are available on the Web at [http://www.bmc.com/support\\_home](http://www.bmc.com/support_home).

---

# General Conventions

This book uses the following general conventions:

Item	Format	Example
information that you are instructed to type	bold	Type <b>END</b> to return to WKLIST.
specific (standard) keyboard key names	bold	Press <b>Enter</b> .
field names	bold	In the <b>Initial status</b> field, type one of the following values.
directories, file names, Web addresses, e-mail addresses, option names	bold	The BMC Software home page is at <b>www.bmc.com</b> .
commands, view names, nonspecific key names, keywords, parameters	uppercase	Use the TIME command to specify the current time and date.  Display the EMSTAT view.  Use the HELP function key.
commands that can be shortened	required letters uppercase and other letters in lowercase	You can use the CONtext command.
code examples, syntax statements, system messages, screen text	Courier font	//PARMLIB DD DSN= <i>hi level</i> .UBBPARM,  SAVE THIS MEMBER IN A PROCLIB AND CONTINUE ON TO THE NEXT STEP.
emphasized words, new terms, variables	italics	When finished, perform <i>one</i> of the following tasks.  <i>A view table</i> is a family of views that display the same type of data.

This book uses the following types of special text:

**Note:** Notes contain important information that you should consider.

**Warning!** Warnings alert you to situations that could cause problems, such as loss of data, if you do not follow instructions carefully.

**Tip:** Tips contain information that might improve product performance or that might make procedures easier to follow.

---

## Special Characters

This book uses the following special characters in MAINVIEW command notation:

Item	Use	Example
. (period)	used to direct a command to a specific window without changing the default window specification	W2.VIEWS
;(semicolon)	used to separate two or more independent commands	VIEWS;W3;JFLOW;ASU
? (question mark)	used as a wildcard character for a single character	W3;JFLOWS LGS?2
* (asterisk)	used with the CONtext command to specify the system to which you are currently connected  used with the TIME command to specify the current time or date (or both)	CONtext * MVMVS  TIME * *
= (equal sign)	used with the CONtext command to specify the system that is currently active in the window  used with the TIME command to retain the time or date (or both) set by a previously issued TIME command	CONtext = =  TIME = =

---

---

# Chapter 1 Introduction to the MAINVIEW Total Object Manager (TOM)

This chapter introduces the MAINVIEW Total Object Manager. This chapter includes the following topics:

Overview . . . . .	1-1
Benefits of TOM . . . . .	1-3
TOM Requirements . . . . .	1-3
TOM Terms and Concepts . . . . .	1-4
How TOM Works . . . . .	1-8

## Overview

The MAINVIEW Total Object Manager, also referred to as TOM, provides a basic infrastructure that manages objects such as started tasks, jobs, and networks in z/OS or OS/390 environments. The 6.3.01 release of MAINVIEW AutoOPERATOR provides the TOM infrastructure and the MAINVIEW Started Task Manager (STM), a plug-in module that you can use to manage the availability of Started Tasks (STCs).

TOM operates its own address space, which is associated with a single MAINVIEW AutoOPERATOR address space and acts as a server. With TOM, you can manage objects on a single system, throughout a sysplex, or across sysplexes.

TOM provides an object driver that you can use to

- define Started Tasks to TOM as either individual objects or grouped together in sets
- set up calendar dependencies where you can specify when objects should be stopped
- associate automation EXECs with an object that can be scheduled before TOM starts an object, after the object starts, before TOM stops the object, and after it stops
- define multiple systems on which an object is eligible to be started
- determine which available systems an object can be started on or moved to
- determine when an object should be active

TOM also provides

- a user interface to manage objects and sets of objects
- an API interface that you can use to interact with objects through EXECs
- console commands to control objects from a console
- a log for capturing TOM activity (including the starting and stopping of objects)

For more information about the object driver, see “How TOM Starts and Stops Objects” on page 1-10.

## Benefits of TOM

TOM provides the following important benefits:

- easy management of objects on different systems and sysplexes

With TOM, you can see the objects that are defined on a single sysplex. Also, when a new system is added to the sysplex where TOM is already operating, the TOM registry from an existing system is automatically copied to the new system.

- simple definition of objects

With TOM, you can specify just the Started Task name and step name for MVS Started Tasks, and TOM will manage the availability of those objects by checking the MVS control blocks for the presence of those Started Tasks.

Also, TOM objects can inherit definitions from a model object. If you make changes to the model object, all objects that are based on that model receive the same changes.

## TOM Requirements

For TOM, the MAINVIEW AutoOPERATOR for OS/390 option is required and a valid key for the MAINVIEW AutoOPERATOR for z/OS option must be present in BBPARM member BBKEYS. For more information about BBKEYS, refer to the *MAINVIEW Common Customization Guide*.

For information about implementing TOM, refer to Chapter 2, “Implementing and Customizing TOM.”

# TOM Terms and Concepts

The following sections describe important terms and concepts that are related to TOM:

**Table 1-1**      **Where to Find Important Terms and Concepts**

<b>Information</b>	<b>Page reference</b>
Objects	page 1-4
Sets	page 1-5
Status and Dependency Property	page 1-6
Schedule and Calendar Dependencies	page 1-6
TOM-plex	page 1-7
Single Image Mode	page 1-7
How TOM Starts	page 1-8
How TOM Starts and Stops objects	page 1-10
How TOM Recovers Failed Objects	page 1-12
How to Specify Start Command Limits	page 1-14

## Objects

You can define the following different types of objects in TOM:

- normal
- transient
- model
- layer

**Note:** In addition to the following sections, the introduction to Chapter 3, “Creating and Managing Objects” and Chapter 8, “Managing Objects across Sysplexes with Layering” also contain information about objects.

### Normal Objects

A normal object is a Started Task that, when defined to TOM, has its state controlled by TOM. Examples of these objects include TSO, CICS regions, and JES. After the objects are defined, TOM becomes completely responsible for automatically starting, stopping, and restarting the objects.

## Transient Objects

A transient object is an MVS Started Task that is started by TOM once during the life of TOM, but the state of this type of object is *not managed by TOM* after the object is started. A transient object, can, for example, be an object that starts, performs a batch process, and terminates normally when the job is completed. After TOM starts these objects, TOM ceases to monitor or manage the object.

## Models

Models are special objects that you can use to create objects that have identical or very similar definitions (without having to define each object separately).

Creating complex definition properties for an object can be very labor intensive when you need to define many similar objects that will run on many systems within a sysplex. When a model is defined, it can be used by other objects that have identical or similar properties. In addition, when you change any definition of a model, the change is automatically reflected in all objects that are based on that model.

Refer to “Defining a Model” on page 3-7 for more information.

## Layer Objects

With Layer objects, you can create objects for many systems within many sysplexes that are also running TOM.

Layer objects are part of a larger process that is described in Chapter 8, “Managing Objects across Sysplexes with Layering” on page 8-1.

## Sets

Sets are collections of (or other sets of objects) that TOM manages as a single entity. Refer to “Creating and Managing Sets” on page 4-1 for more information about when you might use sets, what actions you can take against a set, and how to create sets.

## Object Status and Dependencies

Every object that is managed by TOM has a status at all times. TOM attempts to keep the status of every object as ACTIVE (which means it is running and available). However, due to various system conditions and changes, an object can have many object statuses (refer to “TOM Statuses” on page A-1).

An object’s status can have profound effects when one object is defined to be dependent on the status of another object. This means that TOM will make an object’s status ACTIVE only when the dependency on another object (or set) is true. You can define these dependencies between objects (and objects and sets) on the Started Task Definition panel on the Dependency Property pop-up panel.

## Scheduling and Calendar Dependencies

When objects are defined to TOM with no schedule, TOM tries to maintain an object as ACTIVE (or available) 24 hours a day, 7 days a week. To define when TOM takes an object out of ACTIVE status, you can specify a schedule as part of the object’s definition on the Started Task Definition panel (described in “Creating and Managing Objects” on page 3-1).

You can also create calendar dependencies for other object definitions, such as

- Start and Stop retry commands for an object
- EXECs that are associated with starting or stopping an object

When you are defining these object properties, for example, you can specify for TOM issue the Start or Stop retry command, or for TOM to schedule any of the associated EXECs based on the date or time of day.

To simplify scheduling and calendar dependencies, TOM provides the Calendar Bases feature (Option 3 from the TOM Primary Options Menu). With Calendar Bases, you can create dates and times (called calendar entries) and use them in an object’s Schedule pop-up panels, or use them wherever a hyperlink for Calendar Dependencies appears for the other eligible object properties.

To learn more about defining a Schedule, refer to “Defining a Schedule” on page 3-19.

For a complete discussion of Calendar Bases including examples for how to use calendar entries as part of the object’s definition, refer to “Creating Calendar Bases” on page 5-1.

## TOM-Plex

A TOM-plex differs from a sysplex. Within a sysplex, many systems can exist and each of these systems can have a TOM system running on it. The TOM systems that are running within a sysplex are referred to as a TOM-plex. A TOM-plex can be a subset of the sysplex or, by having a TOM system started on every system, can be as large as the sysplex.

A TOM system can manage the status of objects on another system within the sysplex only when another TOM system has been started that system. Refer to “How TOM Communicates with Other TOM Systems in a Sysplex” on page 7-2 for more information.

For example, suppose you are logged on to the TOM1 system on SYSA. Your sysplex has systems SYSA, SYSB, and SYSC. Unless TOM systems are available on systems SYSB and SYSC that TOM1 can communicate with, the status of objects on these systems cannot be properly determined and are set to STOPPED. Also, objects cannot be moved between SYSA and SYSB.

When TOM systems are started on systems in the sysplex, they connect with the existing TOM system(s) and then the objects on those systems can be managed by TOM (and the size of the TOM-plex increases).

In addition, you can start a TOM system in *Single Image Mode* so that this system is considered “invisible” in the TOM-plex and cannot communicate with the other TOM systems. Refer to “Single Image Mode” for more information.

## Single Image Mode

To start a TOM system in single image mode, specify the SIM keyword on the OS parameter of the TOMALLOC JCL as shown in the following example:

```
//STEP1 EXEC PGM=TXLOAD,PARMS='SIM'
```

When a TOM system is started in single image mode, you can define all objects for the system, and these additions and revisions will not affect any of the other TOM systems within the sysplex. You can do this process for testing, or when creating very complex definitions that you do not want to be shared by the other TOM systems yet, or when you are performing maintenance tasks on an individual system.

Running a system in single image mode is the only way that a TOM system within the sysplex can use an object Registry that contains unique sets of definitions. In addition, TOM systems in single image mode, do not perform any Registry synchronization or log message updating with the other TOM systems in the sysplex.

The following considerations apply:

- If an object's definition specifies that the object can be started on the system that is running a TOM system in single image mode and on another system in the sysplex, it is possible for this object to be started on these two systems. Normally, an object can be started on only one system.
- Any updates that you make to the TOM system in single image mode will be lost when you stop that system and restart it as part of the TOM-plex (see "Overview of Cloning" on page 7-2 for more information).
- Any log data that is recorded on the single image mode TOM system is not merged with log data of the other TOM systems in the TOM-plex.

## How TOM Works

The following sections describe how TOM systems start, how TOM starts and stops objects, how TOM recovers failed objects, and how you can specify command limits for an object.

### How to Start TOM

TOM uses automation functions provided

- to define events that signify when an object starts, stops or ends abnormally
- to schedule and execute Pre-start, Post-start, Pre-stop, and Post-stop EXECs as part of an object's definition.

Therefore, MAINVIEW AutoOPERATOR must be started and MAINVIEW AutoOPERATOR services must be available before TOM can start or stop any object. If MAINVIEW AutoOPERATOR is not running, any attempt to start or stop objects causes the objects state to be set to FAILURE-AOLINK.

If you attempt to start a TOM system before MAINVIEW AutoOPERATOR is active, the following message is issued:

```
TX2002W Connection to <ssid> lost - TOM processing
disabled
```

where *ssid* is the subsystem ID of the MAINVIEW AutoOPERATOR PAS specified in the TOM startup JCL.

This message indicates that the TOM system cannot detect the MAINVIEW AutoOPERATOR PAS that is specified in the TOM startup JCL. This message is reissued every 30 seconds until the MAINVIEW AutoOPERATOR PAS becomes available.

When the specified MAINVIEW AutoOPERATOR PAS is available, the following message is issued:

```
TX2100I Connection to KMZ2 established
```

where *KMZ2* represents the MAINVIEW AutoOPERATOR subsystem ID.

TOM should be started after MAINVIEW AutoOPERATOR issues the following message:

```
AA1113I AUTOOPERATOR RELEASE v.r.m INITIALIZED
```

After the AA1113I message is issued, you can start TOM by issuing the MVS start command.

```
S procname.optional Step Name,optional Parameters
```

where

- *procname* represents the name of the procedure that you use to start the TOM address space
- *optional Step Name* represents a one to eight-character Started Task step name of this TOM address space.
- *optional Parameters* represents any other parameters that you choose to use when starting TOM

When TOM is fully initialized, the TOM system issues the message:

```
TX0110I TOM tom ID version 1.1.0 initialized
```

TOM can now access the required automation services and you can perform TOM procedures when the following message is issued:

```
BD0105I TOM X-mem services are available
```

## How TOM Starts and Stops Objects

TOM uses an object driver component to start or stop an object. TOM also performs auto-discovery processes every 60 seconds to determine whether the state of an object has changed or needs to be changed. For each object, the auto-discovery process determines:

- what calendar or schedule criteria is specified for the object
- what cross-system dependencies are defined
- what other object dependencies are defined for the object

To track an object's state, TOM monitors the Started Task that the object represents as it runs on the system. The object driver also tracks all properties that were defined for the object (its schedule, dependencies, and so on). When the object driver determines that the state of an object needs to be changed, the driver also determines how to obtain that state.

For example, when an object is not active but is supposed to become active, TOM issues the object start command (or EXEC) that is defined with this object.

The object driver passes the start request to the initiation controller which starts a task to process this request. Each task acts on behalf of a specific object, and only one task can be processing for one object at a time.

When the task begins, it determines if a Start verification message was specified as part of the object definition. If the Start verification message is specified, the task waits until you reply YES to the outstanding write-to-operator (WTOR) message. Activity that might be occurring for other objects is not impacted during this time.

After you respond to the Start verification message, TOM compares the object's Start command count value for the target system the against the maximum value. If the limit has not been exceeded, processing continues.

TOM checks the object definition for any Pre-start EXECs (or Pre-stop EXECs). You can specify many EXECs as part of an object's startup or shutdown process.

Because certain objects might have to start in different conditions, you can also specify that a Pre-start (or Pre-stop) EXEC has a calendar dependency, where the EXEC is only scheduled when the time requirements in the Calendar dependency are met (refer to "Entering Calendar Dependencies for Object Definitions" on page 5-22). When a schedule that is associated with an EXEC indicates that the EXEC should not be run, TOM evaluates the next EXEC (if applicable).

When an EXEC does not have a schedule defined for it (or the schedule indicates that it is time to process the EXEC), the task waits for each EXEC as it is scheduled and processed by the MAINVIEW AutoOPERATOR PAS. TOM checks the return codes from the EXECs. A return code of zero means that TOM can schedule the next Pre-start (or Pre-stop) EXEC. When all return codes are zero (indicating that the EXEC completed successfully), TOM schedules the next EXEC and continues until all EXECs are complete.

**Note:** If an EXEC returns a non-zero return code, the object is not started (or stopped), and TOM sets the object's status to FAILURE-PRESTART (or FAILURE-PRESTOP).

At this point, TOM attempts to intercept the start, stop, and termination events (messages) that are defined for the object. In the case of Started Task objects, TOM communicates event information to the associated MAINVIEW AutoOPERATOR PAS, and MAINVIEW AutoOPERATOR Rules are dynamically defined to the MAINVIEW AutoOPERATOR Rules Processor.

Each Rule can have a message ID, message text, job or started task name, and a step name defined so when the message for an object is detected by the Rule, TOM knows that the start, stop or abend message has been issued for a particular object.

**Note:** The MAINVIEW AutoOPERATOR Rules working inside TOM processing cannot be seen from the MAINVIEW AutoOPERATOR Automation Control panel and cannot be disabled. These Rules exist only for the life of the object and do not affect the performance of the Rules Processor in MAINVIEW AutoOPERATOR.

At this point, TOM issues the object's start (or stop) command. If you use variables in the object's start command, those variables are resolved at this time. After issuing the command, the task waits for the amount of time that is specified in the timeout value. When the object starts (or stops) and it generates an event that matches the one defined, the Rules Processor notifies TOM. When TOM receives the notification, it interrupts the waiting task and cancels the wait. The object is placed in an ACTIVE state (or STOPPED state when TOM stops an object).

If TOM does not receive a notification from the Rules Processor before the timeout value is exceeded, the wait period ends and TOM issues any recovery commands that might be part of the object's definition. If the notification still does not occur after TOM issues all the commands, TOM places the object in a state named FAILURE-REC.

## How TOM Recovers Failed Objects

TOM automatically attempts to recover objects from the following types of failures:

- command failures
- abnormal terminations

### Command Failures

Command failures might occur when TOM issues a start or a stop command against a defined object.

When TOM issues a start command or stop command for an object, TOM waits for a defined amount of time. During that time, TOM waits to receive user-specified events (refer to “Defining Events for Object Startup or Shutdown” on page 3-43).

User-specified events are not required as part of an object’s definition. When an event is not specified, TOM periodically checks for evidence of the successful initiation (or shutdown) of the address space for this object. When TOM detects that the address space has started (or stopped), TOM continues to track the state of the object.

However, if a user-specified event is defined and TOM does not receive notification of this event before the specified time limit is exceeded, processing for this object goes into recovery mode.

**Note:** When a user-specified event is specified, TOM must detect the existence of an address space and must also receive the user-specified event’s message to consider the object successfully initialized.

To perform recovery for a failed command, TOM uses the user-specified list of optional retry commands that you can define as part of each object’s definition. TOM issues the retry commands until they are all issued or TOM is notified that the user-specified event has occurred.

When no retry commands are specified, the object’s status is set to FAILURE-REC-INIT (for objects that did not start) or FAILURE-REC-TERM (for objects that did not stop).

The following factors can determine whether a retry command is issued and how it is issued:

- calendar dependency

TOM selects the first retry command that is defined for the object. Each retry can be defined with a calendar dependency, which specifies when a command may be issued (or not). If the time that is specified in the dependency definition indicates that the command is not to be issued at this time, TOM moves to the next command in the list.

- command count

Each start or stop retry command has a command count attribute and a command interval attribute. The count indicates how many times the command can be issued. The interval controls how long to wait between issuing commands. The task that issues the command waits either until TOM received the user-specified event (object is now ACTIVE for a start, STOPPED for a stop) or until the waiting period expires.

When the wait period is exceeded, TOM checks the command count attribute. If the command count has not been exceeded, TOM continues to issue the start (or stop) command. When the count is exceeded, the counter is reset, and the next command is issued.

When all retry commands are exhausted, the object's status is set to FAILURE-REC-INIT (for objects that did not start) or FAILURE-REC-TERM (for objects that did not stop).

## Abnormal Terminations

Abnormal terminations can occur for any object when the object terminates before TOM has instructed the object to stop. When you define an abnormal termination event (optional) for the object as part of its definition and the event occurs, TOM is notified that an abnormal termination has occurred. Refer to "Defining Events for Abnormal Object Termination" on page 3-46.

When an abnormal termination takes place, TOM takes the following actions:

- increments the abend counter for the object
- sets the object's state to FAILURE-ABENDED
- issues the IC1704I message:

```
IC1704I Abend event detected for object_name
```

However if you define that the object should be started on an alternate system when it fails on the current system (refer to “Defining Alternate Systems” on page 3-49), TOM attempts to restart the whole process on the next system in the object’s definition until the object restarts or the startup process has failed on each system on the list.

If all attempts have been made, and the object cannot start or stop successfully, on the Started Task Overview panel, TOM displays **YES** under the **EXCEPTION** column heading.

When this situation occurs, you must determine why the object has not started, make adjustments or adjust the object’s definition, and let TOM try to start the object.

## How to Specify Start Command Limits

You can define a maximum number of times that TOM attempts to start a failed object. For example, you specify a value of 3 for the start command limit. When TOM detects a failed object, TOM automatically issues the start command, trying to restart the object. If TOM cannot start the object, TOM displays **YES** under the **EXCEPTION** column heading on the Started Task Overview panel and the object’s state is set to **FAILURE-REC-INIT**.

If the object eventually restarts and fails again, TOM issues the start command (second attempt). If the object restarts and fails again, TOM issues the start command (third and final attempt).

If the object fails a fourth time, TOM does not attempt to restart the object and issues the IC1202W message. The object’s state is set to **FAILURE-CMDCOUNT**, indicating that it has exceeded the specified number of times that TOM should attempt to restart the object.

When an object has exceeded its start command limit, you can restart the object in one of following ways:

- On the Started Task Definition panel, enter the **S** (for Start) line command next to the object and enter **Y** (for yes) for the **Bypass Command Count Check?** field on the verification pop-up panel. Refer to “Starting an Object” on page 6-6.
- Write and schedule an EXEC that issues a TOM API function to reset the start command count (specifying the keyword **STARTCMDCT**). Refer to “RESET Function” on page 9-25.

- On the Started Task Definition panel, enter the R (for RESET) line command next to the object and enter **Y** (for yes) for **Reset Start Command Count? field** on the verification pop-up panel. Refer to “Resetting an Object” on page 6-11.

You can also define that TOM automatically resets the object’s start command count when

- an object successfully terminates
- after a prescribed number of minutes following a successful start

**Note:** The command count is reset automatically for all objects when TOM is initialized after an IPL.

To define that a reset takes place after a successful object termination, specify **Y** (for Yes) in the **Reset start count at termination** field on the Started task Definition panel as part of the object’s definition.

To define that an object’s start command count resets at some point after a successful start, specify a number of minutes in the **Reset start count after: x minutes** field on the Started task Definition panel.

**Tip:** You can choose to not have the count reset after an IPL by specifying **NO** in both fields.



---

---

# Chapter 2    Implementing and Customizing TOM

This chapter describes how to implement and customize the TOM application after it is installed.

This chapter includes the following topics:

Preparing to Start and Use TOM .....	2-2
Starting the TOM Address Space .....	2-2
Running Multiple TOM Systems within a Sysplex .....	2-4
Starting TOM for the First Time .....	2-6
Implementing Security for TOM Systems .....	2-7
Additional Information .....	2-7

## Preparing to Start and Use TOM

To use TOM, you must perform the following actions:

- Install and bring up MAINVIEW for AutoOPERATOR version 6.3.01.

MAINVIEW AutoOPERATOR is required for TOM to work properly.

- Apply IBM PTF UA00713 to your system.

IBM APAR OW56486 documents a problem that affects TOM ISPF panels, including the TOM Started Task Definition panel. The problem involves incorrect scrolling on panels that contain hyperlink fields.

The IBM PTF UW94273 resolves the issue described in this APAR.

Furthermore, IBM PTF UA00713 supersedes PTF UW94273. BMC Software recommends that you consult IBM Customer Support to determine the latest and most appropriate PTF to apply to solve the problem described in APAR OW56486.

- Apply the IBM PTF that fixes IBM APAR OA03333.

This fix prevents an abend S378 from occurring when you are trying to use the online Help in TOM.

## Starting the TOM Address Space

Perform the following actions:

- allocate a pair of log data sets and a registry data set, modifying the data sets for your site, and submit the TOMALLOC JCL member that was supplied in the BBSAMP library

**Warning!** Registry and log data sets cannot be shared between TOM address spaces. Attempting to do so can cause unpredictable results! In addition, all Registries should be defined as having the same size. Refer to “Sharing Data Sets” on page 2-4.

- prepare a Started Task

These tasks can be accomplished through AutoCustomization, which includes the corresponding steps “Allocate TOM data sets” and “Create TOM start procedure”. If you do not use AutoCustomization, consult the *MAINVIEW Common Customization Guide* for procedures to manually perform these tasks.

## Running Compiled REXX Procedures under TSO

The ISPF user interface for TOM is written in compiled REXX.

**Note:** If your environment already supports running compiled REXX programs, proceed to the next section, “Start MAINVIEW AutoOPERATOR” .

**Step 1** Enable your environment for running compiled REXX.

**1.A** If you do not have a copy of the IBM REXX/370 Library, activate the free REXX/370 Alternate Library by following the instructions in Chapter 9, “Activating the REXX/370 Alternate Library,” in the *MAINVIEW AutoOPERATOR Customization Guide*.

The REXX/370 Alternate Library supports running compiled REXX programs and provides performance similar to running interpreted REXX.

**Step 2** Ensure that each TSO user ID that accesses the TOM ISPF user interface has a STEPLIB to the BBLINK data set containing the REXX/370 Alternate Library modules.

If the TSO user ID does not have a STEPLIB (for example, you place the BBLINK data set in the LINKLIST concatenation), the ID will load module IRXCMPTM from the LPA instead of the BBLINK data set making the REXX/370 Alternate Library ineffective.

Therefore, you *must* load IRXCMPTM from the BBLINK data set for the REXX/370 Alternate Library to be effective.

## Start MAINVIEW AutoOPERATOR

You must always start MAINVIEW AutoOPERATOR before starting a TOM system. If you attempt to start a TOM system without first starting MAINVIEW AutoOPERATOR, TOM cannot start any objects, and all objects will have a status of FAILURE-AOLINK.

## Running Multiple TOM Systems within a Sysplex

This section describes two considerations that are important when you are planning to implement multiple TOM systems in a sysplex:

- how to share data sets
- how to group multiple TOM address spaces

### Sharing Data Sets

To enable sharing data sets, you must allocate a registry data set and two log data sets for each TOM address space that you plan to operate.

If you do not use AutoCustomization, you must tailor these data sets appropriately and submit the TOMALLOC JCL member that was supplied in the BBSAMP library. Refer to the *MAINVIEW Common Customization Guide* for documentation about how to accomplish these tasks manually.

**Warning!** Registry and log data sets cannot be shared between TOM address spaces. Attempting to do so can cause unpredictable results! In addition, all Registries should be defined as having the same size.

### Grouping Multiple TOM Systems in a Sysplex

By default, each TOM address space synchronizes with and broadcasts to all other TOM address spaces that are running within the same sysplex. This process allows each TOM to effectively monitor and manage the defined objects on each LPAR.

Any changes in an object's status or definition are automatically broadcast to the other TOM systems in the sysplex by way of XCF communications. All TOM systems in a sysplex are always in sync or are in the process of synchronizing with each other.

You might, however, want to group specific TOM systems within the sysplex and limit the broadcasting of information to only those TOM systems.

As an example, suppose you have six LPARs in a sysplex. Each LPAR has one TOM address space (or system) to manage its objects. Three of the LPARs perform production processing. Two LPARs are allocated for testing applications. The last LPAR is used for development work. You want to keep the administration, monitoring, and management of the three groups separate.

To separate the TOM systems:

- Step 1** Edit BBPARM member XCFDEF00, GROUP keyword. The default value for the keyword is BMCTOM.
- Step 2** Specify a one- to eight-character alphanumeric value for GROUP.

The TOM address space that uses this parameter member broadcasts and receives only to other TOM systems that are also defined in this group.

**Note:** TOM supports only one GROUP setting for each LPAR. For example, if you specify GROUP(BMCTOM) for a TOM system named TOM1, a second TOM system named TOM2 with GROUP(BMCTOM) cannot initialize on the same LPAR.

The following example shows sample contents of the BBPARM member XCFDEF00 and how to specify a new GROUP name called PROD:

---

```
* PURPOSE: TOM XCF COMMUNICATIONS
*
* PARAMETER(DEFAULT) ----- DESCRIPTION -----
*
* DEBUG(NO)          OPTIONAL, WRITES DEBUGGING MESSAGES TO
*                    JOB LOG. CAUTION - CAN GENERATE EXTREME
*                    MESSAGE VOLUME. USE ONLY AT THE
*                    DIRECTION OF BMC SOFTWARE PRODUCT SUPPORT
*
* GROUP(*NONE*)      THE XCF GROUP NAME THAT THIS TOM WILL BE
*                    ASSOCIATED WITH
*                    CHANGE ONLY IF YOU REQUIRE DIST GROUPS OF
*                    TOMS IN THE SAME SYSPLEX
*
*                    1 - 8 CHARACTERS (A-Z, 0-9, NATIONAL
*                    CHARACTERS), AVOID 'SYS' AND 'UNDESIG' *
DEBUG(NO)
GROUP( PROD)
```

---

The XCFDEF00 member can be shared among the three TOM address spaces that are managing the production resources. This member is stored in a data set that is associated with the TOMPARM DD in the TOM address space procedure.

## Starting TOM for the First Time

**Note:** Before attempting to start TOM, you must perform the steps described in “Preparing to Start and Use TOM” on page 2-2.

TOM can run the first time you start TOM with an empty registry.

**Step 1** Start TOM with an MVS start command.

You are prompted to initialize the registry and the log data sets. Reply as indicated in the messages.

**Note:** This initialization process occurs only when a new data set has been allocated and you are starting a TOM system for the first time. After the first time, you should start TOM with a backup copy of the Registry.

**Step 2** Select option 8 from the MAINVIEW AutoOPERATOR menu (Figure 2-1) to display the Automation Menu (Figure 2-2 on page 2-7).

**Figure 2-1** Accessing the AutoOPERATOR Primary Option Menu

```

----- MAINVIEW AutoOPERATOR-----
OPTION  ===>                                DATE  -- 2003/10/09
                                           TIME  -- 15:12:05
                                           USERID -- BAONIS2
                                           MODE  -- ISPF 5.2

Operator Workstations
  1 ALERTS      ALERT Management
  2 MVS         OS/390 or z/OS Resources
  3 CICS        CICS Resources
  4 IMS         IMS Resources
  5 NETVIEW     NetView Resources
  6 TAPESHARE   Tape Drive Management
  7 MQSERIES    WebSphere MQSeries

Administration
  8 AUTOMATION  Basic and Advanced Automation
  9 PARMS       Dynamic Parameter Manager

General Services
  C CYCLE       Service Refresh Cycle Setup
  L JOURNAL     Journal Log
  M MESSAGES    Messages and Codes

                                Copyright BMC Software, Inc. 2003

```

**Figure 2-2** Selecting TOM from the Automation Menu

```

BMC Software ----- AUTOMATION MENU ----- AutoOPERATOR
OPTION ===>                                DATE -- 2003/10/09
                                           TIME -- 12:24:35

Basic Automation:
  1 Event Activity Statistics
  2 Display/Modify Rules and Rule Sets
  3 Continuous State Manager - Global Overview
  4 Total Object Manager

Advanced Automation:
  6 Shared Object Facility
  7 Display/Modify EXEC Status
  8 Time-Initiated EXEC Requests
  9 Open Systems Procedural Interface (OSPI)

                                           PF1/13 HELP PF3/15: EXIT

Copyright 2003, BMC Software, Inc. All rights reserved.

```

**Step 3** Select option 4 to access TOM.

**Note:** If you are an existing CSM user, you may consider converting the CSM repository to a TOM registry. See “Converting the Continuous State Manager Version 6 Repository” on page 10-1 for CSM database conversion instructions.

## Implementing Security for TOM Systems

Refer to *Implementing Security for MAINVIEW Products* for information about implementing security for accessing the panels in the TOM user interface, issuing TOM API functions, and other areas of TOM functionality.

## Additional Information

The MAINVIEW TOM component ships with the following TOM-specific members in BBPARM:

- BDINIT00
- CMDMGR00
- CNINIT00
- ICINIT00
- LCKMGR00

- LOGGER00
- ODINIT00
- REGSTY00
- SCINIT00
- XCFDEF00

For information about these members, refer to the important comments within these BBPARM members. You should not need to edit the settings in these members for TOM to be implemented and running properly.

## Specifying Multiple BBPARM Members for TOM

The TOMCFG DD statement allows an installation to maintain a set of TOM-specific BBPARM members in a single data set. Every TOM-specific BBPARM member can have a two-character, alphanumeric suffix.

The default for these two characters is 00; for example: ODINIT00, LOGGER00, XCFDEF00 and so on.

You can override one or more of the default suffixes by specifying a TOMCFG DD statement in the JCL of the TOM Started Task; for example:

```
//TOMCFG DD DISP=SHR,DSN=dsn
```

where `dsn` is the name of a sequential data set that can contain any number of lines where all of the suffixed members are listed. Otherwise, the names of the members of the TOM-specific BBPARM data set is specified by using the SUFFIX() keyword; for example:

Example:

```
SUFFIX(ODINIT01)  
SUFFIX(REGSTYXX BDINIT02)
```

The parameter name suffix must always be two characters; do not list the same member name more than once.

**Note:** This syntax differs from the BBICFG DD statement for the JCL of the MAINVIEW AutoOPERATOR PAS, which is documented in the *MAINVIEW Common Customization Guide*.

---

---

# Chapter 3      **Creating and Managing Objects**

This chapter includes the following topics:

Overview . . . . .	3-2
Defining a Model . . . . .	3-7
Defining an Object in SUSPEND Mode . . . . .	3-10
Defining a Layer Object . . . . .	3-12
Viewing Set Associations . . . . .	3-13
Defining Valid Systems for the Object to Start On . . . . .	3-16
Defining a Schedule . . . . .	3-19
Defining How to Start an Object . . . . .	3-23
Defining How to Stop the Object . . . . .	3-29
Defining Start or Stop Retry Commands . . . . .	3-32
Associating EXECs with an Object . . . . .	3-36
Defining Events for Object Startup or Shutdown . . . . .	3-43
Defining Events for Abnormal Object Termination . . . . .	3-46
Defining Alternate Systems . . . . .	3-49
Defining System Affinity . . . . .	3-51
Defining Object Dependencies . . . . .	3-55
Defining Reinstatement of an Object . . . . .	3-58
Verifying the Starting or Stopping of an Object . . . . .	3-61

# Overview

This section describes basic concepts about objects.

## Types of Objects

With the MAINVIEW AutoOPERATOR Started Task Manager (STM) option, you can define the following types of objects:

- objects that represent Started Tasks

You can define

- normal objects (where TOM completely manages the start, stop, restart, recovery, and other aspects of the object)
- transient objects (where TOM starts the objects once after an IPL but does not continue to monitor and manage the state of the object for the rest of the IPL cycle)

Examples of transient objects are batch processes that TOM starts after an IPL but does not attempt to restart when the jobs end.

- models

You can create a single definition from which to create similar objects. You can also create object definitions from a model and selectively override certain definitions with object-specific definitions.

**Note:** Models are never started.

For more information about models, refer to “Defining a Model” on page 3-7.

- Layer objects, which are objects that you create to propagate an object’s definitions and changes to other sysplexes that are running TOM

**Note:** Layer objects are never started.

For more information about layer objects, refer to Chapter 8, “Managing Objects across Sysplexes with Layering” on page 8-1.

## How Objects Are Stored

Objects and their definitions are stored in definition databases. When TOM first starts, a default definition database is created, named DEFAULT. You can create multiple definition databases to address different situations, such as

- having a test database that you can use temporarily when making a large number of changes
- creating a disaster-recovery database that you can use to quickly re-establish a functioning system
- developing alternative databases that you can use for weekend activity or to meet other recurring system requirements

**Note:** Only one definition database can be active for a sysplex at any one time.

## Defining an Object

Objects and their definitions are entered on the Started Task Definition panel. The following procedure briefly describes how to enter values on the Started Task Definition panel to define an object:

- Step 1** To access the Started Task Definition panel, enter **1** on the **Option** line of the TOM Primary Options Menu, as shown in Figure 3-1.

**Figure 3-1 TOM Primary Options Menu**

```

BMC Software                TOM Primary Options Menu

Option ==> 1

1 Started Task      - Define, display and manage objects      User ID: BAOMXY1
2 Sets             - Define, display and manage sets          Server Id: TOMN1
3 Status           - Define, display and manage systems        Release: 1.1D
4 Calendar         - Define, display and manage calendars      System: SJSD

A Administration   - System Administration
M Messages         - List messages
L Log              - Display Log

X Exit             - Exit

                                Copyright BMC Software, Inc. 2003

```

The Started Task Definition panel is displayed. This procedure describes the following fields in the upper portion of the panel (see also Table 3-18 on page 3-65):

- **Name**
- **Description**
- **Last modified** (display-only field)
- **Product** (display only field)
- **STC name**
- **Step name**
- **Type**

Figure 3-2 shows an example of the upper portion of the Started Task Definition panel.

**Figure 3-2** Defining an Object: Name, Description, Product, Last modified, STC name, Step name, and Type

```

Started Task Definition
Command ==>

General information                                     More:  +
Name: CICS.SYSTEM1
Description: CICS on SYSTEM1                          Product STM
Last modified: 2003/07/17 15:31:21 BAORMB3
STC name: CICS0001 Step name:                          Type: N ( TRAN or NORM )
.
.
.

```

**Step 2** In the **Name** field, enter a 1- to 64-character name of a Started Task that TOM will monitor.

Table 3-1 on page 3-6 describes valid naming conventions.

**Step 3** In the **Description** field, enter a 1- to 40-character description of the object.

**Step 4** In the **STC name** and **Step name** fields, enter the STC name and step name of the object that TOM will monitor.

**Step 5** In the **Type** field, enter N for a Normal object or T for a Transient object.

**Note:** The **Product** and **Last modified** fields are display-only fields.

Table 3-1 describes the fields in the upper portion of the Started Task Definition panel.

**Table 3-1 Name, Description, Product, Last modified, STC name, Step name, and Type Fields**

Field Name and Description	Valid Values
<p><b>Name</b> Specify a required, user-defined object name of a started task that TOM will monitor. You can use combinations of numbers, upper and lower case letters. Object names are case sensitive which means an object with the same characters as another object but using different capitalization creates a new separate object.</p> <p>The object name can contains any characters, including blanks, except:</p> <ul style="list-style-type: none"> <li>• A slash (/) anywhere within the object name</li> <li>• A single plus sign (+) as the object name</li> </ul>	<p>1 to 64-characters Default: None Required: Yes</p>
<p><b>Description</b> Specify a brief description of the object. On several panels such as the Started Task Overview panel, the Description is shown (by default) as the first column. It is recommended that you provide a description so that you can easily identify the objects on the panels.</p>	<p>1 to 40-characters Default: None Required: No</p>
<p><b>Product</b> Display-only field showing that the MAINVIEW AutoOPERATOR Started Task Manager (STM) option is installed.</p>	<p>STM</p>
<p><b>Last Modified</b> Display-only field that shows the date, time and user ID of the last person who saved changes to this object definition.</p>	<p>N/A: Display-only field</p>
<p><b>STC name</b> Specify the name of a Started Task (STC) that TOM will monitor.</p> <p>Although it is not required that you specify the STC name when defining an object, it is recommended that you specify an STC name so that TOM can keep track of the object. For example, the object can be started from an EXEC or with a console command. In these cases, when the STC name is specified, TOM can automatically discover the object (a process called autodiscovery) and continue to monitor and control the object. When the STC name is not defined, TOM cannot determine the state of the object.</p>	<p>1 to 8 characters Default: None Required: No</p>
<p><b>Step name</b> Specify the step name of the started task to be monitored. If both an STC name and Step name are specified, TOM uses values from both these fields to track objects.</p> <p>Therefore, when you use the STC name field and the Step name fields, you should not create 2 objects with the same STC name and the same Step name. If you do, TOM cannot tell the difference between the objects.</p>	<p>1 to 8 characters Default: None Required: No</p>
<p><b>Type</b> Defines the object type. Objects can be</p> <ul style="list-style-type: none"> <li>• Normal: TOM controls the start and stop of Normal objects</li> <li>• Transient: TOM controls only the start of Transient objects.</li> </ul>	<p>N for Normal or T for transient Default: N Required: No</p>

## Defining a Model

Models are special objects that you can use to create objects that have identical or very similar definitions as the model (without having to define each object separately).

For example, you might need to define many CICS objects for each system in a sysplex. For each CICS object, you might only need to change the definition for each object's Step name property and all the other properties should be identical.

By creating a model CICS object first, specifying the rest of the CICS objects are modeled after this object, and specifying the unique Step name property for each additional CICS object, you can save the effort of individually creating multiple CICS objects. All of these CICS objects that are modeled after the model CICS object have their properties resolved at runtime.

When the objects derive their properties from a model, they are *inheriting* their definitions.

## Inheritance

When you create a model, TOM does not actually start and stop that model. Instead, when TOM starts, all objects whose definitions are based on a model inherit their definitions based on the model's definitions at runtime. This process prevents you from having to enter identical or nearly identical definitions for multiple identical or similar objects.

In addition, when you edit and change any definitions of a model, the new definition is automatically reflected in all the objects that are based on that model.

## How to Define a Model

This section describes how to use the fields on the Started Task Definition panel to define a model.

Figure 3-3 shows an example of the upper portion of the Started Task Definition panel. The **This is a Model only** and **Modeled from** fields are highlighted in bold.

**Figure 3-3 This is a Model only and Modeled from Fields**

```

Started Task Definition
Command ==>
More:      +
General information
Name: CICS.MODEL1
Description: CICS Model for BBSYSPLEX          Product STM
Last modified: 2003/07/17 15:31:21 BAORMB3
STC name: CICS0001 Step name:                   Type: N ( TRAN or NORM )
This is a Model only: Yes ( Y or N )
Modeled from:
.
.

```

The following procedure briefly describes how to enter values for the model fields on the Started Task Definition panel:

- Step 1** In the **Name** field, enter the name of the model (for example, **CICS.MODEL1**).
- Step 2** In the **This is a Model only** field, enter **Y** (for Yes).
- Step 3** Fill in the rest of the object definitions on the panel for the model.

When other objects are modeled from this object, they will inherit these definitions.

- Step 4** After all the model definitions are entered, press PF3 to save to model.

## Specifying an Object That Inherits All Properties from a Model

To define an object that will inherit all of the model's definitions:

- Step 1** In the **Name** field, enter the name of the object.
- Step 2** In the **Modeled from** field, enter the name of the model from which you want to create the new object.
- Step 3** Press PF3 to save the object.

The object inherits all property definitions of the model at runtime.

## Specifying an Object That Inherits Some Properties from a Model

To define objects that will inherit some (but *not* all) of the model's definitions:

- Step 1** In the **Name** field, enter the name of the object.
- Step 2** In the **Modeled from** field, enter the name of the model from which you want to create the new object.
- Step 3** Enter the object-specific definitions for the new object in the appropriate fields on the Started Task Definition panel.
- Step 4** (*optional*) If you do not want an object to inherit certain definitions of the model, and you want to leave the definitions undefined, specify a single plus sign (+) in these fields.

This symbol is allowed in all non-numeric fields (except for Y or N fields).

- Step 5** Press PF3 to save the object.

Table 3-2 describes the fields for creating and using a model.

**Table 3-2 This is a Model only and Modeled from Fields**

Field Name and Description	Valid Values
<p><b>This is a Model only</b> Specifies that this definition is a Model that you can use for other objects, at runtime, to inherit their properties from.</p> <p>When you specify Y and create a Model, you are planning to use this Model's definition to create additional objects with all (or some) of the same properties as the Model. The objects based on a Model inherit all or some of their object definitions from the Model at runtime. TOM never starts a Model.</p>	<p>Y (yes) or N (no) Default: N (no) Required: No</p>
<p><b>Modeled from</b> Specify the name of the Model or object that this object should inherit some or all of its properties from.</p> <p>Enter a question mark (?) in this field to see a list of all the objects and Models. From that list, you can select the object or Model for this field.</p> <p>The properties inherited from the Model or object are resolved at runtime when TOM initiates.</p>	<p>Name of a Model or object Default: None Required: No</p>

## Defining an Object in SUSPEND Mode

You can define an object that TOM keeps track of but does not start, stop, or try to restart. You might want to create an object in SUSPEND mode when you do not want TOM to manage the object's state but you have other objects that have dependencies on an object in SUSPEND mode.

For example, you can create a dependency between TSO and VTAM where VTAM must be running before TSO can start. However, you do not want TOM to manage the state of VTAM. If you create the VTAM object in SUSPEND mode, the TSO object can still have valid dependencies on the VTAM object, and TOM does not change VTAM's state.

You can also define an object in SUSPEND mode when you want to take an object temporarily out of TOM's control. An object can be switched from ACTIVE to SUSPEND mode (and back again) from the Started Definition Task panel, with a line command on the Started task Overview panel, from an EXEC or a console command.

Figure 3-4 shows an example of the upper portion of the Started Task Definition panel with the **Control** field highlighted in bold.

**Figure 3-4 Control Field**

```

Started Task Definition
Command ==>
More: +
General information
Name: CICSPROD.SYSA
Description: CICS Model for BBSYSPLX Product STM
Last modified: 2003/07/17 15:31:21 BAORMB3
STC name: CICS0001 Step name: Type: N ( TRAN or NORM )
This is a Model only: ( Y or N ) Control: SUSPEND ( Active or Suspend )
Modeled from:
.
.

```

To define an object in SUSPEND mode:

In the **Control** field, enter **SUSPEND** or leave the field blank.

**Note:** The **Control** field does not apply when defining a model.

You can also define an object in ACTIVE mode by entering ACTIVE in the **Control** field. When an object is in ACTIVE mode, it means that TOM manages the starting, stopping and recovery of the object's status.

Table 3-3 describes the field for defining SUSPEND or ACTIVE mode.

**Table 3-3 Control Field**

Field Name and Description	Valid Values
<p><b>Control</b>                      Specify whether TOM controls the state of this object.                      You can specify:</p> <ul style="list-style-type: none"> <li>• ACTIVE - means that the state of the object is monitored and controlled by TOM. The ACTIVE mode is not the same as the status ACTIVE; the status ACTIVE means that the object is currently running on the system.</li> <li>• SUSPEND - means that the object can be active and running but TOM does not start, stop or recover the object.</li> </ul> <p>TOM monitors the object and its state; for example: TOM displays the object's state as STOPPED if object terminates.                      Therefore all objects using this object's state as a dependency are processed as defined, for example, started and stopped according to the state of the SUSPEND object.</p>	<p>ACTIVE or SUSPEND                      Default: For a Model, this field is not used.                      If you leave the field blank or enter blanks in this field, the value is SUSPEND.                      Required: No</p>

**Note:** For information about changing the status of a SUSPEND object back to ACTIVE, refer to “Defining Reinstatement of an Object” on page 3-58.

## Defining a Layer Object

The **Pattern** field of the Started Task Definition field is used when you are creating a Layer object whose definitions can be propagated to object definitions on other sysplexes. Figure 3-5 shows the **Pattern** field on the Started Task Definition panel:

**Figure 3-5**      **Pattern Field**

Started Task Definition

Command ==>

More:      +

General information

Name: CICSPROD.SYSA

Description: CICS Model for BBSYSPLEX                      Product STM

Last modified: 2003/07/17    15:31:21    BAORMB3

STC name: CICS0001 Step name:                      Type: N ( TRAN or NORM )

This is a Model only:      ( Y or N )      Control: SUSPEND ( Active or Suspend )

Modeled from:

**Pattern:**

.

.

For more information about Layer objects, refer to Chapter 8, “Managing Objects across Sysplexes with Layering.”

## Viewing Set Associations

Sets are collections of objects or other sets. You can collect objects together in a set, for example, because you have multiple DB2 address spaces on the system and you want to manage them as a single entity.

The **Set associations** field displays how many sets an object belongs to. The field is also a hyperlink to pop-up panels where you can browse the sets the object belongs to.

Figure 3-6 shows an example of the **Set associations** field on the Started Task Definition panel.

**Figure 3-6 Set associations Field**

```

                                     Started Task Definition
Command ==>

                                                                 More:  +
General information
Name: CICSPROD.SYSA
      Description: CICS Model for BBSYSPPLEX          Product STM
      Last modified: 2003/07/17  15:31:21  BAORMB3
STC name: CICS0001 Step name:                Type: N ( TRAN or NORM )
This is a Model only:      ( Y or N )      Control: SUSPEND ( Active or Suspend )
Modeled from:
Pattern:
.
      Set associations: 0
.
.

```

To browse set associations:

**Step 1** Place the cursor on the **Set associations** hyperlink and press **Enter**.

The Started tasks sets pop-up panel is displayed (Figure 3-7).

**Figure 3-7 First Started tasks sets Pop-Up Panel**

```

                Started tasks sets
                Row 0 to 0 of 0
Command ==>          Scroll ==> CSR

    Line commands: B-browse

LC Description                Members
                               Number
-----
>-----
***** End of Data *****

```

When the object has not yet been added to a set, the panel shows no set names. See Figure 3-8 for an example of an object that belongs to a set named CICS Objects Set and BackUp Objects Set.

**Figure 3-8 Second Started tasks sets Pop-Up Panel**

```

                Started tasks sets
                Row 0 to 0 of 0
Command ==>          Scroll ==> CSR

    Line commands: B-browse

LC Description                Members
                               Number
-----
>-----
__ BackUp Objects Set          1
__ CICS Objects Set            1
***** End of Data *****

```

**Step 2** Type **S** in the LC field next to the set name and press **Enter**.

The Members of Set pop-up panel is displayed (Figure 3-9):

**Figure 3-9 Members of Set Pop-Up Panel**

Members of Set		
Command ==>		Row 0 to 0 of 0 Scroll ==> CSR
Short NAME	Member Type	Member Name
BACKUP	OBJECT	CICSPROD.SYSA
***** End of Data *****		

All of the objects that are a part of this set are displayed.

**Step 3** Press END to exit.

**Step 4** Press END to exit again and return to the Started Task Definition panel.

Table 3-4 describes the **Set associations** field.

**Table 3-4 Set associations Field**

Field Name and Description	Valid Values
<p><b>Set associations</b></p> <p>Displays the number of sets this object is defined to. It is also a hyperlink to a pop-up panel listing all the set names that this object belongs to.</p> <p>The definition and limitations of a set are:</p> <ul style="list-style-type: none"> <li>• A set is a group of objects or other sets.</li> <li>• A set can contain a set within a set within another set, up to 16 "nested" sets.</li> <li>• A set cannot have the same name as an object.</li> <li>• A set can contain an unlimited number of objects.</li> <li>• The objects inside a set can have different states.</li> </ul>	<p>Use pop-up panels.</p> <p>Default: None</p> <p>Required: No</p>

For more information about creating and managing sets, refer to Chapter 4, “Creating and Managing Sets” on page 4-1.

## Defining Valid Systems for the Object to Start On

You can define an object to be managed on more than one system in case a system becomes unavailable or because it is a critical object and you want TOM to try to start it on any and every available system. The **Valid Systems** field on the Started Task Definition panel is where you can create a list of systems that TOM attempts to start and use to manage the object.

If you want TOM to attempt to start an object on all systems listed in the Valid Systems list, you must also specify **Y** (for Yes) on the **Try alternate system upon failure** field. This means that if an object cannot start on the first system in the Valid Systems list (perhaps because a Pre-start EXEC returns a non-zero return code or for another reason), TOM attempts to start it on the next system in the list and so on.

However, if TOM cannot start an object on a system because the system is not *active*, TOM attempts to bring up the object on the other systems in the Valid systems list regardless of the value you specify in the **Try alternate system upon failure** field. Refer to “Defining Alternate Systems” on page 3-49 for more information.

### How TOM Progresses through the Valid Systems List

TOM uses the list of systems sequentially, which means that TOM first attempts to start the object on the first system that is listed. If that system is not available, TOM attempts to start the object on the next system in the Valid System list and so on. For example, if you specify three systems in the following order, TOM attempts to start the object on system SYS\_J first, system SYS\_A second, and system SYS\_D last:

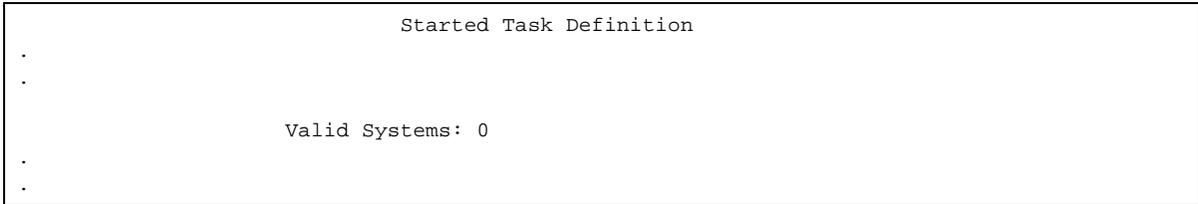
```
SYS_J  
SYS_A  
SYS_D
```

**Note:** If you do not enter at least one system in the **Valid Systems** field, TOM starts the object on the system where the object was originally defined.

The **Valid Systems** field is a hyperlink to a pop-up panel listing the names of all available systems and the current status of the object on that system. An object can have a status of ACTIVE on only one system at a time.

Figure 3-10 shows an example of the **Valid Systems** field on the Started Task Definition panel.

**Figure 3-10 Valid Systems Field**

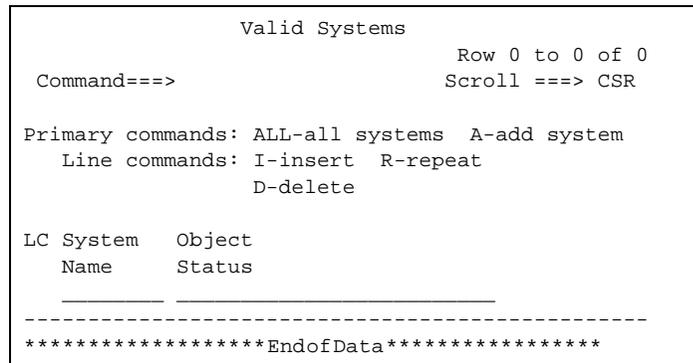


To enter system names for the Valid Systems pop-up panels on the Started Task Definition panel:

**Step 1** Place the cursor on the **Valid Systems** hyperlink and press **Enter**.

The Valid Systems pop-up panel is displayed (Figure 3-11).

**Figure 3-11 Valid Systems Pop-Up Panel**



**Step 2** On the **Command** line, enter **ALL** to display a list of all available systems.

**Step 3** Enter **S** in the **LC** field next to the system that you want to add.

**Step 4** Press **END** to save.

**Step 5** Press **END** to save again.

The **Valid Systems** field on the Started Task Definition panel reflects the update.

**Note:** The **Valid Systems** field works in conjunction with the **Try alternate system upon failure** and **Affinity** fields. For more information about **Try alternate system upon failure**, refer to “Defining Alternate Systems” on page 3-49. For more information about **Affinity**, refer to “Defining System Affinity” on page 3-51.

Table 3-5 describes the **Valid Systems** field.

**Table 3-5 Valid Systems Field**

Field Name and Description	Valid Values
<p><b>Valid Systems</b>                      This field displays the number of valid systems that TOM can start this object on. It is also a hyperlink to a pop-up panel listing all the valid systems this object can be started on and the current status of the object on that system.                      When more than one system is defined and the first system is not available, TOM tries to start the object on the next system listed.</p> <p>At least one system must be specified in the Valid systems field for the object to run. If you do not enter at least one system, TOM starts the object on the MVS system where the object was originally defined.</p> <p>This field works in conjunction with the fields <b>Try alternate system upon failure</b> and <b>Affinity</b>.                      For more information about the <b>Try alternate system upon failure</b> field, refer to "Defining Alternate Systems" on page 3-49.</p> <p>For more information about the <b>Affinity</b> field, refer to "Defining System Affinity" on page 3-51.</p>	<p>System name                      Default: New objects have a pre-populated entry of the system name where the TSO user ID is currently active                      Required: No</p>

## Defining a Schedule

By default, when no schedule is defined for an object, TOM attempts to start an object and maintain the object status as ACTIVE 24 hours a day, 7 days a week.

On the Started Task Definition panel, with the Schedules pop-up panels, you can specify dates and times that TOM should change the object status to STOPPED.

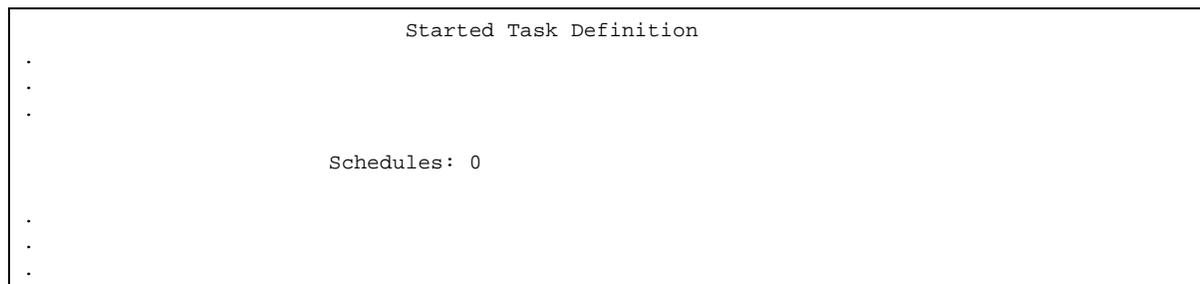
The Schedules pop-up panel also allows you to specify predefined periods of time, also referred to as calendar entries, which you create and are stored in a Calendar Base.

**Note:** BMC Software recommends that you become familiar with the information in Chapter 5, “Creating Calendar Bases” and “Entering Calendar Dependencies for Object Definitions” on page 5-22. This section does not fully explain all the calendar terminology.

You can also use the Schedules pop-up panels to specify the dates and times when TOM should stop an object without having to first create calendar entries.

Figure 3-12 shows an example of the **Schedules** field on the Started Task Definition panel.

**Figure 3-12 Schedules Field**



To enter values for the Schedules pop-up panels on the Started Task Definition panel:

**Step 1** Place the cursor on the **Schedules** hyperlink and press **Enter**.

The Schedule Calendar Dependencies pop-up panel is displayed (Figure 3-13).

**Figure 3-13 Schedule Calendar Dependencies Pop-Up Panel**

```

          SCHEDULE Calendar Dependencies
          Row 0 to 0 of 0
Command ==>                               Scroll ==> CSR

      Calendar base: DEFAULT
Primary Commands: ADD CMDSHOW
      Line commands: B-browse D-delete E-edit I-insert

      Include logic:  AND   (AND or OR)

LC T During Set          During Period      During Day
- -----
>-----
*****EndofData*****
    
```

**Step 2** On the **Command** line, type **ADD** and press **Enter**.

The Calendar Dependency panel is displayed (Figure 3-14).

**Figure 3-14 Calendar Dependency Pop-Up Panel**

```

          Calendar Dependency
COMMAND ==>
Type:      ( I -Include or E -Exclude )

During named
  Period:                (Enter ? to select from list)
   Set:                  (Enter ? to select from list)
   Day:                  (Enter ? to select from list)
   Time:                (Enter ? to select from list)

From named             To named
  Period:                Period:
   Day:                  Day:
   Time:                Time:

From specific
  Date:      /      /      Time:      :      ( yyyy/mm/dd hh:mm )
To specific
  Date:      /      /      Time:      :      ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

**Step 3** Complete the Calendar Dependency pop-up panel as follows:

**3.A** In the **Type** field, type **I** (for Include) or type **E** (for Exclude):

- Include means TOM starts the object during the time that is specified on this panel.

- Exclude means TOM stops the object during the specified time period.

**Note:** The Calendar Dependency pop-up panel has separate entry areas. You can enter calendar specifications for only one area per object each time that you display this pop-up panel. You can redisplay the panel to enter additional calendar specifications as needed.

**3.B** Choose one of the following areas in which to enter specifications:

- In the **During Named** fields, select a previously defined period of time.

An error message is displayed if you try to make more than one entry.

- In the **From named** and **To named** fields, select one previously defined period of time.

If you select one type of predefined period in **From named** (such as Day), you must select a matching predefined period in **To named**.

- In the **From specific** and **To Specific** fields, enter a date and time range.

This type of definition is specific to this object's definition and cannot be reselected for other objects.

**Note:** For information about how to use these fields and for examples, refer to Chapter 5, "Creating Calendar Bases" and also "Entering Calendar Dependencies for Object Definitions" on page 5-22 for information and examples about how to fill in this panel.

**Step 4** Press END to save.

On the Schedule Calendar Dependencies panel, the **T** column shows whether you chose I (Include) or E (Exclude).

The selected periods of time are also displayed.

**Note:** You might have to press PF11/Right and PF10/Left to see all of the columns.

**Step 5** Press END to save.

The **Schedules** field on the Started Task Definition panel reflects the updates.

**Note:** To specify additional schedules, repeat Step 1 through Step 5.

Table 3-6 describes the **Schedules** field.

**Table 3-6 Schedules Field**

Field Name and Description	Valid Values
<p><b>Schedules</b>                      This field displays the number of schedule definitions created for this object. It is also a hyperlink to the Schedule Calendar Dependencies panel for the object.                      If no schedule definitions are entered, the object has a schedule of being always ACTIVE, 24 hours a day, 7 days a week.</p>	<p>Use pop-up panels.                      Default: None                      Required: No</p>

## Defining How to Start an Object

This section describes the fields that define how TOM starts an object. The fields on the Started Task Definition field that are described in this section are

- **Lock next IPL**
- **Restart only**
- **Shutdown mode**
- **Reset start count after**
- **Reset start count at termination**
- **Start command type**
- **Start command**
- **Start time out and Max start count**

The values you specify for the fields listed below affect how TOM starts an object and you can find information about them in their respective sections:

- **Post-start EXECs** (refer to “Associating EXECs with an Object” on page 3-36)
- **Pre-start EXECs** (refer to “Associating EXECs with an Object” on page 3-36)
- **Start retry commands** (refer to “Defining Start or Stop Retry Commands” on page 3-32)
- **Startup validation** (refer to “Defining Events for Object Startup or Shutdown” on page 3-43)

Figure 3-15 shows the fields for defining how an object is started:

**Figure 3-15 Fields for Defining Object Startup**

```

Started Task Definition
Command ===>
More:      +
.
.
.
Lock next IPL:      ( Y or N )
Restart only:      ( Y or N )
Shutdown mode:      ( Normal or Quick )
Reset start count after:      Minutes
Reset start count at termination:      ( Y or N )
Pre-start EXECs: 0
Start command type: MVS      (Enter ? for the list)
Start command: S AAOCSMN4
Start time out: 5      Max start count: 20
Start retry commands: 0
Startup validation: 0
Post-start EXECs: 0
.
.
.

```

To enter definitions for starting an object:

**Step 1** In the **Lock next IPL** field, perform one of the following actions:

- Enter **Y** (for Yes).

TOM sets the object's status to **LOCKED** the next time that the system is IPLed. TOM does not try to start this object during subsequent IPLs.

- Enter **N** (for No) or leave this field blank if you want TOM to start (or stop) the object according to its definitions.

**Step 2** In the **Restart only** field, enter **Y** (for YES) if this object is managed by another scheduling product or is manually started outside of TOM but you want TOM to monitor its status.

**Note:** Although the Shutdown mode is strictly related to how TOM shuts down an object, the field is described in this section.

**Step 3** In the **Shutdown mode** field, enter **Q** (for Quick) if TOM is supposed to change the status of the object immediately to STOPPED when shutting down this object, without performing any automation.

Enter **N** (for Normal) if you want TOM to stop the object and while also performing any associated automation tasks.

**Step 4** In the **Reset start count after** field, enter the number of minutes that TOM waits after an object is started before TOM resets the number of issued Start commands to 0 (zero).

**Note:** Specifying 0 minutes instructs TOM not to reset the Start command count after an object is successfully started.

**Step 5** In the **Reset start count at termination** field, enter **Y** (for Yes) to specify that, after TOM stops the object, TOM resets the Start command count to 0 (zero).

**Step 6** Click the **Pre-Start EXECs** field to view EXEC information.

For more information about the **Pre-start EXECs** field, refer to “Associating EXECs with an Object” on page 3-36.

**Step 7** In the **Start command type** field, enter **MVS** or **EXEC** to specify that TOM use an MVS command or an EXEC to start the object.

**Step 8** In the **Start command** field, enter an MVS command or an EXEC name that TOM uses to start the object.

**Note:**

- Do not place a # symbol before an MVS command.
- If you do not specify a start command (or EXEC), TOM does not attempt to start the object.

**Step 9** In the **Start time out** field, enter the maximum number of minutes to wait for the object to become active after TOM issues the start command (or EXEC).

If the state of the object does not become ACTIVE within the specified interval, the start attempt ceases. If you specified additional commands in the Start retry commands pop-up panel, TOM issues these commands to try to start the object. For more information about Start retry commands, refer to “Defining Start or Stop Retry Commands” on page 3-32.

**Step 10** In the **Max start count** field, enter the number of times that TOM attempts to start the object after the object has ended abnormally.

For information about the **Start retry commands** field, refer to “Defining Start or Stop Retry Commands” on page 3-32.

For information about the **Startup validation** field, refer to “Defining Events for Object Startup or Shutdown” on page 3-43.

For information about the **Post-start EXECs** field, refer to “Associating EXECs with an Object” on page 3-36.

Table 3-7 describes the fields for defining object startup.

**Table 3-7 Defining an Object: Startup Fields**

Field Name and Description	Valid Values
<p><b>Lock at next IPL</b>                      Specifies that when the system is IPL'd, TOM sets this object's state to LOCKED. Regardless of its defined properties, for subsequent IPLs, TOM does not attempt to start this object.</p> <ul style="list-style-type: none"> <li>• When you specify Y (Yes), TOM sets the object's state to LOCKED when the system is IPL'd and TOM does not try to start the object during subsequent IPLs.</li> <li>• When you specify N (No) or leave this field blank, TOM starts and stops the object according to its property definitions.</li> </ul>	<p>Y (Yes) or N (No)                      Default: N (No)                      Required: No</p>
<p><b>Restart only</b>                      Specifies whether TOM automatically restarts this object. Use this field for objects that are managed by another scheduling product or are manually started outside of TOM but their availability should be monitored by TOM. When you specify Y (Yes), when the system is IPL'd, TOM does not start this object. In addition:</p> <ul style="list-style-type: none"> <li>• After the object starts TOM attempts to restart it if unexpectedly becomes inactive.</li> <li>• TOM performs any pre-start actions and issues the Start and Start retry commands.</li> <li>• TOM also checks the <b>Affinity</b> and <b>Valid Systems</b> fields and manages the object according to the Affinity level specified when more than one system is specified in the Valid Systems list.</li> </ul> <p>When you specify N (No), TOM starts, stops and restarts the object according to the specified schedule.</p>	<p>Y (Yes) or N (No)                      Default: N (No)                      Required: No</p>
<p><b>Shutdown mode</b>                      Specifies the shutdown mode for an object. When a quick shutdown mode is specified, TOM changes the status of the object immediately to STOPPED without performing any Pre-stop or Post-stop automation. You might set a quick shutdown for objects that are managed by another scheduling product or are manually started but their availability is monitored by TOM.</p>	<p>N (Normal) or Q (Quick)                      Default: N (Normal)                      Required: No</p>
<p><b>Reset start count after xxxx Minutes</b>                      Specifies how long TOM waits after an object is successfully started before TOM resets the number of issued Start commands.</p> <p>Specifying 0 minutes means that TOM does not reset the Start command count after object is started. TOM attempts to start the object the number of times specified in the Max Start Count field only when the object terminates with a failure.</p>	<p>0 - 1440 minutes                      Default: 0 minutes                      Required: No</p>

Table 3-7 Defining an Object: Startup Fields (continued)

Field Name and Description	Valid Values
<p><b>Reset start count at termination</b></p> <p>Specifies whether TOM resets the Start command count when the object is stopped under TOM's control.</p> <ul style="list-style-type: none"> <li>When you specify Y, TOM resets the Start command count every time TOM stops the object.</li> <li>When you specify N, TOM does not reset the Start command count, unless you specify a nonzero value in the <b>Reset start count after xxx min</b> field or you use the Reset line command from a panel or through an EXEC (using the API).</li> </ul>	<p>Y (Yes) or N (No)</p> <p>Default: N (No)</p> <p>Required: No</p>
<p><b>Pre-start EXECs</b></p> <p>Refer to "Associating EXECs with an Object" on page 3-36 for information about associating EXEC with an object.</p>	<p>Use pop-up panel</p> <p>Default: None</p> <p>Required: No</p>
<p><b>Start command type</b></p> <p>Specifies the type of command for the Start command for this object.</p> <p>When you specify a Start command to start this object, you must specify what type of command this is.</p> <p>You can enter a question mark (?) in this field to see a list of valid command types.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> <li>MVS: The command is issued as an MVS command</li> <li>EXEC: The command is scheduled as an EXEC on the AutoOPERATOR system connected to TOM</li> </ul>	<p>MVS or EXEC</p> <p>Default: MVS</p> <p>Required: No</p>
<p><b>Start command</b></p> <p>Specify a command or an EXEC to start the object.</p> <p><b>Note:</b> TOM checks if the object status is already ACTIVE before issuing the Start command or EXEC.</p> <p>If you use an EXEC to start the object and TOM cannot schedule the EXEC (perhaps because the EXEC does not exist), TOM sets the state of the object to FAILURE-CMD-INIT.</p> <p>In addition:</p> <ul style="list-style-type: none"> <li>When specifying a command, do not enter a pound sign (#) before the command.</li> <li>If you do not specify a start command (or EXEC), TOM does not start this object.</li> <li>You must enter a system for the object to run on the <b>Valid Systems</b> field. If you do not enter at least one system, the object does not start.</li> </ul>	<p>MVS command (up to 126 bytes) or EXEC name (up to 127 bytes)</p> <p>Default: None</p> <p>Required: No</p>

**Table 3-7 Defining an Object: Startup Fields (continued)**

Field Name and Description	Valid Values
<p><b>Start time out</b></p> <p>Specifies the maximum number of minutes to wait for the object status to become ACTIVE after TOM issues the start command or EXEC.</p> <p>TOM identifies that an object becomes ACTIVE in one of two ways:</p> <ul style="list-style-type: none"> <li>• When a Startup validation message ID is defined, TOM waits for the message ID to confirm that the object has started.</li> <li>• When a startup validation is not defined, TOM considers the object status ACTIVE when it discovers the object's address space exists.</li> </ul> <p>If the object does not become ACTIVE within the specified interval, the start attempt expires and TOM does not try to start the object.</p> <p>When the start attempt expires, if you have specified additional commands in the Start retry commands panel, TOM issues these commands to try to start the object.</p> <p>If you did not specify any Start retry commands, TOM stops trying to start the object and sets the object's state to FAILURE-RECOVERY.</p>	<p>0 to 1440 minutes                      Default: 0                      Required: No</p>
<p><b>Max start count</b></p> <p>Specifies the number of times TOM issues a Start command for the object when the object has ended abnormally.</p> <p>TOM maintains a count of how many start attempts are issued for an object. TOM issues the start attempts as long as the command count is not greater than the number specified in this field.</p> <p>The value specified in the Max start count field does not apply when the object ends normally.</p> <p>After TOM has tried starting the object the number of times specified in this field, TOM does not issue additional Start commands and sets the object state to FAILURE-CMDCOUNT.</p>	<p>0 to 99,999                      Default: 1                      Required: No</p>

## Defining How to Stop the Object

This section describes the fields that define how TOM stops the object. The fields on the Started Task Definition field that are described in this section are as follows:

- **Stop command type**
- **Stop command**
- **Stop time out**

The values you specify for the fields listed below affect how TOM starts an object and you can find information about them in their respective sections:

- **Pre-stop EXECs** and **Post-stop EXECs** (refer to “Associating EXECs with an Object” on page 3-36)
- **Stop retry commands** (refer to “Defining Start or Stop Retry Commands” on page 3-32)
- **Shutdown validation** and **Abnormal termination validation** (refer to “Defining Events for Object Startup or Shutdown” on page 3-43)

Figure 3-16 shows the fields for defining how to stop an object:

**Figure 3-16** Fields for Defining How to Stop an Object

```

Started Task Definition
Command ==>
                                                    More:      +
.
.
.
                Pre-stop EXECs: 0
Stop command type: MVS          (Enter ? for the list)
  Stop command: P AAOCSMN1
Stop time out: 5
                Stop retry commands: 0
                Shutdown validation: 0
Abnormal termination validation: 0
                Post-stop EXECs: 0
.
.
.

```

To enter definitions for stopping an object:

**Step 1** In the **Stop command type** field, enter **MVS** or **EXEC** to specify that TOM use an MVS command or an EXEC to stop the object.

**Step 2** In the **Stop command** field, enter an MVS command or an EXEC name that TOM uses to stop the object.

**Note:**

- Do not place a # symbol before an MVS command.
- If you do not specify a stop command (or EXEC), TOM does not attempt to stop the object.

**Step 3** In the **Stop time out** field, enter the maximum number of minutes to wait for the object to be stopped after TOM issues the stop command (or EXEC).

If the state of the object does not become STOPPED within the specified interval, the stop attempt ceases. If you have specified additional commands in the Stop retry commands pop-up panel, TOM issues these commands to try to stop the object. For more information about Stop retry commands, refer to “Defining Start or Stop Retry Commands” on page 3-32.

For information about the **Stop retry commands** field, refer to “Defining Start or Stop Retry Commands” on page 3-32.

For information about the fields **Shutdown validation** and **Abnormal termination validation**, refer to “Defining Events for Object Startup or Shutdown” on page 3-43.

For information about the fields **Pre-stop EXEC** and **Post-stop EXECs**, refer to “Associating EXECs with an Object” on page 3-36.

Table 3-8 describes the fields for defining how to stop an object.

**Table 3-8 Defining an Object: Stop command type, Stop command, Stop time out, and Stop retry commands Fields**

Field Name and Description	Valid Values
<p><b>Stop command type</b>            Specifies the type of command for the Stop command for this object.            When you specify a Stop command to stop this object, you must specify what type of command this is.            You can enter a question mark (?) in this field to see a list of valid command types.            Possible values are:</p> <ul style="list-style-type: none"> <li>• MVS: The command is issued as an MVS command</li> <li>• EXEC: The command is scheduled as an EXEC on the AutoOPERATOR system connected to TOM</li> </ul>	<p>MVS or EXEC            Default: MVS            Required: No</p>
<p><b>Stop command</b>            Specify a command or an EXEC to stop the object.  <b>Note:</b> TOM checks if the object status is already STOPPED before issuing the Start command or EXEC.            If you use an EXEC to stop the object and TOM cannot schedule the EXEC (perhaps because the EXEC does not exist), TOM sets the state of the object to FAILURE-CMD-INIT.            In addition:</p> <ul style="list-style-type: none"> <li>• When specifying a command, do not enter a pound sign (#) before the command.</li> <li>• If you do not specify a stop command, TOM does not stop this object.</li> </ul>	<p>MVS command (up to 126 bytes) or EXEC name (up to 127 bytes)            Default: None            Required: No</p>
<p><b>Stop time out</b>            Specifies the maximum number of minutes to wait for the object state to become STOPPED after TOM issues the stop command or EXEC.            TOM identifies that the status of an object becomes STOPPED in one of two ways:</p> <ul style="list-style-type: none"> <li>• When a Shutdown validation message ID is defined, TOM waits for the message ID to confirm that the object has started.</li> <li>• When a Shutdown validation event is not defined, TOM considers the object status STOPPED when it discovers the object's address space no longer exists.</li> </ul> <p>If the object does not become STOPPED within the specified interval, the Stop command expires and TOM does not try to stop the object.            When the stop attempt expires, if you have specified additional commands in the Stop retry commands field, TOM issues these commands to try to stop the object.            If you did not specify any Stop retry commands, TOM stops trying to stop the object and sets the object's state to FAILURE-RECOVERY.</p>	<p>0 to 1440 minutes            Default: 0            Required: No</p>

## Defining Start or Stop Retry Commands

This section describes the **Start retry commands** and **Stop retry commands** fields.

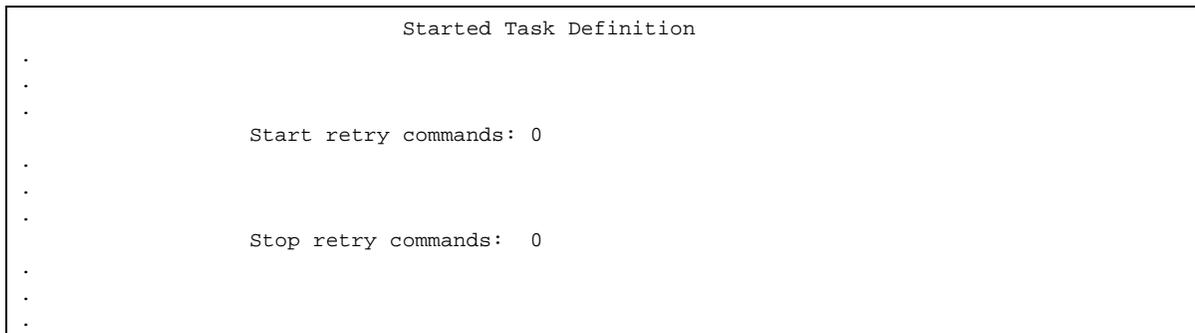
TOM might not be able start or stop an object under certain circumstances, such as the following situations:

- The command (or EXEC) that is specified in the **Start command** or **Stop command** fields fails to start or stop the object.
- The amount of time that is set in the **Start time out** field or **Stop time out** field expires before the object status becomes ACTIVE or STOPPED.

To handle these kinds of situations, you can specify additional start or stop commands and EXECs for TOM to use to attempt to start or stop an object.

Figure 3-17 shows an example of the **Start retry commands** field and the **Stop retry commands field** on the Started Task Definition panel.

**Figure 3-17** Start retry commands and Stop retry commands Fields



To enter additional start commands or EXEC names for TOM to use for objects:

**Note:** Because the procedures for specifying additional stop commands or EXECs are very similar to specifying additional start commands, use the following procedure for defining stop commands and EXECs for the **Stop retry commands** field.

**Step 1** Place the cursor on the **Start retry commands** hyperlink and press **Enter**.

The **Start retry commands** pop-up panel is displayed (Figure 3-18).

**Figure 3-18 Start retry commands Pop-Up Panel—First Panel**

```

                                Start retry commands
                                Row 0 to 0 of 0
Command ==>                                Scroll ==> CSR

Primary commands: ADD  CMDSHOW
Line commands:  B-browse  D-delete  E-edit
                I-insert  R-repeat

LC Description                                Command  Time Max
                                                Type    out  Count
-----
>-----
***** End of Data *****

```

**Step 2** On the **Command** line, enter **ADD** to specify additional MVS start commands (or EXEC names) for TOM to use to start the object.

A second Start retry commands pop-up panel is displayed (Figure 3-19).

**Figure 3-19 Start retry commands Pop-Up Panel—Second Panel**

```

                                Start retry commands
Command ==>

Command type:           Time out:           Max count:

Command:

Description:

Calendar dependencies:           0

Press END to save or CANCEL to exit

```

**Step 3** In the **Command type** field, enter **MVS** (for an MVS start command) or enter **EXEC**.

**Step 4** (*optional*) In the **Time out** field, enter a value for number of minutes.

If the start command or EXEC exceeds this value, the start attempt fails.

**Step 5** (*optional*) In the **Max count** field, enter a number of times that TOM attempts to start the object with the command or EXEC.

**Step 6** In the **Command type** field, enter **MVS** (or the EXEC name that TOM will use to start the object).

**Step 7** *(optional)* Place the cursor on the **Calendar dependencies** hyperlink and press **Enter**.

The Calendar Dependencies panel is displayed. On this panel, you can enter a specific time or other named periods of time. When calendar dependencies are defined, TOM attempts to schedule the EXECs based on your calendar definitions.

For more information about how to add calendar dependencies, refer to Chapter 5, “Creating Calendar Bases” and “Entering Calendar Dependencies for Object Definitions” on page 5-22.

**Step 8** Press END to save.

**Step 9** To enter more start commands or EXEC names, repeat Step 2 through Step 8.

**Step 10** Press END to save again.

The **Start retry commands** field on the Started Task Definition panel displays the number of additional defined start retry commands (or EXECs).

Table 3-9 describes the **Start retry commands** and **Stop retry commands** fields.

**Table 3-9 Defining an Object: Start retry and Stop retry commands Fields**

Field Name and Description	Valid Values
<p><b>Start retry commands</b></p> <p>This field is a hyperlink to a display listing the retry commands (or EXECs) TOM uses to start the object if:</p> <ul style="list-style-type: none"> <li>• The initial Start command (or EXEC) fails to start the object</li> <li>• If the amount of time set in the <b>Start time out</b> field ends before the state of the object becomes ACTIVE</li> </ul> <p>It also displays the number of Start retry commands currently defined for the object. To specify additional Start commands (or EXECs) if the initial start command does not work, use the pop-up panels to specify additional commands (or EXECs) to start the object.</p> <p>On the pop-up panel, you can specify that each command (or EXEC) can run multiple times using the <b>Time Out</b> and <b>Max count</b> fields. TOM issues the command (or EXEC) using the values set in these fields before proceeding the next Start command (or EXEC). When commands are specified, the commands run synchronously.</p> <p>If after using the Start retry commands TOM cannot start the object on this system, TOM can attempt to start the object on other systems when al of the following situations exist:</p> <ul style="list-style-type: none"> <li>• All the specified Start retry commands fail</li> <li>• More than one system is listed in the Valid systems field</li> <li>• You specify YES in the Try alternate system upon failure field</li> </ul> <p>When all these conditions are true, TOM tries to start the object on all systems listed in the Valid systems field.</p> <p>If the final retry is attempted on the last system listed in the Valid systems field and the object still does not become active, TOM sets the state of the object to FAILURE-RECOVERY and an exception is noted on the Started Task Overview panel. For more information about the <b>Try alternate system upon failure</b> field, refer to "Defining Alternate Systems" on page 3-49.</p>	<p>See pop-up panel Default: None Required: No</p>
<p><b>Stop retry commands</b></p> <p>This field is a hyperlink to a display listing the retry commands (or EXECs) TOM uses to stop the object if:</p> <ul style="list-style-type: none"> <li>• The initial Stop command (or EXEC) fails to stop the object</li> <li>• If the amount of time set in the <b>Stop time out</b> field ends before the state of the object becomes STOPPED</li> </ul> <p>It also displays the number of Stop retry commands currently defined for the object. To specify additional Stop commands (or EXECs) if the initial Stop command does not work, use the pop-up panels to specify additional commands (or EXECs) to start the object.</p> <p>On the pop-up panel, you can specify that each command (or EXEC) can run multiple times using the <b>Time Out</b> and <b>Max count</b> fields. TOM issues the command (or EXEC) using the values set in these fields before proceeding the next Stop command (or EXEC). When commands are specified, the commands run synchronously.</p> <p>If the final retry is attempted and the object still does not become STOPPED, TOM sets the state of the object to FAILURE-RECOVERY.</p>	<p>See pop-up panel Default: None Required: No</p>

## Associating EXECs with an Object

You can associate an EXEC to be scheduled for any of the following situations:

- before TOM starts an object—specify a Pre-start EXEC
- after TOM starts an object—specify a Post-start EXEC
- before TOM stops an object—specify a Pre-stop EXEC
- after TOM stops an object—specify a Post-stop EXEC

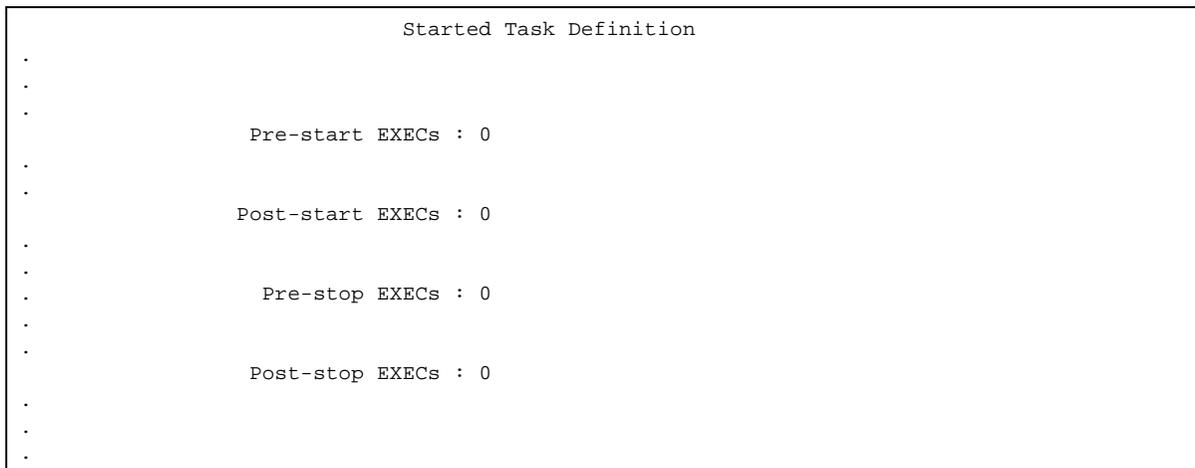
Any of these EXECs can be associated with and scheduled for an object. The procedure to associate an EXEC with an object startup or shutdown processing is essentially the same.

When EXECs are specified as part of an object’s definition, and you change the status of the object, you have to specify whether the associated EXECs are also scheduled.

For example, if you issue the Stop line command on the Started Task Overview panel, a pop-up panel is displayed, containing the option to stop the object with full automation. When you specify Yes, any Pre-stop and Post-stop EXECs that are associated with the object are scheduled as part of stopping the object.

Figure 3-20 shows an example of the EXECs hyperlinks on the Started Task Definition panel.

**Figure 3-20 Pre-start EXECs, Post-start EXECs, Pre-stop EXECs, and Post-stop EXECs Fields**



To specify a Pre-start EXEC with an object on the Started Task Definition panel:

**Step 1** Place the cursor on the **Pre-start EXECs** hyperlink and press **Enter**.

The Pre-start EXECs pop-up panel is displayed (Figure 3-21).

**Figure 3-21 Pre-start EXECs Pop-Up Panel—First Panel**

```

                                Pre-start EXECs
                                Row 0 to 0 of 0
Command ===>                               Scroll ===> CSR

Primary commands: ADD  CMDSHOW
Line commands:  B-browse  D-delete  E-edit
                I-insert  R-repeat

LC Description
_____
>-----
***** End of Data *****
    
```

**Step 2** On the **Command** line, enter **ADD** to add the name of an EXEC that you want TOM to schedule prior to starting the object.

A second Pre-start EXECs pop-up panel is displayed (Figure 3-22).

**Figure 3-22 Pre-start EXECs Pop-Up Panel—Second Panel**

```

                                Pre-start EXECs
Command===>

EXEC:

Description:

Calendar dependencies:           0

Press END to save or CANCEL to exit
    
```

**Step 3** In the **EXEC** field, enter the name of the EXEC.

**Step 4** In the **Description** field, enter a brief description of the EXEC.

**Step 5** (*optional*) Place the cursor on the **Calendar dependencies** hyperlink and press **Enter**.

The Calendar Dependencies panel is displayed. On this panel, you can enter a specific time or other named periods of time. When calendar dependencies are defined, TOM attempts to schedule the EXECs based on your calendar definitions.

For more information about how to add calendar dependencies, refer to Chapter 5, “Creating Calendar Bases” and “Entering Calendar Dependencies for Object Definitions” on page 5-22.

**Step 6** Press END to save.

**Step 7** To add additional Pre-start EXECs, repeat Step 2 through Step 6.

**Step 8** Press END to save.

The **Pre-start EXECs** field on the Started Task Definition panel displays the number of defined Pre-start EXECs.

Table 3-10 describes the Pre-start, Post-start, Pre-stop, and Post-stop EXECs fields.

**Table 3-10 Defining an Object: Pre-start EXECs, Post-start EXECs, Pre-stop EXECs, and Post-stop EXECs Fields**

Field Name and Description	Valid Values
<p><b>Pre-start EXECs</b></p> <p>This field displays the number of Pre-start EXECs to run before the object starts. It is also a hyperlink to a pop-up panel listing all the Pre-start EXECs for this object. To specify Pre-start EXECs for this object, place the cursor on the hyperlink and press enter. Use the pop-up panels to enter the EXEC names.</p> <p>In addition:</p> <ul style="list-style-type: none"> <li>• Variables are supported</li> <li>• TOM waits for each EXEC to finish running and checks for a return code of 0 before scheduling the next EXEC</li> <li>• You can specify Calendar Dependencies to define specific times when the Pre-start EXECs are scheduled</li> </ul> <p><b>Pre-start variables</b></p> <p>The following LOCAL variable pool variables are available to a Pre-start EXEC:</p> <ul style="list-style-type: none"> <li>• TOM_OBJECT: Object name that the EXEC has been scheduled for</li> <li>• TOM_OBJTYPE: Type of the object (in this case, STM)</li> <li>• TOM_SYSNAME: Name of the MVS system TOM is running on.</li> <li>• TOM_REQUEST: Can be 'initialization', 'termination' or 'poststr'</li> </ul> <p><b>Return Codes</b></p> <p>The Pre-start EXEC sets the return code with the IMFEXEC EXIT CODE() statement. The default return code is zero.</p> <p>When TOM receives a zero return code from a Pre-start EXEC, TOM continues to initiate the next Pre-start EXEC until each Pre-start EXEC is scheduled.</p> <p>If TOM detects a non-zero return code for any of the EXECs, the Pre-start EXECs stop and TOM does not start the object. TOM issues a message indicating the name of the EXEC that returned the non-zero return code and sets the state of the object as FAILURE-PRESTART.</p> <p>When an object has a FAILURE-PRESTART state, the object can be returned to TOM's control by issuing the Reset line command from the Started Task Overview panel.</p> <p><b>Calendar Dependencies</b></p> <p>You can specify Calendar Dependencies to define when each Pre-start EXEC should be scheduled. If Calendar Dependencies are not defined, the EXEC is scheduled every time before TOM starts the object. When Calendar Dependencies are defined, TOM checks the Calendar Definitions before scheduling the EXEC.</p> <p>If the Calendar Definitions indicate that the EXEC should not be scheduled, TOM continues to the next Pre-start EXEC, or with the Start command if there are no more Pre-start EXECs defined.</p> <p>Also note:</p> <ul style="list-style-type: none"> <li>• If you specify an EXEC that does not exist, the start attempt stops.</li> <li>• Each Pre-start EXEC can run only once.</li> <li>• You cannot specify a command to start an object in the Pre-start EXEC.</li> </ul>	<p>See pop-up panel Default: None Required: No</p>

**Table 3-10 Defining an Object: Pre-start EXECs, Post-start EXECs, Pre-stop EXECs, and Post-stop EXECs Fields**

Field Name and Description	Valid Values
<p><b>Post-start EXECs</b></p> <p>This field displays the number of Post-start EXECs scheduled to run after the object starts. It is also a hyperlink to a pop-up panel listing all the Post-start EXECs for this object.</p> <p>To specify Post-start EXECs for this object, place the cursor on the hyperlink and press Enter. Use the pop-up panels to enter the EXEC names.</p> <p>In addition:</p> <ul style="list-style-type: none"> <li>• Variables are supported</li> <li>• EXECs are scheduled to run asynchronously and TOM does not wait to process return codes</li> <li>• You can specify Calendar Dependencies to define specific times when the Post-start EXECs are scheduled</li> </ul> <p><b>Post-start variables</b></p> <p>The following LOCAL variable pool variables are available to a Post-start EXEC:</p> <ul style="list-style-type: none"> <li>• TOM_OBJECT: Object name that the EXEC has been scheduled for</li> <li>• TOM_OBJTYPE: Type of the object (in this case, STM)</li> <li>• TOM_SYSNAME: Name of the MVS system TOM is running on.</li> <li>• TOM_REQUEST: Can be 'initialization', 'termination' or 'poststrt'</li> </ul> <p><b>Calendar Dependencies</b></p> <p>You can specify Calendar Dependencies to define when each Pre-start EXEC should be scheduled. If Calendar Dependencies are not defined, the EXEC is scheduled every time before TOM starts the object. When Calendar Dependencies are defined, TOM checks the Calendar Definitions before scheduling the EXEC.</p> <p>If the Calendar Definitions indicate that an EXEC should not be scheduled, TOM continues to the next Post-start EXEC, and counts it against the Max count limit.</p> <p>Also note:</p> <ul style="list-style-type: none"> <li>• You can specify that each Post-start EXEC can run multiple times using the <b>Time Out</b> and <b>Max count</b> fields. TOM schedules the EXECs using the values set in these fields before proceeding the next Post-start EXEC.</li> <li>• If you specify a Post-start EXEC that does not exist or causes an error condition, TOM ignores the error and continues scheduling this EXEC until the total number of times exceeds the value specified in the <b>Max count</b> field.</li> </ul>	<p>See pop-up panel</p> <p>Default: None</p> <p>Required: No</p>

**Table 3-10 Defining an Object: Pre-start EXECs, Post-start EXECs, Pre-stop EXECs, and Post-stop EXECs Fields**

Field Name and Description	Valid Values
<p><b>Pre-stop EXECs</b></p> <p>This field displays the number of Pre-stop EXECs scheduled to run before the object stops. It is also a hyperlink to a pop-up panel listing all the Pre-stop EXECs for this object. To specify Pre-stop EXECs for this object, place the cursor on the hyperlink and press Enter. Use the pop-up panels to enter the EXEC names.</p> <p>In addition:</p> <ul style="list-style-type: none"> <li>• Variables are supported</li> <li>• TOM waits for each EXEC to finish running and checks for a return code of zero before scheduling the next EXEC</li> <li>• You can specify Calendar Dependencies to control when the Pre-stop EXECs are scheduled</li> </ul> <p><b>Pre-stop variables</b></p> <p>The following LOCAL variable pool variables are available to a Pre-stop EXEC:</p> <ul style="list-style-type: none"> <li>• TOM_OBJECT: Object name that the EXEC has been scheduled for</li> <li>• TOM_OBJTYPE: Type of the object (in this case, STM)</li> <li>• TOM_SYSNAME: Name of the MVS system TOM is running on.</li> <li>• TOM_REQUEST: Can be 'initialization', 'termination' or 'poststr'</li> </ul> <p><b>Return Codes</b></p> <p>The Pre-stop EXEC sets the return code with the IMFEXEC EXIT CODE() statement. The default return code is zero.</p> <p>When TOM receives a zero return code from a Pre-stop EXEC, TOM continues to initiate the next Pre-stop EXEC until each Pre-stop EXEC has been scheduled.</p> <p>If TOM detects a non-zero return code for any of the EXECs, the Pre-stop EXECs stop and TOM does not stop the object. TOM issues a message indicating the name of the EXEC that returned the non-zero return code and sets the state of the object as FAILURE-PRESTOP.</p> <p>When an object has a FAILURE-PRESTOP state, the object can be returned to TOM's control by issuing the Reset line command from the Started Task Overview panel.</p> <p><b>Calendar Dependencies</b></p> <p>You can specify Calendar Dependencies to define when each Pre-stop EXEC should be scheduled. If Calendar Dependencies are not defined, the EXEC is scheduled every time before the object is started. When Calendar Dependencies are defined, TOM checks the definitions before scheduling the EXEC.</p> <p>If the Calendar definitions indicate that the EXEC should not be scheduled, TOM continues to the next Pre-stop EXEC, or with the Stop command if there are no more Pre-stop EXECs defined.</p> <p>Also note:</p> <ul style="list-style-type: none"> <li>• If you specify an EXEC that does not exist, the stop attempt stops.</li> <li>• Each Pre-stop EXEC can run only once.</li> <li>• You cannot specify a command to stop an object in the Pre-stop EXEC.</li> </ul>	

**Table 3-10 Defining an Object: Pre-start EXECs, Post-start EXECs, Pre-stop EXECs, and Post-stop EXECs Fields**

Field Name and Description	Valid Values
<p><b>Post-stop EXECs</b></p> <p>This field displays the number of Post-stop EXECs to run after the object stops. It is also a hyperlink to a pop-up panel listing all the Post-stop EXECs for this object.</p> <p>To specify Post-stop EXECs for this object, place the cursor on the hyperlink and press Enter. Use the pop-up panels to enter the EXEC names.</p> <p>In addition:</p> <ul style="list-style-type: none"> <li>• Variables are supported</li> <li>• EXECs are scheduled to run asynchronously and TOM does not wait for return codes</li> <li>• You can specify Calendar Dependencies to control when the Post-stop EXECs are scheduled</li> </ul> <p><b>Post-stop variables</b></p> <p>The following LOCAL variable pool variables are available to a Post-stop EXEC:</p> <ul style="list-style-type: none"> <li>• TOM_OBJECT: Object name that the EXEC has been scheduled for</li> <li>• TOM_OBJTYPE: Type of the object (in this case, STM)</li> <li>• TOM_SYSNAME: Name of the MVS system TOM is running on.</li> <li>• TOM_REQUEST: Can be 'initialization', 'termination' or 'poststr'</li> </ul> <p><b>Calendar Dependencies</b></p> <p>You can specify Calendar Dependencies to define when each Post-stop EXEC should be scheduled. If Calendar Dependencies are not defined, the EXEC is scheduled every time after the object is stopped. When Calendar Dependencies are defined, TOM checks the Calendar Definitions before scheduling the EXEC. If the Calendar Definitions indicate that the EXEC should not be scheduled, TOM continues to the next Post-stop EXEC, and counts it against the Max count limit.</p> <p>Also note:</p> <p>You can specify that each EXEC can run multiple times using the Time Out and Max count fields. TOM schedules the EXECs using the values set in these fields before proceeding the next Post-stop EXEC.</p> <p>If you specify a Post-stop EXEC that does not exist or causes an error condition, TOM ignores the error and continues scheduling this EXEC until the total number of times exceeded the max count</p>	

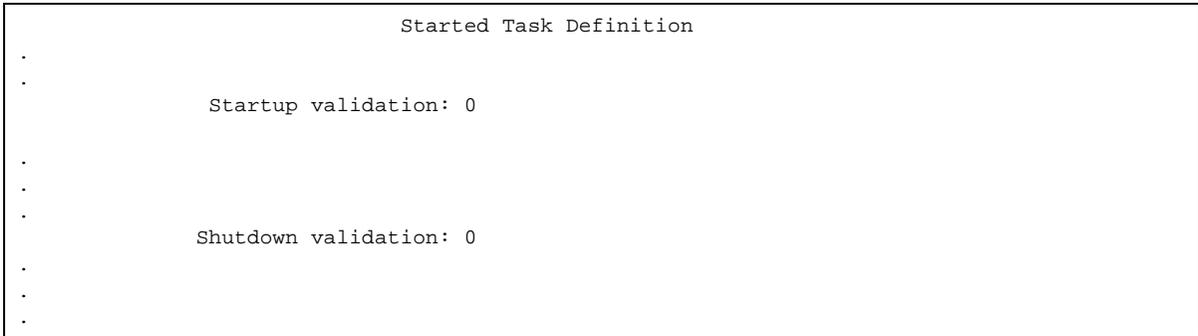
# Defining Events for Object Startup or Shutdown

On the Started Task Definition panel, you can use the **Startup validation** and **Shutdown validation** fields to specify a message ID and message text that signifies to TOM that an object has started or stopped successfully.

**Note:** BMC Software highly recommends that you specify startup and shutdown validation events for an object. When these events are not specified, TOM can detect the object and start or stop the object only when an STC name is defined for the object.

Figure 3-23 shows an example of the hyperlink fields that display pop-up panels where you can enter information about startup and shutdown events.

**Figure 3-23 Startup validation and Shutdown validation Fields**

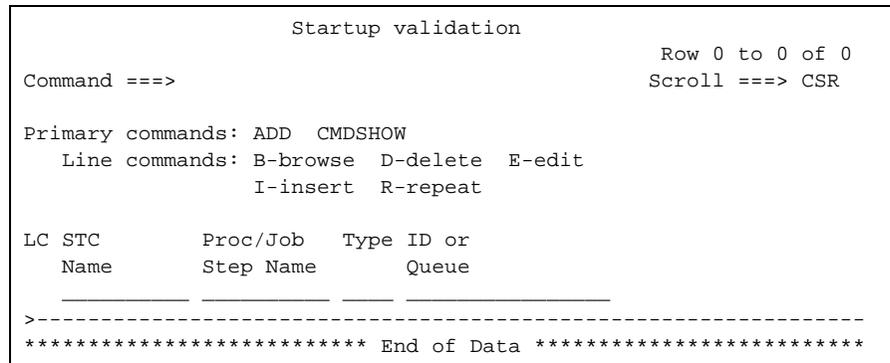


To specify a startup validation event. The process is very similar when specifying a shutdown validation event.

**Step 1** Place the cursor on the **Startup validation** hyperlink and press **Enter**.

The Startup validation pop-up panel is displayed (Figure 3-24).

**Figure 3-24 Startup validation Pop-Up Panel—First Panel**



**Step 2** On the **Command** line, type **ADD** and press **Enter**.

A second Startup validation pop-up panel is displayed (Figure 3-25).

**Figure 3-25 Startup validation Pop-Up Panel—Second Panel**

```

Startup validation
Command===>

Enter type or select and press Enter:
Type:      ( MSG CMD ALRT JRNL )

  _MSG_   _CMD_   _ALRT_   _JRNL_

Press END to save or CANCEL to exit
    
```

You can specify one of four event types that TOM uses to discover that an object has started successfully.

**Step 3** Place the cursor on the hyperlink name for your event type, and then press **Enter**.

In this example, Figure 3-26 shows the pop-up panel that is displayed when you select the MSG event type.

**Figure 3-26 Pop-Up Panel for Startup Message Event Type**

```

Message
Command===>

Message ID: IEF403I
Message:

Originating STC: AAOCSMN4   Proc/Jobstep:

Press END to save or CANCEL to exit
    
```

**Step 4** In the **Message ID** field, enter the ID of the validation message that signifies to TOM that the object has started.

**Step 5** In the **Message** field, enter text of the message that signifies to TOM that the object has started.

**Step 6** Press END to save.

**Step 7** Press END to save again.

The Startup validation pop-up panel shows the information for the startup validation event that you selected (Figure 3-27).

**Figure 3-27 Updated Startup validation Pop-Up Panel**

```

Startup validation
Row 0 to 0 of 0
Command ==> Scroll ==> CSR

Primary commands: ADD CMDSHOW
Line commands: B-browse D-delete E-edit
                I-insert R-repeat

LC STC      Proc/Job  Type ID or
Name       Step Name  Queue
-----
__ AAOCSMN4          MSG  IEF403I
>-----
    
```

Table 3-11 describes the **Startup validation** and **Shutdown validation** fields.

**Table 3-11 Startup validation and Shutdown validation Fields**

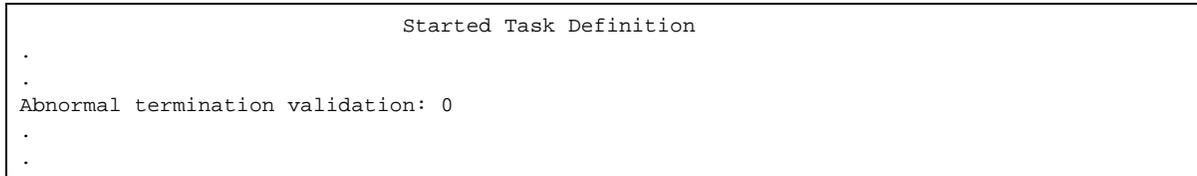
Field Name and Description	Valid Values
<p><b>Startup validation</b></p> <p>This field is a hyperlink to a display listing the events that notify TOM that the object has been successfully initialized. It displays the number of Startup validation events already specified for this object.</p> <p>To specify, place cursor on the field name and press Enter. Use the pop-up panels to enter the message ID and text you selected to inform TOM that the object is ACTIVE.</p> <p>You can specify an unlimited number of events. When any one of the events is detected, TOM sets the state of the object as ACTIVE.</p> <p>If you do not specify any Startup validation events, TOM recognizes and displays the object as ACTIVE when the address space exists and when the STC name of the object is defined. Therefore, if you want TOM to recognize this object as ACTIVE based on a specific message, you must use this field to specify the message ID (and any additional text).</p>	<p>Use pop-up panels. Default: None Required: No</p>
<p><b>Shutdown validation</b></p> <p>This field is a hyperlink to a display listing the events that notify TOM that the object has been successfully shut down or stopped. It displays the number of Shutdown validation events already specified for this object.</p> <p>To specify, place cursor on the field name and press Enter. Use the pop-up panels to enter the message ID and text you selected to inform TOM that the object is STOPPED.</p> <p>You can specify an unlimited number of events. When any one of the events is detected, TOM sets the state of the object as STOPPED.</p> <p>If you do not specify any Shutdown validation events, TOM recognizes and displays the object as STOPPED when the address space no longer exists. Therefore, if you want TOM to recognize this object as STOPPED based on a specific message, you must use this field to specify the message ID (and any additional text).</p>	<p>Use pop-up panels. Default: None Required: No</p>

# Defining Events for Abnormal Object Termination

On the Started Task Definition panel, you can use the **Abnormal termination validation** field to specify a message ID and text that signifies to TOM that an object has stopped unexpectedly or abnormally.

Figure 3-28 shows an example of the hyperlink field that displays pop-up panels where you can enter information about termination events.

**Figure 3-28 Abnormal termination validation Field**

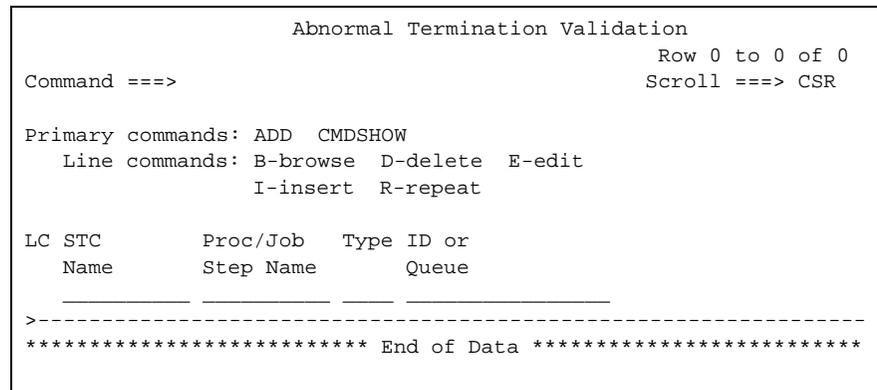


To add an abnormal termination event:

- Step 1** Place the cursor on the **Abnormal termination validation** hyperlink and press **Enter**.

The Abnormal Termination Validation pop-up panel is displayed (Figure 3-29).

**Figure 3-29 Abnormal Termination Validation Pop-Up Panel**



- Step 2** On the **Command** line, type **ADD** and press **Enter**.

The ABEND validation pop-up panel is displayed (Figure 3-30).

**Figure 3-30 ABEND validation Pop-Up Panel**

```

                                ABEND validation
Command===>

Enter type or select and press Enter:
Type:          ( MSG CMD ALRT JRNL )

  _MSG_   _CMD_   _ALRT_   _JRNL_

Press END to save or CANCEL to exit

```

You can specify one of four message types that TOM uses to detect that an object has terminated abnormally.

- Step 3** Place the cursor on the hyperlink name for the event type, and then press **Enter**.

In this example, Figure 3-31 shows the pop-up panel that is displayed when you select the MSG event type.

**Figure 3-31 Pop-Up Panel for Message Event Type**

```

                                Message
Command===>

Message ID: IEF450I
Message:

Originating STC: AAOCSMN4   Proc/Jobstep:

Press END to save or CANCEL to exit

```

- Step 4** In the **Message ID** field, enter the ID of the message that signifies to TOM that the object has terminated abnormally.
- Step 5** In the **Message** field, enter some text of the message TOM uses to identify the message which is issued when the object has terminated abnormally.
- Step 6** Press **END** to save the message.
- Step 7** Press **END** to return to the Started Task Definition panel.

The Abnormal Termination Validation pop-up panel shows the information for the abnormal termination event that you selected (Figure 3-32).

**Figure 3-32 Updated Abnormal Termination Validation Pop-Up Panel**

```

                                Abnormal Termination Validation
                                Row 0 to 0 of 0
Command ==>                                Scroll ==> CSR

Primary commands: ADD  CMDSHOW
Line commands:  B-browse  D-delete  E-edit
                I-insert  R-repeat

LC STC      Proc/Job  Type ID or
Name        Step Name Queue
-----
__ AAOCSMN4          MSG  IEF450I
>-----
    
```

Table 3-12 describes the **Abnormal termination validation** field.

**Table 3-12 Abnormal termination validation Field**

Field Name and Description	Valid Values
<p><b>Abnormal termination validation</b></p> <p>This field is a hyperlink to a display that lists the events that notify TOM that the object has abnormally ended. It displays the number of Abnormal termination events already specified for this object.</p> <p>To specify, place cursor on the field name and press Enter. Use the pop-up panels to enter the message ID and text you selected to inform TOM that the object has abnormally ended.</p> <p>You can specify an unlimited number of events. When any one of the events is detected, TOM sets the state of the object as FAILURE-ABEND.</p>	<p>Use pop-up panels.</p> <p>Default: None</p> <p>Required: No</p>

## Defining Alternate Systems

You can use the **Valid Systems** field to specify that an object can be started on more than one system. When you also specify **Yes** in the **Try Alternate system upon failure** field, TOM can attempt to start objects on the other systems that are listed in the Valid Systems list.

When more than one system is defined in the **Valid Systems** field, and the first system is not available or the object has failed to start on the first system, TOM attempts to start the object on the next system.

**Note:** This field is significant only when an object fails to start on a active system in the Valid Systems list. However, if TOM cannot start an object because the system is not available, TOM attempts to bring up the object on the other systems in the Valid Systems list regardless of the value you specify in this field.

Figure 3-33 shows an example of the **Try alternate system upon failure** field.

**Figure 3-33** Try alternate system upon failure Field

Started Task Definition
.
.
Try alternate system upon failure: ( Y or N )
.
.
.

To specify a value for the **Try Alternate system upon failure** field:

In the **Try alternate system upon failure** field, enter **Y** (for Yes).

TOM progresses sequentially through the Valid Systems list. For more information, refer to “Defining Valid Systems for the Object to Start On” on page 3-16.

**Note:** The value that you set for the **Affinity** field also has an effect on how TOM attempts to start objects from the Valid Systems list. Refer to “Defining System Affinity” on page 3-51 for more information.

If TOM has attempted to start the object on every listed system, and the object still does not become active, it is shown as an Exception on the Started Task Overview panel.

Table 3-13 describes the **Try alternate system upon failure** field on the Started Task Definition panel.

**Table 3-13 Try alternate system upon failure Field**

Field Name and Description	Valid Values
<p><b>Try alternate system upon failure</b></p> <p>Specifies whether TOM will try to start this object on another system because it has failed to start on the first system. When more than one system is defined in the Valid systems field, and the first system is not available or the object has failed to start on the first system, TOM attempts to start the object on the next system.</p> <p>This field is significant only when an object fails to start on the first system listed.</p> <p>If TOM has attempted to start the object on every system listed and the object still does not become active, it is shown as an Exception on the Started Task Overview panel.</p> <p>TOM sets the state of the object to the appropriate failure status on each system depending on the reason for the failure.</p> <p>If an object does not become active because the system is not available, TOM attempts to bring up the object on the other systems in the Valid systems list regardless of the value you specify in this field.</p>	<p>Y (Yes) or N (No)                      Default: N (No)                      Required: No</p>

---

## Defining System Affinity

When multiple systems are listed in the **Valid Systems** field, TOM can start objects on the other systems if

- the initial system is unavailable
- the other dependency definitions for the object are not satisfied on the initial system.

TOM progresses sequentially through the systems list, attempting to start the object on these systems.

The value that you specify in the **Affinity** field determines the priority TOM gives to each of the eligible systems in the **Valid Systems** field when attempting to start the object.

In other words, the Affinity value determines how *closely* TOM follows the Valid Systems list while attempting to start the object. The Affinity value defines that TOM should attempt to start the object on one system (or systems) over another.

Possible values for the **Affinity** field are

- High
- Medium
- Low

Affinity has an effect on how objects get started from a list of systems only when more than one system is listed in the **Valid Systems** field. Therefore, on the **Valid Systems** field, you must carefully list the systems in the order you want TOM to start the object.

For more information about the Valid Systems list, refer to “Defining Valid Systems for the Object to Start On” on page 3-16.

### High Affinity

When you set the **Affinity** field to High, TOM always tries to start the object on the highest system (the system at the top of the list), even if the object was successfully started on a system that appears lower on the list. Refer to “Example of High Affinity” on page 3-52.

### Example of High Affinity

For this example, the following systems have been listed in the **Valid Systems** field:

- SYSX
- SYSM
- SYSD

When the **Affinity** field is set to High, TOM always tries to start the object on SYSX, the highest system in the list, before attempting to start the object on systems SYSM or SYSD.

If the object cannot be started on SYSX, TOM tries to start the object on the next system that is listed, SYSM. If the first system, SYSX, becomes available while the object is active on SYSM, TOM *shuts down the object* on SYSM and tries to *start it again* on system SYSX.

If TOM cannot start the object on either system SYSX or SYSM, and the object started successfully on system SYSD, SYSM becomes available, TOM shuts down the object on SYSC and attempts to start it on SYSM.

### Med (Medium) Affinity

When you set the **Affinity** field to Med (Medium), TOM attempts to start the object on the highest-listed system in the Valid Systems list. However, if a higher-priority system becomes available, TOM *does not* stop the object and restart it on the higher-priority system.

Refer to “Example of Med Affinity”.

### Example of Med Affinity

For this example, the following systems have been listed in the **Valid Systems** field:

- SYSX
- SYSM
- SYSD

When the **Affinity** field is set to Med, TOM tries to start the object on SYSX, the highest system in the list, before attempting to start the object on systems SYSM or SYSD.

However, if the object is started on SYSM, the object continues to run on SYSM even if SYSX becomes available.

## Low Affinity

When the **Affinity** field is set to Low, the system that TOM picks to start the object is unpredictable.

**Note:** BMC Software strongly recommends that you select High or Med for the **Affinity** field when you have more than one system defined for an object in the Valid Systems list.

Figure 3-34 shows an example of the **Affinity** field.

**Figure 3-34** Affinity Field

```

Started Task Definition
.
.
.
Affinity:      ( High Low Med )
.
.
.

```

## Setting the Affinity

To specify a value for the **Affinity** field:

In the **Affinity** field, enter **High**, **Low**, or **Med**.

For example, High instructs TOM to always attempt to start the object on the highest listed system on the Valid Systems list, even if the object might be ACTIVE on a system listed lower on the list.

TOM progresses sequentially through the list of systems. For more information, refer to “Defining Valid Systems for the Object to Start On” on page 3-16.

**Note:** If TOM has attempted to start the object on every listed system, and the object still does not become active, it is shown as an Exception on the Started Task Overview panel.

Table 3-14 describes the **Affinity** field on the Started Task Definition panel.

**Table 3-14 Affinity Field**

Field Name and Description	Valid Values
<p><b>Affinity</b>                      This field is used in conjunction with the Valid Systems field.                      Determines the amount of priority TOM gives to the list of eligible systems listed for the object in the Valid Systems field.</p> <p>When multiple systems are specified in the Valid Systems field, TOM attempts to start objects on the systems in the order the systems are listed.                      When you specify a value in the Affinity field, you control how much priority TOM gives the list of systems when TOM attempts to start the object.</p> <p>Refer to “Defining System Affinity” on page 3-51 for examples and more information.</p>	<p>High, Med, or Low                      Default: Low                      Required: No</p>

## Defining Object Dependencies

One of the most important definitions of an object is its dependency on the state of another object. For example, CICS objects and TSO objects will likely have a dependency on NET (the VTAM address space) where NET must have a object status of ACTIVE before TOM attempts to make CICS and TSO objects active.

Objects can also have a dependency on a set of objects. You can also define that a set has a dependency on the state of an object or another set. You can use Dependency Property value pop-up panels to specify dependencies between objects and sets.

Figure 3-35 shows an example of the hyperlink field that displays pop-up panels where you can enter information about dependencies.

**Figure 3-35** Dependency Property value Field

```

Started Task Definition
.
.
.
Dependency Property value: 0
.
.
.

```

To add a simple dependency value for an object to another object:

**Step 1** Place the cursor on the **Dependency Property value** hyperlink and press **Enter**.

The Dependency pop-up panel is displayed (Figure 3-36).

**Figure 3-36** Defining an Object: Dependency Pop-Up Panel

```

Dependency
Command ==>                               Scroll ==> CSR

Name: CICSPROD2.SYSB
Property: @STATUS
On system: SYS_F                            (only valid for @STATUS)

Condition: EQ ( EQ NE LT GT LE GE = <> /= < > <= >= == != )
Property Value: (Enter ? below for @STATUS list)
ACTIVE

```

**Step 2** To make CICSPROD1.SYSA dependent on CICSPROD2.SYSB, enter **CICSPROD2.SYSB** in the **Name** field.

**Step 3** In the **Property** field, enter the word **@STATUS**.

@STATUS represents the state of the CICSPROD2.SYSB object while it is being managed by TOM.

**Step 4** In the **On system** field, enter the name of the system on which CICSPROD2.SYSB runs.

For example, CICSPROD2.SYSB might run on System B most of the time. But in some cases, such as during a system backup, CICSPROD2.SYSB might be running on System F (denoted as SYS\_F on the panel).

**Step 5** In the **Condition** field, enter the appropriate Boolean logic symbol or abbreviation.

For example, the abbreviation EQ means *equal to*.

**Step 6** In the **Property Value** field, enter the object status of CICSPROD2.SYSB.

**Step 7** Press END to save.

The **Dependency Property value** field shows a value of 1. CICSPROD1.SYSA has one dependency property.

Based on the definitions in this procedure, TOM will attempt to start CICSPROD1.SYSA only when CICSPROD2.SYSB has an @STATUS value of ACTIVE on System F.

Table 3-15 describes the **Dependency Property value** field and the fields in the Dependency pop-up panel.

**Table 3-15** Dependency Property value Fields

Field Name and Description	Valid Values
<p><b>Dependency Property value</b>                      This field is a hyperlink to the Dependency pop-up panel. It also displays the number of dependency properties already defined for this object.                      Use this panel to define a condition for this object to be ACTIVE. When the condition is not met, TOM does not start the object.                      To specify a dependency, place cursor on the field name and press Enter.</p>	<p>Use pop-up panels.                      Default: None                      Required: No</p>
<p><b>Name</b>                      Enter the name of the object (or set) on which this object depends. For a list of available objects and sets, place a question mark (?) in the <b>Name</b> field and press Enter.                      An object can be dependent on only one object or one set.                      To create an object that is dependent on multiple objects, those objects must be in a set.</p>	<p>N/A</p>
<p><b>Property</b>                      Specify @STATUS (or a user-defined property).                      The @STATUS value represents the state of the object while it is being managed by TOM.                      You can also create a user-defined property from an EXEC using a TOM API statement and define a dependency based on the property.</p>	<p>N/A</p>

**Table 3-15**      **Dependency Property value Fields**

<b>Field Name and Description</b>	<b>Valid Values</b>
<p><b>On system</b> Valid only when @STATUS is defined in the <b>Property</b> field. Specify the system name that TOM checks the property on.</p>	N/A
<p><b>Condition</b> Specify one of the comparison symbols shown on the panel. For example, NE means “not equal”.</p>	N/A
<p><b>Property Value</b> For a list of valid properties, place a question mark (?) below the Property Value field and press Enter. Possible values are ACTIVE, ACTIVE-BOUNCE, ACTIVE-ERR, ACTIVE-NOAUTO, ACTIVE-SD, BLOCKED, BLOCKED-AUTO, BLOCKED-ERR, COMPLETE, FAILURE-ABEND, FAILURE-AOLINK, FAILURE-CMD-INIT, FAILURE-CMD-TERM, FAILURE-CMDCOUNT, FAILURE-CONFIRM, FAILURE-PRESTART, FAILURE-PRESTOP, FAILURE-PROPERTY, FAILURE-REC-INIT, FAILURE-REC-TERM, LOCKED, LOCKED-ERR, PENDING, SCHEDULED, SCHEDULED-LOCKED, SCHEDULE-BLOCKED, STARTING, STOPPED, UNKNOWN, WAIT-START.</p>	N/A

For more information about TOM statuses, refer to “TOM Statuses” on page A-1.

## Defining Reinstatement of an Object

This section describes the following fields that are associated with how TOM removes an object that is defined with SUSPEND control (refer to “Defining an Object in SUSPEND Mode” on page 3-10):

- **Reinstatement**
- **Count**
- **Require confirmation**

You can use the **Reinstatement** field to specify how TOM removes an object when it is defined with SUSPEND control. For a suspended object, TOM keeps track of the state of the object but does not start, stop, or try to restart the object if it stops. You might want to create an object in SUSPEND mode when you do not want TOM to manage the object’s state but you have other objects that have dependencies on a suspended object’s state.

If you enter M (for Manual), the object remains in SUSPEND mode until one of the following conditions is met:

- The **Control** field is updated to ACTIVE (see “Defining an Object in SUSPEND Mode” on page 3-10).
- The Reset line command is issued against the object from the Started Task Overview panel.
- An EXEC is scheduled containing an API statement that resets the state of the object to ACTIVE.
- A console command is issued that resets the state of the object to ACTIVE.

**Note:** For information about TOM-related API statements and console commands, refer to Chapter 9, “Using AOAnywhere TOM Functions and Console Commands” on page 9-1.

If you enter I (for IPL), TOM checks the values in the **Count** and **Require confirmation** fields before starting the object at the next IPL.

Figure 3-37 shows an example of the **Reinstatement**, **Count**, and **Require confirmation** fields.

**Figure 3-37 Reinstatement, Count, and Require confirmation Fields**

```

Started Task Definition
.
.
Management
Reinstatement:      ( Manual or IPL )
Count and Confirmation fields are valid only for IPL:
Count:              Require confirmation:      ( Y or N )
.
.
.

```

To enter values for the **Reinstatement**, **Count**, and **Require confirmation** fields on the Started Task Definition panel:

**Step 1** In the **Reinstatement** field, perform one of the following actions:

- Specify **M** (for Manual) and proceed to Step 3.

The object remains in SUSPEND mode until the Control definition is manually updated to ACTIVE or a Reset line command is issued against the object from the Started Task Overview panel (or from the API or console commands).

- Specify **I** (for IPL) and proceed to Step 2.

**Step 2** If you specify I (for IPL), enter values in the **Count** and **Require confirmation** fields to start the object at the next IPL:

**2.A** In the **Count** field, enter the number of IPLs that TOM should wait before restarting the object.

For example, if you enter 5, TOM waits 5 IPLs before attempting to start the object. The default value for the **Count** field is 1, which means TOM restarts the object during the next IPL.

**2.B** In the **Require confirmation** field, specify **Y** (for Yes) or **N** (for No) to determine whether TOM issues a WTOR before it restarts the object.

**Note:** If you do not respond to the WTOR, TOM does not restart the object.

**Step 3** To save this definition, press **Enter**.

Table 3-16 describes the **Reinstatement**, **Count**, and **Require confirmation** fields.

**Table 3-16 Reinstatement, Count, and Require confirmation Fields**

Field Name and Description	Valid Values
<p><b>Reinstatement</b>                      Specifies how TOM removes an object from a SUSPEND control.                      When you specify Manual, the object remains SUSPENDED until the Control definition is updated to ACTIVE or a Reset line command is issued against the object from the Started Task Overview panel (or from the API or with console commands).                      When you specify IPL, TOM uses the Count and Require Confirmation fields to start the object at the next IPL(s).</p>	<p>Manual or IPL                      Default: Manual                      Required: No</p>
<p><b>Count</b>                      Specifies the number of IPLs after which TOM starts the object when it is in SUSPEND control and IPL is specified in the Reinstatement definition.                      The default is 1 which means TOM starts the object after the next IPL. If you specify 3, TOM starts the object after 3 IPLs.</p>	<p>1 - 99,999                      Default: 1                      Required: No</p>
<p><b>Require confirmation</b>                      Specifies whether TOM issues a WTOR after an IPL and waits for response to confirm starting an object when it is in SUSPEND control and IPL is specified in the Reinstatement field.                      When you specify Y, you will receive a WTOR after system IPL and you must respond before TOM can start the object.                      If you specify N, you will not receive a WTOR.</p>	<p>Y (Yes) or N (No)                      Default: N (No)                      Required: No</p>

## Verifying the Starting or Stopping of an Object

You might need to explicitly control the starting or stopping of an object (instead of letting TOM start and stop the object automatically according to its definitions and schedule). For these situations, you can use the **Verify Start** and **Verify Stop** fields.

When you specify Y (Yes) for either of these fields, TOM issues a write-to-operator-with-response (WTOR) message that requires the operator to issue a response before TOM starts or stops the object.

Figure 3-38 shows an example of the State control portion of the Started Task Definition panel.

**Figure 3-38** Verify Start and Verify Stop Fields

```

                                     Started Task Definition
Command ==>>
.
.
State control
Verify  Start: NO ( Y or N )
Verify  Stop:  NO ( Y or N )
.
.

```

To enter values for the **Verify Start** field on the Started Task Definition panel. The process is the same for entering values on the **Verify Stop** field.

**Step 1** In the **Verify Start** field, specify **Y** (for Yes).

**Note:** When Y (Yes) is specified, TOM issues a WTOR before starting the object and waits for a response to confirm starting the object. If there is no reply to the WTOR, TOM does not start the object.

**Step 2** To save this specification, press **Enter**.

Table 3-17 describes the fields for verifying the starting and stopping of an object.

**Table 3-17      Verify Start and Verify Stop Fields**

Field Name and Description	Valid Values
<p><b>Verify Start</b>                      Specifies whether TOM issues a WTOR each time before TOM starts the object.                      When you specify Y (Yes), TOM issues a WTOR before each start of the object and waits for a response to confirm starting the object. If there is no reply to the WTOR, TOM does not start the object.                      If you specify N (No), TOM starts the object without issuing a WTOR.</p>	<p>Y (Yes) or N (No)                      Default: N (No)                      Required: No</p>
<p><b>Verify Stop</b>                      Specifies whether TOM issues a WTOR each time before TOM stops the object.                      When you specify Y (Yes), TOM issues a WTOR before each stop of the object and waits for a response to confirm stopping the object. If there is no reply to the WTOR, TOM does not stop the object.                      If you specify N (No), TOM stops the object without issuing a WTOR.</p>	<p>Y (Yes) or N (No)                      Default: N (No)                      Required: No</p>

# Using Variables on the Started Task Definition Panel

The following sections describe where you can use variables in the Started Task Definition panel while defining object definitions.

## Where Variables Can Be Entered

You can use variables in the following areas for an object's definition:

- **Start command** field
- **Stop command** field
- For the Pre-start and Pre-stop EXEC name and parameters
- For the Post-start and Post-stop EXEC name and parameters
- In the following fields in the pop-up panel for the CMD, JRNL, or MSG events for the Startup, Shutdown, and Abnormal termination validation hyperlinks:
  - **Message ID** field
  - **Message text** field
  - **Originating STC** field
  - **PROC/JOB step** field

To see an example of the pop-up panel for MSG events, Figure 3-26 on page 44, or to Figure 3-31 on page 47.

Using variables in any of these fields can improve the flexibility of that definition object's definition. The variables entered in these fields are resolved at runtime in the TOM address space.

## How Variables Are Resolved

Variables can come from one of three sources:

- The SHARED variable pool of the MAINVIEW AutoOPERATOR BBI-SS PAS that the TOM address space is associated with

- The MVS symbol list from the logical partition (LPAR) that the TOM address space is running on
- BMC Software-supplied reference variables for the object. Refer to Table 3-18 on page 3-65.

When you decide to enter a variable as part of an object definition, you must prefix the variable name with an ampersand (&). Depending on the source of the variable, the ampersand might be followed by another character before the variable name begins. For example:

- a single ampersand (&) before the variable name indicates the variable will be resolved from values in the SHARED variable pool
- a double ampersand (&&) before the variable name indicates the variable will be resolved from the MVS symbol list
- a single ampersand followed by a single at sign (&@) before the variable name indicates the variable will be resolved from the list of the object's reference variables
- The delimiters for a variable are either a blank space or a backward slash (\). For example:

```
S &@STCNAME\,JOBNAME=ABCDXYZ (backward slash as  
delimitter)
```

Another example where a blank is a delimiter between &@STCNAME and FIRST:

```
S &@STEPNAME\,PARM=('&@STCNAME FIRST')
```

where STEPNAME has a value of AOKMZ32 and STCNAME has a value of AOKMZ01 and the command resolves to:

```
S AOKMZ32,PARM=('AOKMZ01 FIRST')
```

## Example of Resolving Variables from Different Sources

The following is an example of a variable from each source where

- the variable QIMFID has a value of KMZ2
- the variable SYSNAME has a value of SJSC

- the variable STCNAME has a value of AOKMZ01

**Table 3-18 Example of Resolving Variables**

Variable Source	Variable	Resolves to...
SHARED pool	HELLO &QIMFID	HELLO KMZ2
MVS symbol list	HELLO &&SYSNAME	HELLO SJSC
Reference variable list	HELLO &@STCNAME	HELLO AOKMZ01

## List of Reference Variables

In general, there is a reference variable for most of the object definitions that you can specify when defining an object for TOM to manage. To use a reference variable, specify the variable name in the appropriate field as part of the object's definition. At runtime, TOM resolves the value from the variable source and substitutes the variable name with the associated value.

The available reference variables, their values, and the fields on the Started Task Definition panel where you can use them are described in Table 3-19.

**Table 3-19 List of Reference Variables**

Reference Variable Name	Resolves to...
CONTROL	setting of either ACTIVE or SUSPEND Specified on the <b>Control</b> field.
DESCRIPTION	object description Specified on the <b>Description</b> field.
AFFINITY	affinity setting of High, Med, or Low Specified on the <b>Affinity</b> field.
TYPE	object type of either Norm (for normal objects) or TRAN (for transient objects) Specified on the <b>Type</b> field.
IPLSTART	setting of either Yes or No Specified on the Lock next IPL field.
RESTART	setting of either Yes or No Specified on the <b>Restart only</b> field.
SHUTDOWN	setting of either Normal (for a normal shutdown where automation is also performed) or Quick (where TOM shuts down the object without performing any associated automation tasks) Specified on the <b>Shutdown Mode</b> field.
RESTIME	number of minutes to wait before resetting start count Specified on the <b>Reset start count after xxxx Minutes</b> field.

**Table 3-19 List of Reference Variables**

<b>Reference Variable Name</b>	<b>Resolves to...</b>
MDLTYPE	setting of either Yes (this is a Model) or No (this is not a model) Specified on the <b>This is a model only</b> field.
PRODUCT	product name that this object is associated with (currently, STM) Specified on the <b>Product</b> field.
START#.CMD	start command Specified on the <b>Start command</b> field.
START#.TIMEOUT	Numeric start timeout value Specified on the <b>Start time out</b> field.
START#.TYPE	command type setting of either MVS (for an MVS command) or EXEC (for an EXEC) Specified on the <b>Start command type</b> field.
HOSTS#.SYSNAME	name of system in the object's eligible systems list Specified on the <b>Valid Systems</b> pop-up panels
STCNAME	name of a started task (STC) Specified on the <b>STC name</b> field.
STEPNAME	name of a step of a started task (STC) Specified on the <b>Step name</b> field.
STOPEVT#.ID	message ID from the stop event Specified on the CMD, JRNL or MSG events pop-up fields for the <b>Shutdown validation</b> field.
STOPEVT#.TYPE	event type (either CMD, JRNL, or MSG) from the stop event Selected from one of the <b>Shutdown validation</b> pop-up panels.
STOPEVT#.TEXT	message text from the stop event Specified on the CMD, JRNL, or MSG events pop-up fields for the <b>Shutdown validation</b> field.
STOPEVT#.ORIGINSTC	name of the originating started task from the stop event Specified on the CMD, JRNL or MSG events pop-up fields for the <b>Shutdown validation</b> field.
STOPEVT#.STEP	PROC or JOB step name for the stop event Specified on the CMD, JRNL or MSG events pop-up fields for the <b>Shutdown validation</b> field.
STARTEVT#.ID	message ID from the startup event Specified on the CMD, JRNL or MSG events pop-up fields for the <b>Startup validation</b> field.
STARTEVT#.TYPE	the event type (either CMD, JRNL, or MSG) from the startup event Selected from one of the <b>Startup validation</b> pop-up panels.

**Table 3-19 List of Reference Variables**

<b>Reference Variable Name</b>	<b>Resolves to...</b>
STARTEVT#.TEXT	message text from the startup event Specified on the CMD, JRNL, or MSG events pop-up fields for the <b>Startup validation</b> field.
STARTEVT#.ORIGINSTC	the name of the originating started task from the startup event Specified on the CMD, JRNL or MSG events pop-up fields for the <b>Startup validation</b> field.
STARTEVT#.STEP	PROC or JOB step name for the start event Specified on the CMD, JRNL or MSG events pop-up fields for the <b>Startup validation</b> field.
ABENDEVT#.ID	message ID from the abend event Specified on the CMD, JRNL or MSG events pop-up fields for the <b>Abnormal termination validation</b> field.
ABENDEVT#.TYPE	the event type (either CMD, JRNL, or MSG) from the abend event Selected from one of the <b>Abnormal termination validation</b> pop-up panels.
ABENDEVT#.TEXT	message text from the abend event Specified on the CMD, JRNL, or MSG events pop-up fields for the <b>Abnormal termination validation</b> field.
ABENDEVT#.ORIGINSTC	the name of the originating started task from the abend event Specified on the CMD, JRNL or MSG events pop-up fields for the <b>Abnormal termination validation</b> field.
ABENDEVT#.STEP	PROC or JOB step name for the abend event Specified on the CMD, JRNL or MSG events pop-up fields for the <b>Abnormal termination validation</b> field.



---

---

# Chapter 4    **Creating and Managing Sets**

This chapter describes how to create sets. The chapter includes the following topics:

Understanding Sets . . . . .	4-1
When to Use Sets . . . . .	4-2
Adding Objects and Sets to a New Set . . . . .	4-3
Managing Sets. . . . .	4-9

## **Understanding Sets**

Sets are collections of objects or other sets. For example, you might choose to group all of your CICS subsystems into a set. When you issue commands against the set, the commands apply to all of the CICS systems at once.

You can also group together unrelated objects. Additionally, you can specify that one set contain another set of objects. This “nesting” of sets can be done up to sixteen levels. For example, Set1 can contain Set2, which can contain Set3 (and so on) until Set16.

## When to Use Sets

When you group objects (or sets or both) into a set, you can

- manage the set from the TOM user interface

From the Started Task Sets application, you can issue line commands against all objects within the set simultaneously. Refer to “Managing Sets” on page 4-9 for more information.

- manage the set by using EXECs

You can schedule EXECs that use AOEXEC TOM commands to perform automation tasks against all objects within the set simultaneously.

- manage the set by using console commands

You can issue console commands against all objects within the set simultaneously.

- create dependencies where one object (or set) has a dependency on multiple objects that are grouped into a set

As described in “Defining Object Dependencies” on page 3-55, an object can have only one dependency definition specified. However, if you want an object to have dependencies on multiple objects, you can add those objects to a set. On the Dependency Property panels, enter the name of the set, thereby allowing one object to have dependencies on more than one object.

## Adding Objects and Sets to a New Set

The following procedure describes how to use the Started Task Sets application to add objects and sets to a new set:

**Note:** You can also add objects to a set from the Started Task Definition Panel. Refer to “Viewing Set Associations” on page 3-13 for more information.

**Step 1** On the TOM Primary Options Menu, select option 2, as shown in Figure 4-1.

**Figure 4-1 TOM Primary Options Menu**

```

BMC Software                TOM Primary Options Menu

Option ==> 2

1 Started Task  - Define, display and manage objects      User ID: BAOMXY1
2 Sets         - Define, display and manage sets          Server Id: TOMN1
3 Status       - Define, display and manage systems       Release: 1.1D
4 Calendar     - Define, display and manage calendars     System: SJSD

A Administration - System Administration
M Messages      - List messages
L Log           - Display Log

X Exit          - Exit

Copyright BMC Software, Inc. 2003

```

The Started Task Sets panel is displayed (Figure 4-2).

**Figure 4-2 Started Task Sets Panel**

```

Started Task Sets                Row 1 to 1 of 1
Command ==>                      Scroll ==> CSR

Primary commands: ADD  CMDSHOW  CUST  SORT

Line commands: B-browse  E-edit    D-delete  DD-delete block  R-repeat
                : S-start   P-stop    O-bounce  L-lock    UL-unlock  J-propagate
                : C-status  H-shutset HL-shutset/lock  U-suspend  A-activate
                : T-reset   M-move    K-block   UK-unblock

LC Description                Members Set
                               Type Number Status
-----
>-----
***** End of Data *****

```

**Step 2** On the **Command** line, enter **ADD**.

A pop-up panel is displayed (Figure 4-3).

**Figure 4-3 New Set Name Pop-Up Panel**

```

New set name
Command ==>

Enter the name for the new Set below:
>

Description:

Layer Pattern:

Press ENTER to continue or CANCEL to exit
    
```

**Step 3** At the arrow (>), enter the name of the new set.

Set names follow the same naming convention as object names. For more information about object naming conventions, refer to Table 3-1 on page 3-6.

**Note:** A set cannot have the same name as an object.

**Step 4** (optional) In the **Description** field, enter a description up to 36 characters.

**Step 5** (optional) For a Layer set, enter a 1- to 51-character pattern name in the **Layer Pattern** field.

**Note:** Refer to Chapter 8, “Managing Objects across Sysplexes with Layering,” for more information about Layer sets.

**Step 6** Press **Enter**.

The Members of Set pop-up panel is displayed (Figure 4-4).

**Figure 4-4 Members of Set Pop-Up Pane**

```

Members of Set
Row 0 to 0 of 0
Command ==> Scroll ==> CSR

Primary commands: AO-add object AS-add set
Line commands: D-delete

Short Member Member
LC NAME Type Name
-----
***** End of Data *****
    
```

**Step 7** Perform one the following actions:

- To add objects to the set, proceed to Step 8.
- To add a set to the set, proceed to Step 9.

**Step 8** To add objects to the set:

**8.A** On the **Command** line, enter **AO** (for add object).

An Objects panel is displayed (Figure 4-5).

**Figure 4-5 Objects Panel**

```

                                Objects
                                Row 1 to 2 of 2
Command ==>                               Scroll =

Primary commands: CMDSHOW  CUST  SORT
Line commands: S-select

LC Sel Object Name
-----
>-----
___ AutoOperator 123
___ AAOCSMN1
___ AAOCSMN2
___ AAOCSMN3
***** End of Data *****
    
```

**8.B** In the LC field, enter **S** next to each object that you want to add.

The word **Yes** appears in the **Sel** column, indicating that an object was added to the set (Figure 4-6).

**Figure 4-6 Adding an Object to a Set**

```

                                Objects
                                Row 1 to 2 of 2
Command ==>                               Scroll =

Primary commands: CMDSHOW  CUST  SORT
Line commands: S-select

LC Sel Object Name
-----
>-----
___ Yes AutoOperator 123
___ AAOCSMN1
___ AAOCSMN2
___ AAOCSMN3
***** End of Data *****
    
```

**8.C** Press **Enter** to save.

The Members of Set pop-up panel is redisplayed.

**Figure 4-7** Defining a Set: Members of Set Pop-Up Panel with Object Selected

```

Members of Set
Row 0 to 0 of 0
Command ==> Scroll ==> CSR

Primary commands: AO-add object AS-add set
Line commands: D-delete

Short Member Member
LC NAME Type Name
-----
__ OBJECT AutoOperator 123
***** End of Data *****
    
```

**8.D** (optional) In the **Short NAME** field, enter a 1- to 8-character name for the set.

**Figure 4-8** Defining a Set: Members of Set Pop-Up Panel with Object Selected

```

Members of Set
Row 0 to 0 of 0
Command ==> Scroll ==> CSR

Primary commands: AO-add object AS-add set
Line commands: D-delete

Short Member Member
LC NAME Type Name
-----
__ AO123SET OBJECT AutoOperator 123
***** End of Data *****
    
```

**8.E** Press END to save and return to the Started Task Sets panel

**Step 9** To add a set to the set:

**9.A** On the **Command** line, enter **AS** (for add set).

A list of defined sets is displayed (Figure 4-9).

**Figure 4-9** Sets Panel

```

Sets
Row 1 to 5 of 5
Command ==> Scroll ==> CSR

Primary commands: CMDSHOW CUST SORT
Line commands: S-select

LC Sel Set Name
-----
>-----
__ SYSPROG set
__ production set 2
__ CICS Object set 1
__ Layer Set for objects
***** End of Data *****

```

**9.B** In the **LC** field, enter **S** next to each set that you want to add.

The word **Yes** appears in the **Sel** column, indicating that a set was added to the set (Figure 4-10).

**Figure 4-10** Adding the Set to a Set

```

Sets
Row 1 to 5 of 5
Command ==> Scroll ==> CSR

Primary commands: CMDSHOW CUST SORT
Line commands: S-select

LC Sel Set Name
-----
>-----
__ SYSPROG set
__ Yes production set 2
__ CICS Object set 1
__ Layer Set for objects
***** End of Data *****

```

**9.C** Press **Enter** to save.

The Members of Set pop-up panel is redisplayed.

**Figure 4-11 Defining a Set: Members of Set Pop-Up Panel (2)**

```

Members of Set
Row 0 to 0 of 0
Command ==> Scroll ==> CSR

Primary commands: AO-add object AS-add set
Line commands: D-delete

Short Member Member
LC NAME Type Name
-----
__ SET
***** End of Data *****
    
```

**9.D** (optional) In the **Short NAME** field, enter a 1- to 8-character name for the set.

**Figure 4-12 Defining a Set: Members of Set Pop-Up Panel with Object Selected**

```

Members of Set
Row 0 to 0 of 0
Command ==> Scroll ==> CSR

Primary commands: AO-add object AS-add set
Line commands: D-delete

Short Member Member
LC NAME Type Name
-----
__ PRODSET2 SET
***** End of Data *****
    
```

**9.E** Press **END** to save and return to the Started Task Sets panel.

## Managing Sets

You can use the Started Task Sets application to manage sets by using line commands. Figure 4-13 shows an example of the Started Task Sets panel and the available line commands.

**Figure 4-13 Started Task Sets Panel**

```

Started Task Sets                                     Row 1 to 1 of 1
Command ==>                                         Scroll ==> CSR

Primary commands: ADD  CMDSHOW  CUST  SORT

Line commands: B-browse  E-edit    D-delete  DD-delete block  R-repeat
                  : S-start   P-stop    O-bounce  L-lock    UL-unlock J-propagate
                  : C-status  H-shutset HL-shutset/lock  U-suspend A-activate
                  : T-reset   M-move    K-block   UK-unblock

LC Description                                     Members  Set
                                                Type  Number  Status
-----
>-----
__ TEST SET OF OBJECTS                            NORM      16  STOPPED
__ SET OF BACKUP OBJECTS                          NORM      25  STOPPED
***** End of Data *****

```

This section describes how to use the following line commands:

- Start (starts all objects within the set)
- Stop (stops all objects within the set)
- Reset (resets all objects within the set to ACTIVE or STOPPED, based on each object's schedule definition)
- Suspend and Activate (change the state of the object between SUSPEND and ACTIVE)

**Note:** The Started Task Sets panel includes online Help for all of the line commands. To display Help for the line commands, place the cursor on the name of the line command on the panel (or in the LC column), and then press **PF1** (HELP).

## Starting a Set

When you issue the Start line command against a set, all the objects within the set are started, including objects in a set within the set. The following procedure describes how to start all objects within a set:

- Step 1** On the Started Task Sets panel, enter **S** in the **LC** field next to the name of the set.

**Figure 4-14 Starting Sets**

```

                Started Task Sets                                Row 1 to 1 of 1
Command ==>>                                         Scroll ==>> CSR

Primary commands: ADD  CMDSHOW  CUST  SORT

    Line commands: B-browse  E-edit    D-delete  DD-delete block  R-repeat
                  : S-start   P-stop    O-bounce  L-lock    UL-unlock  J-propagate
                  : C-status  H-shutset HL-shutset/lock  U-suspend  A-activate
                  : T-reset   M-move    K-block   UK-unblock

LC Description                                Members Set
                                           Type  Number  Status
-----
>-----
S_ TEST SET OF OBJECTS                        NORM      16  STOPPED
__ SET OF BACKUP OBJECTS                      NORM      25  STOPPED
***** End of Data *****
    
```

The confirmation pop-up panel is displayed (Figure 4-15).

**Figure 4-15 Confirm Start Pop-Up Panel**

```

Command ==>>

        Proceed with start?  N ( Y or N )

        Reset at next IPL?  Y ( Y or N )

        Start with full automation?  Y ( Y or N )

        Bypass Command Count Check?  N ( Y or N )

        System name (or ? for list)  ?

        Press ENTER to continue or END to cancel
    
```

- Step 2** For each question on the confirmation pop-up panel, enter the value that determines how TOM starts the objects within the set.

Refer to Table 4-1 on page 4-11 for information about the options for starting the objects within a set.

Table 4-1 describes the various options for starting an object.

**Table 4-1 Starting a Set: Options**

Option	Description
Proceed with start? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM issues the start command (or EXEC) defined for each object inside the set (and each object within sets of the set). When you use the Start line command, TOM ignores any schedule defined for the objects and on the Started Task Overview Panel, YES appears in the Forced column for each object within the set. The status of each object is LOCKED on all other systems defined for the object's in the Valid Systems list. TOM displays the set status as ACTIVE on the Started Task Sets panel until you issue a Stop or Reset line command or until the next IPL (refer to the Reset at next IPL option). No specifies the start command (or EXEC) defined for the objects within the set will not be issued.</p>
Reset at next IPL? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM takes the objects out of Forced status after the next IPL and TOM resumes control of the objects. The objects' schedule and other definitions become effective. No specifies that TOM continues to maintain the objects in Forced ACTIVE status after the next IPL. TOM does not use the objects' schedule or other definitions.</p>
Start with full automation? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM schedules the Pre-start EXECs before issuing the start command (or EXEC) against the objects in the set and the Post-start EXECs after the objects become ACTIVE. No specifies that TOM issues the Start command (or EXEC) for the objects without scheduling the Pre-start and Post-start EXECs.</p>
Bypass command count check? Y or N	<p>Possible values are Y (yes) or N (no)</p> <p>Yes specifies that TOM does not check to see if the start command count specified in the object's definition (on the <b>Maximum Start Count</b> field of the Started Task Definition panel) has been exceeded. When you specify yes, the start command count is still incremented when the start command is issued and the start command count is reset to zero when the object's definition specifies yes in the <b>Reset start count at termination</b> field of the Started Task Definition panel. No means that TOM checks the command count and will not start the objects if the <b>Maximum Start Count</b> is exceeded.</p>
Start on system	<p>This is a required field. You must specify a system name on which to start the objects when issuing the Start line command. You can enter a question mark (?) to display a list of each system and select the system on which to start the object within the set.</p>

## Stopping a Set

When you issue the Stop line command against a set, all objects within the set are stopped, including objects in a set within the set. The following procedure describes how to stop all objects within a set:

- Step 1** On the Started Task Sets panel, enter **P** (for Stop) in the **LC** field next to the name of the set.

**Figure 4-16 Started Task Sets: Stopping Sets**

```

Command ==>>
                                     Started Task Sets
                                     Row 1 to 1 of 1
                                     Scroll ==>> CSR

Primary commands: ADD  CMDSHOW  CUST  SORT

Line commands: B-browse  E-edit    D-delete  DD-delete block  R-repeat
                : S-start   P-stop    O-bounce  L-lock    UL-unlock  J-propagate
                : C-status  H-shutset HL-shutset/lock  U-suspend  A-activate
                : T-reset   M-move    K-block   UK-unblock

LC Description                                Members Set
                                           Type  Number  Status
-----
>-----
P_ TEST SET OF OBJECTS                       NORM    16 ACTIVE
__ SET OF BACKUP OBJECTS                     NORM    25 STOPPED
***** End of Data *****

```

The confirmation pop-up panel is displayed (Figure 4-17).

**Figure 4-17 Confirm Stop Pop-Up Panel**

```

Command ==>>

Proceed with stop?  N ( Y or N )

Reset at next IPL?  Y ( Y or N )

Stop with full automation?  Y ( Y or N )

Press ENTER to continue or END to cancel

```

- Step 2** For each question on the confirmation pop-up panel, enter the value that determines how TOM stops the objects within the set.

Refer to Table 4-2 on page 4-13 for information about the options for stopping the objects within a set.

Table 4-2 describes the various options for stopping an object.

**Table 4-2 Stopping a Set: Options**

Option	Description
Proceed with stop? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM issues the stop command (or EXEC) defined for each object inside the set (and each object within sets of the set). When you use the Stop line command, TOM ignores any schedule defined for the objects.</p> <p>TOM displays the objects within the set as LOCKED status until you issue a Start or Reset line command or until the next IPL (refer to the Reset at next IPL option).</p> <p>No specifies the stop command (or EXEC) defined for the objects within the set will not be issued.</p>
Reset at next IPL? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM takes the objects out of LOCKED status after the next IPL and TOM resumes control of the objects. The objects' schedule and other definitions become effective.</p> <p>No specifies that TOM continues to maintain the objects in LOCKED status after the next IPL. TOM does not use the objects' schedule or other definitions.</p>
Stop with full automation? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM schedules the Pre-stop EXECs before issuing the stop command (or EXEC) against the objects in the set and the Post-stop EXECs after the objects become STOPPED.</p> <p>No specifies that TOM issues the Stop command (or EXEC) for the objects without scheduling the Pre-stop and Post-stop EXECs.</p>

## Resetting a Set

When the Reset line command is issued against a set, all objects within the set—including objects in a set within the set—are reset to either ACTIVE or STOPPED status, based on each object’s definitions. The following procedure describes how to reset all objects within a set:

- Step 1** On the Started Task Sets panel, enter **T** (for Reset) in the **LC** field next to the name of the set.

**Figure 4-18** Resetting Sets

```

Started Task Sets                                     Row 1 to 1 of 1
Command ==>                                         Scroll ==> CSR

Primary commands: ADD  CMDSHOW  CUST  SORT

Line commands: B-browse  E-edit  D-delete  DD-delete block  R-repeat
                : S-start  P-stop  O-bounce  L-lock  UL-unlock  J-propagate
                : C-status  H-shutset  HL-shutset/lock  U-suspend  A-activate
                : T-reset  M-move  K-block  UK-unblock

LC Description                                     Members Set
                                                Type  Number  Status
-----
>-----
T_ TEST SET OF OBJECTS                             NORM    16 OTHER
__ SET OF BACKUP OBJECTS                           NORM    25 STOPPED
***** End of Data *****

```

The confirmation pop-up panel is displayed (Figure 4-19).

**Figure 4-19** Confirm Reset Pop-Up Panel

```

Command ==>

Reset all?  N ( Y or N )

Reset Forced?  N ( Y or N )

Reset Status?  N ( Y or N )

Reset Abend count?  N ( Y or N )

Reset SUSPEND control?  N ( Y or N )

Reset Start command count?  N ( Y or N )

Press ENTER to continue or END to cancel

```

**Step 2** For each question on the confirmation pop-up panel, enter the value that determines how TOM resets the objects within the set.

**Note:** The Reset line command can cause TOM to start or stop objects with full automation where all of the Pre-start, Post-start, Pre-stop, and Post-stop EXECs are scheduled. The Reset line command is also executed against all systems in the object's Valid Systems list.

Refer to Table 4-3 for information about the options for resetting the objects within a set.

**Table 4-3**      **Resetting a Set: Options**

Option	Description
Reset all? Y or N	Possible values are Y (yes) or N (no). Yes specifies that TOM resets the status of all the objects within the set to either ACTIVE or STOPPED, based on the current schedule definition for each of the objects. TOM resumes complete control of the objects' status and manages the objects according to its defined schedule and all other object definitions. When you specify Yes, all of the other options with this line command are ignored. No specifies that you can select other options.
Reset Forced? Y or N	Possible values are Y (yes) or N (no). Yes specifies that TOM resets the status of each object within the set, based on the defined schedule, to ACTIVE or STOPPED if the object was in Forced ACTIVE status. TOM resumes control of the object according to its schedule. No specifies that TOM does not reset the Forced ACTIVE status.
Reset Status? Y or N	Possible values are Y (yes) or N (no). Yes specifies that TOM resets the status of each object within the set, based on the defined schedule, to ACTIVE or STOPPED regardless of the current status of the object. For example this option can be used to clear LOCKED, BLOCKED or any FAILURE status. No specifies that TOM does not reset the status.
Reset Abend count? Y or N	Possible values are Y (yes) or N (no). Yes specifies that TOM resets the Abend count for each object within the set. The Abend count represents the number of times the Abnormal Termination Event occurred since the last time the count was reset. No specifies that TOM does not reset the Abend count.
Reset Suspend Control? Y or N	Possible values are Y (yes) or N (no). Yes specifies that the objects within the set return to TOM's control and SUSPEND state is changed to ACTIVE state. TOM controls the objects' availability according to their schedule. No specifies that TOM does not take the object out of SUSPEND state.
Reset Start command count? Y or N	Possible values are Y (yes) or N (no). Yes specifies that TOM resets the Start command count for each object. Use this option when an object has reached the Max start count limit. No specifies that TOM does not reset the Start command count.

## Suspending or Activating a Set

The Suspend and Activate line commands have the following effects:

- Suspend temporarily takes all objects within the set out of TOM's control. TOM continues to monitor the status of the objects but does not start or stop the objects based on their schedules or definitions.
- Activate reinstates all suspended objects within TOM's control. TOM resumes starting and stopping the objects based on their schedules and definitions.

### Suspending the Set

To take all objects within a set temporarily out of TOM's control:

- Step 1** On the Started Task Set panel, enter U (for Suspend) in the LC field next to the name of the set.

**Figure 4-20 Suspending Sets**

```

                Started Task Sets                                Row 1 to 1 of 1
Command ==>                                           Scroll ==> CSR

Primary commands: ADD  CMDSHOW  CUST  SORT

    Line commands: B-browse  E-edit    D-delete  DD-delete block  R-repeat
                  : S-start   P-stop    O-bounce  L-lock    UL-unlock J-propagate
                  : C-status  H-shutset HL-shutset/lock  U-suspend A-activate
                  : T-reset   M-move    K-block   UK-unblock

LC Description                                     Members Set
                                                Type  Number  Status
-----
>-----
U_ TEST SET OF OBJECTS                             NORM      16 ACTIVE
__ SET OF BACKUP OBJECTS                           NORM      25 STOPPED
***** End of Data *****
    
```

The confirmation pop-up panel is displayed (Figure 4-21).

**Figure 4-21 Confirm Suspend Pop-Up Panel**

```

Command ==>

        Proceed?  N ( Y or N )

Press ENTER to continue or END to cancel
    
```

- Step 2** In the **Proceed** field, enter **Y**.

## Activating the Set

To return control of the objects within a set to TOM, you can perform any of the following actions:

- Issue the Activate line command on the Started Task Set panel next to the set name.
- Issue the Reset line command on the Started Task Set panel next to the set name and specify Y (yes) for the Reset Suspend Control option on the Confirm pop-up panel.
- Edit the Control field of the object’s definition on the Start Task Definition panel and enter ACTIVE.

To return all objects within a set to TOM’s control:

- Step 1** On the Started Task Sets panel, enter A (for Activate) in the LC field next to the name of the set.

**Figure 4-22 Activating Sets**

```

Started Task Sets                               Row 1 to 1 of 1
Command ==>                                     Scroll ==> CSR

Primary commands: ADD  CMDSHOW  CUST  SORT

Line commands: B-browse  E-edit    D-delete  DD-delete block  R-repeat
                : S-start   P-stop   O-bounce  L-lock   UL-unlock J-propagate
                : C-status  H-shutset HL-shutset/lock  U-suspend A-activate
                : T-reset   M-move   K-block  UK-unblock

LC Description                                     Members  Set
                                           Type  Number  Status
-----
>-----
A_ TEST SET OF OBJECTS                            NORM      16 OTHER
__ SET OF BACKUP OBJECTS                          NORM      25 STOPPED
***** End of Data *****

```

The confirmation pop-up panel is displayed (Figure 4-23).

**Figure 4-23 Confirm Activate Pop-Up Panel**

```

Command ==>

Proceed?  N ( Y or N )

Press ENTER to continue or END to cancel

```

- Step 2** In the **Proceed** field, enter **Y**.



---

---

# Chapter 5 Creating Calendar Bases

This chapter describes how to create TOM Calendar Bases.

This chapter includes the following topics:

Overview of Calendar Bases . . . . .	5-1
Creating a New Calendar Base . . . . .	5-6
Adding a Day Entry . . . . .	5-8
Adding a Period Entry . . . . .	5-12
Adding a Time Entry . . . . .	5-15
Adding Items to a Calendar Base Set . . . . .	5-18
Entering Calendar Dependencies for Object Definitions . . . . .	5-22

## Overview of Calendar Bases

The Calendar Base is a collection of periods of time that you can use when creating time-dependent schedules for objects to be ACTIVE or STOPPED. By default, when no schedule is specified for an object, TOM attempts to maintain the object as always ACTIVE. However, objects might need to be inactive.

**Note:** For rare or infrequent situations, you can schedule individual periods of time (on the Started Task Definition panel) when TOM will stop an object and maintain its status as STOPPED.

For frequent or routine periods of time when TOM should stop objects, you can use the Calendar Base application to specify periods of time, assign a name for that period of time, and use that period of time for a multitude of objects.

The periods of time that you can specify in the Calendar Base application are as follows:

- Day
- Period
- Time
- Set

Refer to Table 5-1 on page 5-3 for more information about these terms.

## When to Use a Calendar Base

After you have analyzed the required availability schedule for the objects that TOM will manage and created Calendar Base entries to help you specify these schedules, you can select these periods of time while creating object definitions on the Started Task Definition panel for the following object properties:

- Schedules
- Pre-start EXECs
- Post-start EXECs
- Pre-stop EXECs
- Post-stop EXECs
- Start retry commands
- Stop retry commands

For each of these object properties, you can specify a Calendar Dependency on the pop-up panel where you specify the name of an EXEC that is associated with the starting or stopping of an object (or where you enter any additional object start or stop retry commands).

This Calendar Dependency means that the EXEC or retry command that you specify may be further defined to occur based on Day, Period, Time, or Set Calendar Base definitions or at a single specific date and time.

In addition, on the Started Task Definition panel, you can access the Schedules pop-up panel to specify when TOM should try to make an object active (or inactive).

For more information about entering Calendar Dependencies, refer to (“Entering Calendar Dependencies for Object Definitions” on page 5-22.)

## Calendar Terms and Concepts

Table 5-1 describes important terms and concepts that are used in the Calendar Base application.

**Table 5-1 Calendar Terms and Concepts**

Term	Definition
Calendar Base	<p>Is a 1 to 20-character name for a collection of user-defined and TOM-supplied, named periods of time that you can use when creating schedules for an object's availability and for other object definitions. Only one Calendar Base can be active per sysplex at any one time.</p> <p>The first time TOM is started, the name of the Calendar Base in use is DEFAULT.</p> <p>You can create multiple Calendar Bases. Reasons to have more than one Calendar Base include:</p> <ul style="list-style-type: none"> <li>• Having a Calendar Base that you want to use to test changes being made without causing the production Calendar Base to become unstable</li> <li>• Creating additional Calendar Bases that you can use for specific recurring events or routine changes such as weekend backups.</li> <li>• Having a disaster recovery Calendar Base that you can specify in the event that you need to recover an unexpectedly lost system</li> </ul> <p>When defining a name for additional Calendar Bases, the following names are already used by TOM and not allowed to be specified as a Calendar Base name:</p> <ul style="list-style-type: none"> <li>• DEFAULT</li> <li>• ACTIVE</li> <li>• BASE</li> </ul>
Default	<p>The name of the Calendar Base that comes up the first time TOM is started. To use a different Calendar Base, you must first create one from the Calendar Bases application and then use the Activate line command to switch to the new Calendar Base.</p>
Day	<p>Is a user-specified collection of days.</p> <p>When creating the collection of days, you can select from: MON, TUE, WED, THU, FRI, SAT and SUN.</p> <p>You must also give the group of days a 1 to 20-character Day name (the Day name can contain blanks).</p> <p>For example, you can select the days MON, TUE, WED and THU and name this collection PRODUCTION DAYS. You can use the name PRODUCTION DAYS on the Schedules pop-up (or refer to "When to Use a Calendar Base" on page 5-2 for other pop-ups on the Started Task Definition panel) to specify that an object is either active (or inactive) on every Monday, Tuesday, Wednesday, and Thursday.</p> <p>Another example is to select WED, SAT and SUN and name this selection BACKUPS. When BACKUPS is specified, the object will be active (or inactive) on Wednesdays and Saturday and Sunday.</p> <p>In addition, TOM provides the entries WKN (Saturday and Sunday), WKD (weekdays (Monday through Friday), and DLY (every day of the week) that you can use in object definitions.</p>

**Table 5-1 Calendar Terms and Concepts**

<b>Term</b>	<b>Definition</b>
Period	<p>Is a user-specified From and To set of dates and times of day.</p> <p>You can specify and name a range of dates (in <b>From yyyy/mm/dd, To yyyy/mm/dd</b> format) and times (in <b>From hh:mm, To hh:mm</b> format).</p> <p>You must give a Period name a 1 to 20-character Period name (the Period name can contain blanks).</p> <p>For example, you can create a Period named MARCH from 09:00 on 2003/03/01 to 17:00 on 2003/03/31. You can use the MARCH Period on the Schedules pop-up (or refer to “When to Use a Calendar Base” on page 5-2 for other pop-ups on the Started Task Definition panel) to specify that an object is active (or inactive) from 9 AM, March 1, 2003 to 5:00 PM, March 31, 2003.</p> <p>ALWAYS is the name of a TOM-supplied Period that when used, means the object will be always available or always unavailable.</p>
Time	<p>Is a user-specified time of day.</p> <p>You can select and name a time (in <b>From hh:mm, To hh:mm</b> format) that you can use on the Schedules pop-up panel to specify when an object should be active or inactive. A Time name can be up to 20 characters long and can contain blanks.</p> <p>For example, you can create a Time named LUNCH from 13:00 to 14:00. You can use the LUNCH Time on the Schedules pop-up (or refer to “When to Use a Calendar Base” on page 5-2 for other pop-ups on the Started Task Definition panel) to specify that an object is active (or inactive) from 1 PM to 2 PM.</p>
Set	<p>Is a way to collect together and name any combination of already created Periods, Days, or Times.</p> <p>A Set name can be up to 20 characters long and can contain blanks.</p> <p>Creating a Set allows you to group together Periods, Days, or Times (that multiple objects might have in common) as times to be active (or inactive). Using Sets may be especially useful when a group of objects all have the same complicated schedule. When you use a Set, you do not have to specify the complicated schedule for each object individually.</p>
Validate	<p>Is a line command from the Calendar Bases application that you can use in the LC column next to the name of a Calendar Base when you want to verify that all the Calendar entries used by objects are available in a Calendar Base.</p> <p>When a Validate is performed, TOM ensures that any named calendar entries used by objects in the active Definition Base are also defined in the selected Calendar Base. A short and long message will appear when entries are missing. The long message displays the name of the first entry found that is missing.</p>
Activate	<p>Is a line command from the Calendar bases application that you can use in the LC column next to the name of a Calendar Base you want to become active and stop the current Calendar Base.</p> <p>When you use the Activate line command, a Validate action is also performed before the new Calendar Base is activated. If the Validate action discovers that any calendar entries used by objects in the Definition Base are not defined in the selected Calendar Base, then the Activate command will not complete. An short error message is displayed and the long message displays the name of the first calendar entry that is missing.</p>

**Table 5-1 Calendar Terms and Concepts**

<b>Term</b>	<b>Definition</b>
Calendar Dependency	<p>Is a term you will see while defining the following object definitions for an object on the Started Task Definition panel:</p> <ul style="list-style-type: none"> <li>• Pre-start EXECs</li> <li>• Post-start EXECs</li> <li>• Pre-stop EXECs</li> <li>• Post-stop EXECs</li> <li>• Start retry commands</li> <li>• Stop retry commands</li> </ul> <p>and</p> <ul style="list-style-type: none"> <li>• Schedules</li> </ul> <p>For any of these definitions, you can specify that this definition is effective only during a specific time or date (or a combination). All of the Periods, Days, Times and Sets that you create with the Calendar Bases application can be used (and viewed) from the object definition input fields on Started Task Definition panel.</p>
Include	<p><b>Include</b> specifies that the object status should be ACTIVE during the entered periods of time.</p>
Exclude	<p><b>Exclude</b> specifies that the object status should be STOPPED during the entered periods of time.</p>

# Creating a New Calendar Base

To create a new Calendar Base:

**Note:** You can also follow these steps to edit and change the default Calendar Base.

**Step 1** On the TOM Primary Options Menu, enter **4** on the **Option** line (Figure 5-1).

**Figure 5-1 TOM Primary Options Menu**

```

BMC Software                TOM Primary Options Menu

Option ==> 4

1 Started Task      - Define, display and manage objects      User ID: BAOMXY1
2 Sets             - Define, display and manage sets          Server Id: TOMN1
3 Status           - Define, display and manage systems       Release: 1.1D
4 Calendar         - Define, display and manage calendars     System: SJSD

A Administration   - System Administration
M Messages         - List messages
L Log              - Display Log
X Exit            - Exit

Copyright BMC Software, Inc. 2003
    
```

The Calendar Bases panel is displayed (Figure 5-2).

**Figure 5-2 Calendar Bases Panel**

```

                                Calendar Bases                Row 1 to 4 of 4
Command ==>                                Scroll ==> CSR

    Active base: DEFAULT

Primary commands: ADD CMDSHOW CUST SORT
Line commands: A-activate D-delete E-edit R-repeat V-validate

LC Calendar Base      Active
-----
__ DEFAULT            YES
***** End of Data *****
    
```

**Step 2** To create a new Calendar Base, enter **ADD** on the **Command** line.

The New Calendar Base pop-up panel is displayed (Figure 5-3).

**Figure 5-3 New Calendar Base Pop-Up Panel**

```

Command ==>

New Base Name:

Press END to save or CANCEL to exit
    
```

**Step 3** In the **New Base Name** field, enter the 1- to 20-character name of the new Calendar Base.

**Step 4** Press END to save and exit.

The new Calendar Base name is added on the Calendar Bases panel (Figure 5-4).

**Figure 5-4 Calendar Bases Panel: New Base Added**

```

Command ==>
Calendar Bases
Row 1 to 4 of 4
Scroll ==> CSR

Active base: DEFAULT

Primary commands: ADD CMDSHOW CUST SORT
Line commands: A-activate D-delete E-edit R-repeat V-validate

LC Calendar Base      Active
-----
__ DEFAULT            YES
__ PRODUCTION

***** End of Data *****
    
```

**Note:** To add Days, Periods, Times, and Sets to the new Calendar Base, refer to the other sections in this chapter.

## Adding a Day Entry

A Day entry in the Calendar Base application represents a collection where you can specify individual days of the week or a combination of days of the week.

To add a Day entry to a new Calendar Base:

**Step 1** On the TOM Primary Options Menu, enter **4** on the **Option** line.

The Calendar Bases panel is displayed (Figure 5-5).

**Figure 5-5 Adding Days to a Calendar Base**

```

                                Calendar Bases                                Row 1 to 4 of 4
Command ==>                                                                Scroll ==> CSR

    Active base: DEFAULT

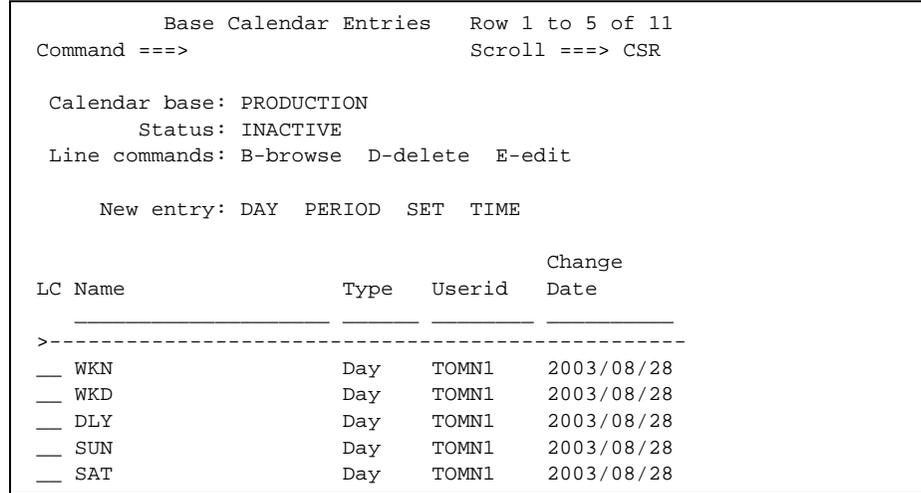
Primary commands: ADD  CMDSHOW  CUST  SORT
Line commands:  A-activate  D-delete  E-edit  R-repeat  V-validate

LC Calendar Base      Active
-----
__ DEFAULT            YES
__ PRODUCTION
***** End of Data *****
    
```

**Step 2** To add a new Day entry to a Calendar Base, enter **E** (for Edit) in the **LC** field next to the Calendar Base.

The Base Calendar Entries pop-up panel is displayed (Figure 5-6).

**Figure 5-6 Base Calendar Entries Pop-Up Panel**



**Note:** The default list of possible entries for Days is WKN, WKD, DLY, SUN, SAT, FRI, THU, WED, TUE, MON, and a default Period of ALWAYS already appears in the pop-up panel.

**Step 3** To add a new Day entry, place the cursor on the **DAY** hyperlink and press **Enter**.

The Calendar Day pop-up panel is displayed (Figure 5-7).

**Figure 5-7 Calendar Day Pop-Up Panel**



**Step 4** In the **Name** field, enter the 1- to 20-character name for the new Day entry.

**Step 5** In the **Days** field, enter the 3-character name for each day that you want to specify together.

For example, you might have an object that needs to be active on Mondays, Wednesdays, and Fridays. Refer to Figure 5-8.

**Figure 5-8 Calendar Day Pop-Up Panel: Example**

```

Command ==>

Name: BATCHDAYS

Days: MON WED FRI  _  _  _  _  (SUN MON TUE WED THU FRI SAT)
    
```

**Step 6** Press END to save and exit.

The new Day entry name (BATCHDAYS) is added (Figure 5-9).

**Figure 5-9 Base Calendar Entries Pop-Up Panel: New Day Entry Added**

```

          Base Calendar Entries   Row 1 to 5 of 11
Command ==>                      Scroll ==> CSR

Calendar base: PRODUCTION
          Status: INACTIVE
Line commands: B-browse D-delete E-edit

          New entry: DAY PERIOD SET TIME

LC Name          Type   Userid   Change
          -----
>-----
_ BATCHDAYS      Day    TOMN1   2003/08/28
_ WKN            Day    TOMN1   2003/08/28
_ WKD            Day    TOMN1   2003/08/28
_ DLY            Day    TOMN1   2003/08/28
_ SUN            Day    TOMN1   2003/08/28
_ SAT            Day    TOMN1   2003/08/28
    
```

**Step 7** Press END to save and exit again.

The Calendar Bases panel is redisplayed.

In this example, the newly added Day entry named BATCHDAYS can now be selected as a Calendar Dependency from the Started Task Definition panel, as part of the definition for any (or all) of the following object definitions:

- Schedules
- Pre-start EXECs
- Post-start EXECs
- Pre-stop EXECs
- Post-stop EXECs
- Start retry commands
- Stop retry commands

See also:

- “Adding a Period Entry” on page 5-12
- “Adding a Time Entry” on page 5-15
- “Adding Items to a Calendar Base Set” on page 5-18

# Adding a Period Entry

A Period entry in the Calendar Base application basically represents a span of time between one date and time to another date and time. When you specify Periods, you must specify both a date and a time.

To add a Period entry to a new Calendar Base:

**Step 1** On the TOM Primary Options Menu, enter **4** on the **Option** line.

The Calendar Bases panel is displayed (Figure 5-10).

**Figure 5-10 Adding Periods to a Calendar Base**

```

                                Calendar Bases                                Row 1 to 4 of 4
Command ==>                                                                Scroll ==> CSR

    Active base: DEFAULT

Primary commands: ADD  CMDSHOW  CUST  SORT
Line commands:  A-activate  D-delete  E-edit  R-repeat  V-validate

LC Calendar Base      Active
-----
__ DEFAULT            YES
__ PRODUCTION
***** End of Data *****
    
```

**Step 2** To add a new Period to a Calendar Base, enter **E** (for Edit) in the **LC** field next to the Calendar Base.

The Base Calendar Entries pop-up panel is displayed (Figure 5-11).

**Figure 5-11 Base Calendar Entries Pop-Up Panel**

```

                                Base Calendar Entries                                Row 1 to 5 of 11
Command ==>                                                                Scroll ==> CSR

    Calendar base: PRODUCTION
    Status: INACTIVE
    Line commands: B-browse  D-delete  E-edit

    New entry: DAY  PERIOD  SET  TIME

LC Name      Type  Userid  Change
-----
>-----
__ WKN       Day  TOMN1  2003/08/28
__ WKD       Day  TOMN1  2003/08/28
__ DLY       Day  TOMN1  2003/08/28
__ SUN       Day  TOMN1  2003/08/28
__ SAT       Day  TOMN1  2003/08/28
    
```

- Step 3** To add a new Period entry, place the cursor on the **PERIOD** hyperlink and press **Enter**.

The Calendar Period pop-up panel is displayed (Figure 5-12).

**Figure 5-12 Calendar Period Pop-Up Panel**

```

                                Calendar Period
Command ==>>

Name:

From
  Date: ____ / __ / __      Time: __ : __      ( yyyy/mm/dd hh:mm )
To
  Date: ____ / __ / __      Time: __ : __      ( yyyy/mm/dd hh:mm )
Press END to save or CANCEL to exit

```

- Step 4** In the **Name** field, enter the 1- to 20-character name for the new Period entry.
- Step 5** In the **From Date** and **From Time** fields, enter the date (in yyyy/mm/dd format) and time (in hh:mm format) for the beginning of the period of time.
- Step 6** In the **To Date** and **To Time** fields, enter the date (in yyyy/mm/dd format) and time (in hh:mm format) for the end of the period of time.

Refer to Figure 5-13 for an example.

**Figure 5-13 Calendar Period Entry Pop-Up Panel: Example**

```

                                Calendar Period
Command ==>>

Name: BATCHDOWNPERIOD

From
  Date: 2003 / 10 / 15      Time: 08 : 00      ( yyyy/mm/dd hh:mm )
To
  Date: 2003 / 10 / 18      Time: 13 : 00      ( yyyy/mm/dd hh:mm )
Press END to save or CANCEL to exit

```

The example in Figure 5-13 shows a new Period entry called **BATCHDOWNPERIOD** which will make an object available or unavailable on October 15, 2003, from 8:00 A.M. to October 18, 2003, at 1:00 P.M.

- Step 7** Press **END** to save and exit.

The new Period entry name is added (Figure 5-14).

**Figure 5-14 Base Calendar Entries Pop-Up Panel: New Period Entry Added**

```

Base Calendar Entries      Row 1 to 5 of 11
Command ==>>>           Scroll ==>>> CSR

Calendar base: PRODUCTION
Status: INACTIVE
Line commands: B-browse D-delete E-edit

New entry: DAY PERIOD SET TIME

LC Name                    Type      Userid    Change
                        _____
>-----
__ TUE                      Day      TOMN1    2003/08/16
__ MON                      Day      TOMN1    2003/08/16
__ BATCHDOWNTIME          Period  BAOMXY1  2003/08/28
__ ALWAYS                  Period  TOMN1    2003/08/16

```

**Step 8** Press END to save and exit again.

The Calendar Bases panel is redisplayed.

In this example, the newly added Period entry named BATCHDOWNTIME can now be selected as a Calendar Dependency from the Started Task Definition panel, as part of the definition for any (or all) of the following object definitions:

- Schedules
- Pre-start EXECs
- Post-start EXECs
- Pre-stop EXECs
- Post-stop EXECs
- Start retry commands
- Stop retry commands

See also:

- “Adding a Day Entry” on page 5-8
- “Adding a Time Entry” on page 5-15
- “Adding Items to a Calendar Base Set” on page 5-18

## Adding a Time Entry

A Time entry in the Calendar Base application is basically a period of time that occurs between one time and another time within the same 24-hour period.

To add a Time entry to a new Calendar Base:

**Step 1** On the TOM Primary Options Menu, enter **4** on the **Option** line.

The Calendar Bases panel is displayed (Figure 5-15).

**Figure 5-15 Adding a Time Entry to a Calendar Base**

```

                                Calendar Bases                                Row 1 to 4 of 4
Command ==>                                                                Scroll ==> CSR

    Active base: DEFAULT

Primary commands: ADD CMDSHOW CUST SORT
Line commands: A-activate D-delete E-edit R-repeat V-validate

LC Calendar Base      Active
-----
__ DEFAULT            YES
__ PRODUCTION

***** End of Data *****

```

**Step 2** To add a new Time to the Calendar Base, enter **E** (for Edit) in the **LC** field next to the Calendar Base.

The Base Calendar Entries pop-up panel is displayed (Figure 5-16).

**Figure 5-16 Base Calendar Entries Pop-Up Panel**

```

                                Base Calendar Entries                        Row 1 to 5 of 11
Command ==>                                                                Scroll ==> CSR

    Calendar base: PRODUCTION
    Status: INACTIVE
    Line commands: B-browse D-delete E-edit

    New entry: DAY PERIOD SET TIME

LC Name                Type      Userid   Change
-----
>-----
__ WKN                 Day      TOMN1   2003/08/28
__ WKD                 Day      TOMN1   2003/08/28
__ DLY                 Day      TOMN1   2003/08/28
__ SUN                 Day      TOMN1   2003/08/28
__ SAT                 Day      TOMN1   2003/08/28

```

- Step 3** To add a new Time entry, place the cursor on the **TIME** hyperlink and press **Enter**.

The Calendar Time pop-up panel is displayed (Figure 5-17).

**Figure 5-17 Calendar Time Pop-Up Panel**

```

                                Calendar Time
Command ==>>

Name:

From:  __ :  __      ( hh:mm )

      To:  __ :  __      ( hh:mm )

Press END to save or CANCEL to exit

```

- Step 4** In the **Name** field, enter the 1- to 20-character name for the new Time entry.
- Step 5** In the **From** field, enter the time (in hh:mm format) for the beginning of the new Time entry.
- Step 6** In the **To** field, enter the time (in hh:mm format) for the end of the period of time.

Refer to Figure 5-18 for an example.

**Figure 5-18 Calendar Time Pop-Up Panel: Example**

```

                                Calendar Time
Command ==>>

Name: BATCHTIME

From: 08 : 00      ( hh:mm )

      To: 10 : 00      ( hh:mm )

Press END to save or CANCEL to exit

```

The example in Figure 5-18 shows a new Time entry called BATCHTIME, which will make an object available or unavailable from 8:00 A.M. to 10:00 A.M.

- Step 7** Press **END** to save and exit.

The new Time entry name is added (Figure 5-19).

**Figure 5-19 Base Calendar Entries Pop-Up Panel: New Time Entry Added**

```

Base Calendar Entries      Row 1 to 5 of 11
Command ===>              Scroll ==> CSR

Calendar base: PRODUCTION
Status: INACTIVE
Line commands: B-browse D-delete E-edit

New entry: DAY PERIOD SET TIME

LC Name                    Type      Userid    Change
                        _____
>-----
__ TUE                     Day      TOMN1    2003/08/16
__ MON                     Day      TOMN1    2003/08/16
__ BATCHDOWNPERIOD        Period   BAOMXY1  2003/08/28
__ BATCHTIME              Time     BAOMXY1  2003/08/28

```

**Step 8** Press END to save and exit again.

The Calendar Bases panel is re-displayed.

In this example, the newly added Time entry named BATCHTIME can now be selected as a Calendar Dependency from the Started Task Definition panel, as part of the definition for any (or all) of the following object definitions:

- Schedules
- Pre-start EXECs
- Post-start EXECs
- Pre-stop EXECs
- Post-stop EXECs
- Start retry commands
- Stop retry commands

See also:

- “Adding a Day Entry” on page 5-8
- “Adding a Period Entry” on page 5-12
- “Adding Items to a Calendar Base Set” on page 5-18

# Adding Items to a Calendar Base Set

For complicated object schedules, you might need to combine Day, Period, or Time entries into a single entity called a Set.

To add a Set to a new Calendar Base:

**Step 1** On the TOM Primary Options Menu, enter **4** on the **Option** line.

The Calendar Bases panel is displayed (Figure 5-20).

**Figure 5-20 Adding a Set Entry to a Calendar Base**

```

                                Calendar Bases                                Row 1 to 4 of 4
Command ==>                                Scroll ==> CSR

    Active base: DEFAULT

Primary commands: ADD  CMDSHOW  CUST  SORT
Line commands:  A-activate  D-delete  E-edit  R-repeat  V-validate

LC Calendar Base      Active
-----
__ DEFAULT            YES
__ PRODUCTION
***** End of Data *****
    
```

**Step 2** To add a new Set to a Calendar Base, enter **E** (for Edit) in the **LC** field next to the Calendar Base.

The Base Calendar Entries pop-up panel is displayed (Figure 5-21).

**Figure 5-21 Base Calendar Entries Pop-Up Panel**

```

                                Base Calendar Entries                                Row 1 to 5 of 11
Command ==>                                Scroll ==> CSR

    Calendar base: PRODUCTION
    Status: INACTIVE
    Line commands: B-browse  D-delete  E-edit

    New entry: DAY  PERIOD  SET  TIME

LC Name      Type      Userid      Change
-----
>-----
__ WKN       Day      TOMN1      2003/08/28
__ WKD       Day      TOMN1      2003/08/28
__ DLY       Day      TOMN1      2003/08/28
__ SUN       Day      TOMN1      2003/08/28
__ SAT       Day      TOMN1      2003/08/28
    
```

**Step 3** To add a new Set entry, place the cursor on the **SET** hyperlink and press **Enter**.

The Calendar Set pop-up panel is displayed (Figure 5-22).

**Figure 5-22 Calendar Set Pop-Up Panel**

```

                                Calendar Set
Command ==>

New Set Name:

Press END to save or CANCEL to exit
    
```

**Step 4** In the **New Set Name** field, enter the 1- to 20-character name for the new Set entry.

Refer to Figure 5-23 for an example.

**Figure 5-23 Calendar Set Pop-Up Panel: Example**

```

                                Calendar Set
Command ==>

New Set Name: BATCHSET

Press END to save or CANCEL to exit
    
```

The example in Figure 5-23 shows a new Set entry called BATCHSET.

**Step 5** Press **END** to save and exit.

The new Set entry name is added (Figure 5-24).

**Figure 5-24 Base Calendar Entries Pop-Up Panel: New Set Entry Added**

```

                                Base Calendar Entries   Row 1 to 5 of 11
Command ==>                                           Scroll ==> CSR

Calendar base: PRODUCTION
Status: INACTIVE
Line commands: B-browse D-delete E-edit

New entry: DAY PERIOD SET TIME

LC Name          Type      Userid   Change
                -----
>-----
__ BATCHSET      Set       BAOMXY1  2003/08/28
__ TUE           Day      TOMN1   2003/08/16
__ MON           Day      TOMN1   2003/08/16
__ BATCHDOWNPERIOD  Period  BAOMXY1  2003/08/28
__ BATCHTIME     Time     BAOMXY1  2003/08/28
    
```

**Step 6** To add Day, Period, or Time entries to the new BATCHSET entry, enter **E** (for Edit) in the **LC** field next to BATCHSET.

The Base Calendar Entries panel is displayed (Figure 5-25).

**Figure 5-25 Base Calendar Entries Pop-Up Panel: Edit New Set Entry**

```

Base Calendar Entries
Row 0 to 0 of 0
Command ==> Scroll ==> CSR

Calendar base: PRODUCTION
Calendar set: BATCHSET
Primary command: ADD
Line commands: B-browse D-delete

LC Name          Type
-----
***** End of Data *****
    
```

**Step 7** Enter the **ADD** primary command on the Command line to display a list of the available calendar entries that you can select (Figure 5-26).

**Figure 5-26 Base Calendar Entries Pop-Up Panel: Edit New Set Entry (2)**

```

Base Calendar Entries
Row 1 to 7 of 7
Command ==> Scroll ==> CSR

Calendar base: PRODUCTION
Calendar set: BATCHSET
Line commands: B-browse S-select

LC Name          Type
-----
__ BATCHDAYS     Day
__ WKN           Day
__ WKD           Day
__ DLY           Day
__ SUN           Day
__ BATCHDOWNPERIOD Period
__ BATCHTIME     Time
***** End of Data *****
    
```

**Step 8** In the **LC** field, enter **S** (for Select) next to the calendar entries that you want to add to the Set named BATCHSET.

**Step 9** Press **END** to save.

Figure 5-27 shows the two calendar entries that were selected in this example.

**Figure 5-27 Base Calendar Entries Pop-Up Panel: Edit New Set Entry (3)**

```

Row 1 to 2 of 2
Command ==>          Scroll ==> CSR

Calendar base: PRODUCTION
Calendar set: BATCHSET
Primary command: ADD
Line commands: B-browse D-delete

LC Name              Type
-----
___ DLY              Day
___ BATCHDAYS       Day
***** End of Data *****

```

**Step 10** Press END to save.

**Step 11** Press END to save again.

The Calendar Bases panel is displayed.

## Entering Calendar Dependencies for Object Definitions

When you are creating object definitions on the Started Task Definition panel, you can use the following fields to specify Calendar Dependencies:

- **Schedules** field (used by TOM to determine when an object might need to be made STOPPED)
- **Pre-start EXECs** field (allows you to specify that TOM schedules an EXEC prior to starting the object)
- **Post-start EXECs** field (allows you to specify that TOM schedules an EXEC after starting the object)
- **Pre-stop EXECs** field (allows you to specify that TOM schedules an EXEC prior to stopping the object)
- **Post-stop EXECs** field (allows you to specify that TOM schedules an EXEC after stopping the object)
- **Start retry commands** field (allows you to specify that TOM attempts to start the object with additional retry start commands)
- **Stop retry commands** field (allows you to specify that TOM attempts to stop the object with additional retry stop commands)

By default, when no Calendar Dependencies are defined as part of the object's definition in the **Schedule** field, TOM will always try to maintain the object's status as ACTIVE. In addition, if you do not specify any Calendar Dependencies for the other six fields, TOM also always tries to take the actions specified in these fields (such as schedule any defined EXECs or issue any defined start or stop retry commands).

When you enter a Calendar Dependency for the **Schedule** field or as part of the object definition for the other fields, TOM checks and evaluates the time entries you specified. Based on the times in the Schedule or Calendar Dependency fields, TOM proceeds with processing the rest of the object definition, such as scheduling an EXEC associated with the object or issuing additional start or stop retry commands.

**Note:** The discussion focuses on how to define Calendar Dependencies for the **Schedule** pop-up panels, but the information is also valid for all other object definitions where Calendar Dependencies can be specified.

To define periods of time when an object should be ACTIVE or STOPPED:

**Step 1** Place the cursor on the **Schedules** hyperlink of the Started Task Definition panel and press **Enter**.

The following panel is displayed:

**Figure 5-28 SCHEDULE Calendar Dependencies Panel**

```

SCHEDULE Calendar Dependencies
Row 0 to 0 of 0
Command ==> Scroll ==> CSR

Calendar base: DEFAULT
Primary Commands: ADD CMDSHOW
Line commands: B-browse D-delete E-edit I-insert

Include logic: AND (AND or OR)

LC T During Day From Date Time To Date Time
-----
>-----
***** End of Data *****
    
```

**Step 2** On the **Command** line, enter **ADD**.

See Figure 5-29 for an example of the Calendar Dependency panel that is displayed.

**Note:** To add additional or multiple Calendar Dependencies, you must redisplay this panel with the ADD primary command from Figure 5-28. You cannot add multiple Calendar Dependencies on the panel at one time.

**Figure 5-29 Calendar Dependency Panel**

```

Calendar Dependency
Command ==>

Type: ( I -Include or E -Exclude )

During named
Period: (Enter ? to select from list)
Set: (Enter ? to select from list)
Day: (Enter ? to select from list)
Time: (Enter ? to select from list)

From named To named
Period: Period:
Day: Day:
Time: Time:

From specific
Date: / / Time: : ( yyyy/mm/dd hh:mm )
To specific
Date: / / Time: : ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

For each Calendar Dependency that you enter, you must select whether this entry is an Include or an Exclude definition, where

- Include specifies that the object status should be ACTIVE during the entered periods of time
- Exclude specifies that the object status should be STOPPED during the entered periods of time

After each Calendar Dependency is entered, the SCHEDULE Calendar Dependencies panel shows what you entered and whether it is an Include or an Exclude definition.

## How the Include logic Field Works

The way in which TOM evaluates multiple Calendar Dependencies also depends on the value that you specify in the **Include logic** field. Valid options are as follows:

- **Include logic=AND**

When you specify AND, all the time definitions that you create with Include must be true for TOM to start the object and make the status ACTIVE.

- **Include logic=OR**

When you specify OR, TOM tries to start the object and make the status ACTIVE when TOM determines that at least one Include specification is true.

The specification in the **Include logic** field does not affect Exclude Calendar Dependency definitions. Refer to page 5-36 for an example of a definition that uses only Exclude Calendar Dependency definitions.

For examples of how specifying AND or OR in the **Include logic** field works for Include definitions, refer to the examples on page 5-29 and page 5-32.

## Calendar Dependency Panel

Figure 5-30 shows the Calendar Dependency panel.

**Figure 5-30 Calendar Dependency Panel**

```

                                Calendar Dependency
Command ==>

Type:      ( I -Include or E -Exclude )

During named
Period:           (Enter ? to select from list)
Set:              (Enter ? to select from list)
Day:              (Enter ? to select from list)
Time:            (Enter ? to select from list)

From named        To named
Period:           Period:
Day:              Day:
Time:             Time:

From specific
Date: / /        Time: : ( yyyy/mm/dd hh:mm )
To specific
Date: / /        Time: : ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

Table 5-2 describes the guidelines for entering a Calendar Dependency on the Calendar Dependency panel.

**Table 5-2 Fields for Entering a Calendar Dependency**

Field Name or Section	Required?	Valid Values and Definition
<b>Type</b>	Yes	Valid values are: Include or Exclude. Specifying Include means that the object should be ACTIVE during the specified times. Specifying Exclude means that the object should be STOPPED during the specified times.
For the rest of the panel, you can select to make an entry in only one of the three sections of the panel: <ul style="list-style-type: none"> <li>• <b>During named</b></li> <li>• <b>From named — To named</b></li> <li>• <b>From specific — To specific</b></li> </ul> <b>Note:</b> The only exception to this rule is that you can specify an entry in the <b>During</b> named section with <b>From specific — To specific</b> to allow for compatibility for those who are converting calendar definitions from the MAINVIEW AutoOPERATOR Continuous State Manager application.		

**Table 5-2 Fields for Entering a Calendar Dependency**

Field Name or Section	Required?	Valid Values and Definition
<b>During named</b>	No	In this section, you can enter only one named entry for the <b>Period</b> , <b>Set</b> , <b>Day</b> , or <b>Time</b> fields. These named entries must be predefined in the currently active Calendar Base. For more information refer to “Creating Calendar Bases” on page 5-1 for information about how to add these named entries.  Enter a question mark, (?) in the Period, Set, Day or Time fields to view a list of the currently defined named entries that you can select from.
<b>From named — To named</b>	No	In this section, you can enter a starting time (From named) and ending time (To named) using previously defined <b>Period</b> , <b>Day</b> , or <b>Time</b> entries.  You cannot enter mixed pairs of entries such as entering a <b>Period</b> for the <b>From named</b> and a <b>Time</b> for the <b>To named</b> fields.
<b>From specific — To specific</b>	No	In this section you can specify dates and times in the following combinations:  <ul style="list-style-type: none"> <li>• <b>From specific, Date and Time</b> and <b>To specific Date and Time</b></li> <li>• <b>From specific, Date</b> (no Time specified) and <b>To specific, Date</b> (no Time specified)</li> <li>• <b>From specific, Time</b> (no Date specified) and <b>To specific, Time</b> (no Date specified)</li> </ul>

## Examples of Entering Calendar Dependencies

The following sections contain examples of how to enter different Calendar Dependencies. The examples also show how choosing AND or OR logic can affect the way that TOM evaluates the Calendar Dependency and what actions TOM takes based on the evaluations.

**Note:** The scenarios in these examples describe how the Calendar Dependency affects how TOM manages the status of the object. However, the way TOM evaluates the Calendar Dependencies is the same even when they are defined for the other properties (such as Pre-start EXECs, Start and Stop retry commands, and so on.)

**Example 1**

To define that TOM should maintain the object status as ACTIVE all the time except from 12:00 P.M. to 1:00 P.M. on weekdays, enter the following Exclude Calendar Dependency:

**Figure 5-31 Entering an Exclude Calendar Dependency**

```

Calendar Dependency
Command ==>

Type: E ( I -Include or E -Exclude )

During named
Period: (Enter ? to select from list)
Set: (Enter ? to select from list)
Day: WKD (Enter ? to select from list)
Time: (Enter ? to select from list)

From named To named
Period: Period:
Day: Day:
Time: Time:

From specific
Date: / / Time: 12 : 00 ( yyyy/mm/dd hh:mm )
To specific
Date: / / Time: 13 : 00 ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

Table 5-3 describes the values that are entered in Figure 5-31 and contains brief explanations.

**Table 5-3 Entry Descriptions for Figure 5-31**

Field	Enter This Value	Explanation
<b>Type</b>	<b>E</b> for Exclude	Exclude means TOM tries to change the status of an object to STOPPED during Exclude specifications.
<b>During named, Day</b>	<b>WKD</b>	Is a predefined entry that includes all the days of the week: Monday, Tuesday, Wednesday, Thursday, and Friday. Refer to "Adding a Day Entry" on page 5-8 for information about Day entries.
<b>From specific, Time</b>	<b>12:00</b>	In a 24-hour clock, represents 12:00 P.M.
<b>To specific, Time</b>	<b>13:00</b>	In a 24-hour clock, represents 1:00 P.M.

**Example 2**

To define that TOM should maintain an object's status as STOPPED all of the time except from 8:00 A.M. to 11:00 A.M. every morning, enter the following Include Calendar Dependency:

**Figure 5-32 Entering an Include Calendar Dependency**

```

                                Calendar Dependency
Command ==>

Type: I   ( I -Include or E -Exclude )

During named
Period:           (Enter ? to select from list)
Set:              (Enter ? to select from list)
Day:              (Enter ? to select from list)
Time:             (Enter ? to select from list)

From named                To named
Period:                   Period:
Day:                       Day:
Time:                       Time:

From specific
Date: / /                Time: 08 : 00   ( yyyy/mm/dd hh:mm )
To specific
Date: / /                Time: 11 : 00   ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

Table 5-4 describes the values that were entered in Figure 5-32 and contains brief explanations.

**Table 5-4 Entry Descriptions for Figure 5-32**

Field	Enter This Value	Explanation
<b>Type</b>	<b>I</b> for Include	Include means TOM tries to change the status of an object to ACTIVE during Include specifications.
<b>From specific, Time</b>	<b>08:00</b>	In a 24-hour clock, represents 8:00 A.M.
<b>To specific, Time</b>	<b>11:00</b>	In a 24-hour clock, represents 11:00 A.M.

**Example 3**

To define that TOM should maintain an object’s status as ACTIVE only on Mondays from 10:00 A.M. to 2:00 P.M. and on Wednesdays from 3:00 P.M. to 5:00 P.M., enter the following two Include Calendar Dependencies (see Figure 5-33 and Figure 5-34 on page 5-30).

In addition, see Figure 5-35 on page 5-31 which is where you must specify OR logic in the **Include logic** field. Using OR logic is the only way to indicate that TOM should try to maintain the object status as ACTIVE when the first Calendar Dependency is true, *or* when the second Calendar Dependency is true.

Figure 5-33 shows how to enter the first Include Calendar Dependency.

**Figure 5-33 Entering an Include Calendar Dependency: Using OR Logic**

```

                                Calendar Dependency
Command ==>

Type: I   ( I -Include or E -Exclude )

During named
  Period:                               (Enter ? to select from list)
  Set:                                   (Enter ? to select from list)
  Day: MON                               (Enter ? to select from list)
  Time:                                  (Enter ? to select from list)

From named                               To named
  Period:                               Period:
  Day:                                   Day:
  Time:                                  Time:

From specific
  Date:  /  /      Time: 10 : 00      ( yyyy/mm/dd hh:mm )
To specific
  Date:  /  /      Time: 14 : 00      ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

Table 5-5 describes the values that were entered in Figure 5-33 and contains brief explanations.

**Table 5-5 Entry Descriptions for Figure 5-33**

Field	Enter This Value	Explanation
<b>Type</b>	<b>I</b> for Include	Include means TOM tries to change the status of an object to ACTIVE during Include specifications.
<b>During named, Day</b>	<b>MON</b>	Is a predefined entry that represents Monday. Refer to “Adding a Day Entry” on page 5-8 for information about Day entries.

**Table 5-5 Entry Descriptions for Figure 5-33**

Field	Enter This Value	Explanation
<b>From specific, Time</b>	<b>10:00</b>	In a 24-hour clock, represents 10:00 AM.
<b>To specific, Time</b>	<b>14:00</b>	In a 24-hour clock, represents 2:00 PM.

Figure 5-34 shows how to enter the second Include Calendar Dependency.

**Figure 5-34 Entering an Include Calendar Dependency: Using OR Logic (2)**

```

                                Calendar Dependency
Command ==>

Type: I   ( I -Include or E -Exclude )

During named
  Period:                (Enter ? to select from list)
   Set:                  (Enter ? to select from list)
   Day: WED              (Enter ? to select from list)
   Time:                (Enter ? to select from list)

From named                To named
  Period:                Period:
   Day:                  Day:
   Time:                 Time:

From specific
  Date:    /    /    Time: 15 : 00    ( yyyy/mm/dd hh:mm )
To specific
  Date:    /    /    Time: 17 : 00    ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

Table 5-6 describes the values that were entered in Figure 5-34 and contains brief explanations.

**Table 5-6 Entry Descriptions for Figure 5-34**

Field	Enter This Value	Explanation
<b>Type</b>	<b>I</b> for Include	Include means TOM tries to change the status of an object to ACTIVE during Include specifications.
<b>During named, Day</b>	<b>WED</b>	Is a predefined entry that represents Wednesday. Refer to "Adding a Day Entry" on page 5-8 for information about Day entries.
<b>From specific, Time</b>	<b>15:00</b>	In a 24-hour clock, represents 3:00 P.M.
<b>To specific, Time</b>	<b>17:00</b>	In a 24-hour clock, represents 5:00 P.M.

Figure 5-35 shows what the SCHEDULE Calendar Dependencies panel looks like after you have saved the two Include Calendar Dependencies in Figure 5-33 on page 5-29 and Figure 5-34 on page 5-30.

**Note:** On the SCHEDULE Calendar Dependencies, you must also change the specification for the **Include logic** field to **OR**.

**Figure 5-35 SCHEDULE Calendar Dependencies Panel with OR Logic Specified**

```

SCHEDULE Calendar Dependencies
                                     Row 1 to 2 of 2
Command ==>                          Scroll ==> CSR

  Calendar base: DEFAULT
  Primary Commands: ADD CMDSHOW
  Line commands: B-browse D-delete E-edit I-insert

  Include logic: OR (AND or OR)

LC T During Day      From Date  Time      To Date  Time
-----
>-----
__ I WED                15:00    17:00
__ I MON                10:00    14:00
***** End of Data *****

```

If the **Include logic** field is left as **AND**, TOM will never attempt to make this object **ACTIVE** because it would never be both **MON** and **WED** at the same time.

**Example 4**

To define that TOM should maintain an object’s status as ACTIVE during a holiday period from 8:00 A.M. to 5:00 P.M., enter two Include Calendar Dependencies and also specify **AND** in the **Include logic** field.

First, create a Calendar Named Period entry (for example, HOLIDAY, which starts extends from December 24, 2003, until January 2, 2004). Refer to “Adding a Period Entry” on page 5-12 for information.

Figure 5-36 shows a Period entry for HOLIDAY, with Include selected in the **Type** field.

**Figure 5-36 Entering an Include Calendar Dependency: Using AND Logic**

```

                                Calendar Dependency
Command ==>

Type: I   ( I -Include or E -Exclude )

During named
  Period: HOLIDAY                (Enter ? to select from list)
   Set:                          (Enter ? to select from list)
   Day:                          (Enter ? to select from list)
   Time:                          (Enter ? to select from list)

From named                       To named
  Period:                         Period:
   Day:                            Day:
   Time:                            Time:

From specific
  Date:   /   /   Time:   :   ( yyyy/mm/dd hh:mm )
To specific
  Date:   /   /   Time:   :   ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

Table 5-7 describes the values that were entered in Figure 5-36 and contains brief explanations.

**Table 5-7 Entry Descriptions for Figure 5-36**

Field	Enter This Value	Explanation
<b>Type</b>	I for Include	Include means TOM tries to change the status of an object to ACTIVE during Include specifications.
<b>During named, Period</b>	<b>HOLIDAY</b>	Is a user-defined Period entry that represents December 24, 2003 until January 2, 2004. Refer to "Adding a Period Entry" on page 5-12 for information about Period entries.

**Note:** You cannot enter a During named Period entry with a specific time in a single Calendar Dependency. To add a time, you must create a second Calendar Dependency. Figure 5-37 shows specific time entry.

**Figure 5-37 Entering an Include Calendar Dependency: Using AND Logic (2)**

```

                                Calendar Dependency
Command ==>

Type: I   ( I -Include or E -Exclude )

During named
Period:           (Enter ? to select from list)
  Set:            (Enter ? to select from list)
  Day:            (Enter ? to select from list)
  Time:           (Enter ? to select from list)

From named                To named
Period:                   Period:
  Day:                     Day:
  Time:                    Time:

From specific
Date: / /                Time: 08 : 00   ( yyyy/mm/dd hh:mm )
To specific
Date: / /                Time: 17 : 00   ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

Table 5-8 describes the values that were entered in Figure 5-37 and contains brief explanations.

**Table 5-8 Entry Descriptions for Figure 5-37**

Field	Enter This Value	Explanation
<b>Type</b>	I for Include	Include means TOM tries to change the status of an object to ACTIVE during Include specifications.

**Table 5-8 Entry Descriptions for Figure 5-37**

Field	Enter This Value	Explanation
<b>From specific, Time</b>	<b>08:00</b>	In a 24-hour clock, represents 8:00 AM.
<b>To specific, Time</b>	<b>17:00</b>	In a 24-hour clock, represents 5:00 PM.

Figure 5-38 shows what the SCHEDULE Calendar Dependencies panel looks like after you have saved the two Include Calendar Dependencies in Figure 5-36 on page 5-32 and Figure 5-37 on page 5-33.

On the SCHEDULE Calendar Dependencies, you must also change the specification for the **Include logic** field to **AND**.

**Figure 5-38 SCHEDULE Calendar Dependencies Panel with AND Logic Specified**

```

SCHEDULE Calendar Dependencies
                                Dependency added
Command ==>                      Scroll ==> CSR

    Calendar base: DEFAULT
    Primary Commands: ADD CMDSHOW
    Line commands: B-browse D-delete E-edit I-insert

    Include logic:  AND   (AND or OR)

LC T During Day          From Date  Time      To Date   Time
-----
>-----
__ I                      08:00    17:00
__ I
***** End of Data *****
    
```

If you specify AND logic, TOM attempts to maintain the object status as ACTIVE from 8:00 A.M. to 5:00 P.M. from December 24 through January 2, and it will not be ACTIVE on any of the rest of the days of the year.

**Example 5**

To define that TOM should maintain an object’s status as STOPPED all of the time, enter the following Exclude Calendar Dependency definition:

**Figure 5-39 Entering an Exclude Calendar Dependency: Stopping an Object**

```

Calendar Dependency
Command ==>

Type: E ( I -Include or E -Exclude )

During named
Period: ALWAYS (Enter ? to select from list)
Set: (Enter ? to select from list)
Day: (Enter ? to select from list)
Time: (Enter ? to select from list)

From named To named
Period: Period:
Day: Day:
Time: Time:

From specific
Date: / / Time: : ( yyyy/mm/dd hh:mm )
To specific
Date: / / Time: : ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

Table 5-9 describes the values that were entered in Figure 5-39 and contains brief explanations.

**Table 5-9 Entry Descriptions for Figure 5-39**

Field	Enter This Value	Explanation
<b>Type</b>	<b>E</b> for Exclude	Exclude means TOM tries to change the status of an object to STOPPED during Exclude specifications.
<b>During named, Period</b>	<b>ALWAYS</b>	Is a predefined entry that represents every day of the week. Refer to “Adding a Period Entry” on page 5-12 for information about Day entries

**Example 6**

To define that TOM should maintain an object’s status as STOPPED during the a holiday period from 8:15 A.M. to 8:30 A.M., enter the following two Exclude Calendar Dependencies.

**Note:** The specification in the **Include logic** field does not affect Exclude Calendar Dependency definitions.

First, create a Calendar Named Period entry (for example, HOLIDAY, which extends from December 24, 2003, until January 2, 2004). Refer to “Adding a Period Entry” on page 5-12 for information.

Figure 5-40 shows a During named Period entry for HOLIDAY with Exclude selected in the **Type** field.

**Figure 5-40 Entering an Exclude Calendar Dependency for a Period of Time**

```

                                Calendar Dependency
Command ==>

Type: E   ( I -Include or E -Exclude )

During named
  Period: HOLIDAY                (Enter ? to select from list)
    Set:                          (Enter ? to select from list)
    Day:                          (Enter ? to select from list)
    Time:                        (Enter ? to select from list)

From named                          To named
  Period:                          Period:
    Day:                            Day:
    Time:                          Time:

From specific
  Date:   /   /                   Time:   :   ( yyyy/mm/dd hh:mm )
To specific
  Date:   /   /                   Time:   :   ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```

Table 5-10 describes the values that were entered in Figure 5-40 and contains brief explanations.

**Table 5-10 Entry Descriptions for Figure 5-40**

Field	Enter This Value	Explanation
<b>Type</b>	<b>E</b> for Exclude	Exclude means TOM tries to change the status of an object to STOPPED during Exclude specifications.
<b>During named, Period</b>	<b>HOLIDAY</b>	Is a user-defined Period entry that represents December 24, 2003, until January 2, 2004. Refer to "Adding a Period Entry" on page 5-12 for information about Period entries.

**Note:** You cannot enter a During named Period entry with a specific time in a single Calendar Dependency. To add a time, you must create a second Calendar Dependency.

Figure 5-41 shows specific time entry.

**Figure 5-41 Entering an Exclude Calendar Dependency for a Specific Time**

```

                                Calendar Dependency
Command ==>

Type: E  ( I -Include or E -Exclude )

During named
  Period:                (Enter ? to select from list)
   Set:                  (Enter ? to select from list)
   Day:                  (Enter ? to select from list)
   Time:                  (Enter ? to select from list)

From named                To named
  Period:                 Period:
   Day:                   Day:
   Time:                   Time:

From specific
Date:    /    /          Time: 08 : 15    ( yyyy/mm/dd hh:mm )
To specific
Date:    /    /          Time: 08 : 30    ( yyyy/mm/dd hh:mm )

Press END to save or CANCEL to exit
    
```



---

---

# Chapter 6 Using the Started Task Overview Panel to Manage Objects

This chapter describes how to use the Started Task Overview panel to manage objects. The chapter includes the following topics:

Understanding the Started Task Overview Panel . . . . .	6-2
Viewing Object Information . . . . .	6-2
Managing Objects . . . . .	6-5
Managing Definition Databases . . . . .	6-18

## Understanding the Started Task Overview Panel

After objects, sets, and calendar dependencies have been created, you can use the Started Task Overview panel to manage objects by

- viewing information about objects that are defined to TOM
- editing, browsing, deleting, and repeating objects
- stopping, starting, changing status of, resetting, suspending, and activating objects
- creating new objects by using the ADD primary command and sorting the view by using the CUST and SORT primary commands
- changing definition databases

The following sections describe the Started Task Overview panel and provide examples of some of the commands that you can issue against objects.

### Viewing Object Information

To view object information on the Started Task Overview panel:

**Step 1** On the TOM Primary Options Menu, select option 1, as shown in Figure 6-1.

**Figure 6-1** Displaying the Started Task Overview Panel

```

BMC Software                TOM Primary Options Menu

Option ==> 1

1 Started Task  - Define, display and manage objects      User ID: BAOMXY1
2 Sets         - Define, display and manage sets          Server Id: TOMN1
3 Status       - Define, display and manage systems       Release: 1.1D
4 Calendar     - Define, display and manage calendars     System: SJSJ

A Administration - System Administration
M Messages      - List messages
L Log           - Display Log

X Exit         - Exit

Copyright BMC Software, Inc. 2003

```

The Started Task Overview panel is displayed (Figure 6-2).

**Figure 6-2** Started Task Overview Panel

```

Started Task Overview

Row 1 to 6 of 6
Scroll ==> CSR

Command ==>

Primary Commands: ADD  CMDSHOW  CUST  V-evaluate

Line Commands: B-browse D-delete DD-delete block E-edit R-repeat
                : O-bounce P-stop S-start CH-change status
                : L-lock UL-unlock T-reset U-suspend A-activate
                : C-status K-block UK-unblock M-move J-propagate
                : BT-browse real time

Definition Base: DEFAULT

LC Description          Except Control Type STC
-----
>-----
__ test.objects.one    NO    SUSPEND  NORM AAOCSMN1
__ CICSPROD.SYSA      NO    SUSPEND  AAOCSMN5
__                    NO    SUSPEND  AAOCSMN4
__                    NO    SUSPEND  AAOCSMN3
__                    NO    SUSPEND  AAOCSMN2
__                    NO    SUSPEND  AAOCSMN1
***** End of Data *****

```

The lower half of the panel shows all objects that are defined within this TOM system. Because the object name can be up to 64 characters, the object description is shown (by default) as the first column.

Most of the data is based on the definitions that were specified on the Started Task Definition panel. (Refer to Chapter 3, “Creating and Managing Objects.”)

The online Help provides detailed information about the origins of the data in each column.

- Step 2** To display Help for any column or command, place the cursor anywhere on the column or command name, and then press **PF1 (HELP)**.

## What the Except Column Shows

The **Except** column does not display a user-specified object definition. This column displays one of the following values:

- YES indicates the object is in an Exception condition.
- NO indicates the object is not in an Exception condition.

When an object is in an Exception condition, the object is not in a TOM status. Refer to Appendix A “TOM Statuses” for information about TOM statuses.

**Note:** When an object shows YES in the **Except** column, enter **C** (for status) to see the status of the object on each system that it is running on.

To display Help for any of the primary commands or line commands, place the cursor within the line commands list on the panel or in the LC column, and press **PF1 (HELP)**.

# Managing Objects

You can use the Started Task Overview panel to perform many important commands against objects. Figure 6-3 shows an example of the Started Task Overview panel and the line commands.

**Figure 6-3 Started Task Overview: Line Commands**

```

Started Task Overview
Row 1 to 6 of 6
Scroll ==> CSR
Command ==>
Primary Commands: ADD  CMDSHOW  CUST  V-evaluate

Line Commands: B-browse D-delete DD-delete block E-edit R-repeat
               : O-bounce P-stop S-start CH-change status
               : L-lock UL-unlock T-reset U-suspend A-activate
               : C-status K-block UK-unblock M-move J-propagate
               : BT-browse real time

Definition Base: DEFAULT

LC Description          Except Control Type STC
-----
>-----
__ test.objects.one    NO    SUSPEND  NORM  AAOCSMN1
__ CICSPROD.SYSA      NO    SUSPEND           AAOCSMN5
__                    NO    SUSPEND           AAOCSMN4
__                    NO    SUSPEND           AAOCSMN3
__                    NO    SUSPEND           AAOCSMN2
__                    NO    SUSPEND           AAOCSMN1
***** End of Data *****

```

This section describes how to use the following line commands:

- S (starts an object)
- P (stops an object)
- CH (changes the object's status—for objects that do not have an STC name defined in the object definition)
- T (resets an object to ACTIVE or STOPPED, based on the object's schedule definition)
- U and A (change TOM control of the object between SUSPEND and ACTIVE)

**Note:** The Started Task Overview panel includes online Help for all of the line commands. To display Help for the line commands, place the cursor on the command name on the panel or in the LC column, and then press **PF1** (HELP).

## Starting an Object

When you issue the Start line command against an object, TOM starts the object, regardless of any calendar dependencies or other object definitions. The following procedure describes how to start an object:

- Step 1** On the Started Task Overview panel, enter **S** in the **LC** field next to the name of the object, as shown in Figure 6-4.

**Figure 6-4 Starting an Object**

```

                Started Task Overview
                Row 1 to 6 of 6
Command ==>                Scroll ==> CSR

Primary Commands: ADD  CMDSHOW  CUST  V-evaluate

    Line Commands: B-browse D-delete  DD-delete block E-edit    R-repeat
                  : O-bounce P-stop   S-start      CH-change status
                  : L-lock   UL-unlock T-reset      U-suspend A-activate
                  : C-status K-block  UK-unblock    M-move    J-propagate
                  : BT-browse real time

Definition Base: DEFAULT

LC Description                Except Control  Type STC
-----
>-----
__ test.objects.one          NO      SUSPEND  NORM AAOCSMN1
_S CICSPROD.SYSA            NO      SUSPEND           AAOCSMN5
__                          NO      SUSPEND           AAOCSMN4
__                          NO      SUSPEND           AAOCSMN3
__                          NO      SUSPEND           AAOCSMN2
__                          NO      SUSPEND           AAOCSMN1
***** End of Data *****
    
```

The confirmation pop-up panel is displayed (Figure 6-5).

**Figure 6-5 Confirm Start Pop-Up Panel**

```

Command ==>

    Proceed with start?  N ( Y or N )

    Reset at next IPL?  Y ( Y or N )

    Start with full automation?  Y ( Y or N )

    Bypass Command Count Check?  N ( Y or N )

    System name (or ? for list)  ?

    Press ENTER to continue or END to cancel
    
```

- Step 2** For each question on the confirmation pop-up panel, enter the value that determines how TOM starts the object.

Refer to Table 6-1 for information about what the options for starting an object.

**Table 6-1 Starting an Object: Options**

Option	Description
Proceed with start? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM issues the start command (or EXEC) defined for the object.</p> <p>When you use the Start line command, TOM ignores any schedule or other definitions specified for the object and on the Started Task Overview Panel, YES appears in the Forced column for the object. The status of the object is LOCKED on all other systems defined for the object in the Valid Systems list.</p> <p>TOM displays the object status as ACTIVE on the Started Task Overview panel until you issue a Stop or Reset line command or until the next IPL (refer to the Reset at next IPL option).</p> <p>In addition, TOM does not start the object if it is already active on any system or if the Start command limit count has been reached.</p> <p>No specifies the start command (or EXEC) defined for the objects within the set will not be issued.</p>
Reset at next IPL? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM takes the object out of Forced state after the next IPL and TOM resumes control of the object. The object's schedule and other definitions become effective.</p> <p>No specifies that TOM continues to maintain the object in Forced ACTIVE state after the next IPL. TOM does not use the object's schedule or other definitions.</p>
Start with full automation? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM schedules the Pre-start EXECs before issuing the start command (or EXEC) against the object and the Post-start EXECs after the object becomes ACTIVE.</p> <p>No specifies that TOM issues the Start command (or EXEC) for the object without scheduling the Pre-start and Post-start EXECs.</p>

**Table 6-1 Starting an Object: Options**

Option	Description
Bypass command count check? Y or N	<p>Possible values are Y (yes) or N (no)</p> <p>Yes specifies that TOM does not check to see if the start command count specified in the object's definition (on the <b>Maximum Start Count</b> field of the Started Task Definition panel) has been exceeded. When you specify Y (yes), the start command count is still incremented when the start command is issued and the start command count is reset to zero when the object's definition specifies yes in the <b>Reset start count at termination</b> field of the Started Task Definition panel.</p> <p>No means that TOM checks the command count and will not start the objects if the <b>Maximum Start Count</b> is exceeded.</p>
Start on system	<p>This option is displayed only when the object has more than one system defined in the Valid Systems list and on those systems, the object has a status other than ACTIVE.</p> <p>For example, this option is displayed if the object is:</p> <ul style="list-style-type: none"> <li>• ACTIVE on SJSA</li> <li>• STOPPED on SJSB</li> <li>• FAILURE-ABENDED on SJSC</li> </ul> <p>and you can choose to start the object on SJSB or SJSC.</p> <p>However, if the object is:</p> <ul style="list-style-type: none"> <li>• ACTIVE on SJSA</li> <li>• ACTIVE on SJSB</li> <li>• FAILURE-ABENDED on SJSC</li> </ul> <p>This option is not displayed and TOM starts the object on SJSC. Specify the name of the system where TOM should start the object or enter a question mark (?) to see the systems listed in the Valid Systems for this object.</p>

## Stopping an Object

When you issue the Stop line command against an object, the object is stopped, regardless of any calendar dependencies or other object definitions. The following procedure describes how to stop an object:

- Step 1** On the Started Task Overview panel, enter **P** (for Stop) in the **LC** field next to the name of the object.

**Figure 6-6 Stopping an Object**

```

Started Task Overview
Row 1 to 6 of 6
Command ==> Scroll ==> CSR
Primary Commands: ADD  CMDSHOW  CUST  V-evaluate

Line Commands: B-browse D-delete DD-delete block E-edit R-repeat
                : O-bounce P-stop S-start CH-change status
                : L-lock UL-unlock T-reset U-suspend A-activate
                : C-status K-block UK-unblock M-move J-propagate
                : BT-browse real time

Definition Base: DEFAULT

LC Description          Except Control Type STC
-----
>-----
__ test.objects.one    NO    SUSPEND  NORM AAOCSMN1
_P CICSPROD.SYSA      NO    SUSPEND           AAOCSMN5
__                    NO    SUSPEND           AAOCSMN4
__                    NO    SUSPEND           AAOCSMN3
__                    NO    SUSPEND           AAOCSMN2
__                    NO    SUSPEND           AAOCSMN1
***** End of Data *****

```

The confirmation pop-up panel is displayed (Figure 6-7).

**Figure 6-7 Confirm Stop Pop-Up Panel**

```

Command ==>

Proceed with stop? N ( Y or N )

Reset at next IPL? Y ( Y or N )

Stop with full automation? Y ( Y or N )

Press ENTER to continue or END to cancel

```

- Step 2** For each question on the confirmation pop-up panel, enter the value that determines how TOM stops the object.

Table 6-2 describes the various options for stopping an object.

**Table 6-2 Stopping an Object: Options**

Option	Description
Proceed with stop? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM issues the stop command (or EXEC) defined for the object.</p> <p>When you use the Stop line command, TOM ignores any schedule or other definitions specified for the object</p> <p>TOM displays the object within the set as LOCKED status until you issue a Start or Reset line command or until the next IPL (refer to the Reset at next IPL option).</p> <p>No specifies the stop command (or EXEC) defined for the object within is not issued.</p>
Reset at next IPL? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM takes the object out of LOCKED status after the next IPL and TOM resumes control of the object. The object's schedule and other definitions become effective.</p> <p>No specifies that TOM continues to maintain the object in LOCKED status after the next IPL. TOM does not use the object's schedule or other definitions.</p>
Stop with full automation? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM schedules the Pre-stop EXECs before issuing the stop command (or EXEC) against the object and the Post-stop EXECs after the object becomes STOPPED.</p> <p>No specifies that TOM issues the Stop command (or EXEC) for the object without scheduling the Pre-stop and Post-stop EXECs.</p>
Force stop suspended object? Y or N	<p>Possible values are Y (yes) or N (no).</p> <p>Yes specifies that TOM issues the Stop command even if TOM's control of this object has been suspended.</p>
Stop on system?	<p>This option is displayed only when the object has more than one system defined in the Valid Systems list and on those systems, the object has a status other than ACTIVE.</p> <p>Specify the name of the system where TOM should stop the object or enter a question mark (?) to see the systems listed in the Valid Systems for this object.</p>

## Resetting an Object

When you issue the Reset line command against an object, the object's status is reset to either ACTIVE or STOPPED, based on the object's definition. The following procedure describes how to reset an object:

- Step 1** On the Started Task Overview panel, enter **T** (for Reset) in the **LC** field next to the name of the object (see Figure 6-8).

**Figure 6-8** Resetting an Object

```

Started Task Overview
Row 1 to 6 of 6
Command ==> Scroll ==> CSR

Primary Commands: ADD  CMDSHOW  CUST  V-evaluate

Line Commands: B-browse D-delete DD-delete block E-edit R-repeat
                : O-bounce P-stop S-start CH-change status
                : L-lock UL-unlock T-reset U-suspend A-activate
                : C-status K-block UK-unblock M-move J-propagate
                : BT-browse real time

Definition Base: DEFAULT

LC Description          Except Control Type STC
-----
>-----
__ test.objects.one    NO    SUSPEND  NORM AAOCSMN1
_T CICSPROD.SYSA      NO    SUSPEND           AAOCSMN5
__                    NO    SUSPEND           AAOCSMN4
__                    NO    SUSPEND           AAOCSMN3
__                    NO    SUSPEND           AAOCSMN2
__                    NO    SUSPEND           AAOCSMN1
***** End of Data *****

```

The confirmation pop-up panel is displayed (Figure 6-9).

**Figure 6-9** Confirm Reset Pop-Up Panel

```

Command ==>

Reset all? N ( Y or N )

Reset Forced? N ( Y or N )

Reset Status? N ( Y or N )

Reset Abend count? N ( Y or N )

Reset SUSPEND control? N ( Y or N )

Reset Start command count? N ( Y or N )

Press ENTER to continue or END to cancel

```

**Step 2** For each question on the confirmation pop-up panel, enter the value that determines how TOM resets the object.

**Note:** The Reset line command can cause TOM to start or stop objects with full automation where all of the Pre-start, Post-start, Pre-stop, and Post-stop EXECs are scheduled. The Reset line command is also executed against all systems in the object’s Valid Systems list.

Refer to Table 6-3 for information about the options for resetting the objects.

**Table 6-3 Resetting an Object: Options**

Option	Description
Reset all? Y or N	Possible values are Y (yes) or N (no). Yes specifies that TOM resets the status of the object to either ACTIVE or STOPPED, based on the current schedule definition for the object. TOM resumes complete control of the object’s status and manages the objects according to its defined schedule and all other object definitions. When you specify Yes, all of the other options with this line command are ignored. No specifies that you can select other options.
Reset Forced? Y or N	Possible values are Y (yes) or N (no). Yes specifies that TOM resets the status of the object, based on the defined schedule, to ACTIVE or STOPPED if the object was in Forced ACTIVE status. TOM resumes control of the object according to its schedule. No specifies that TOM does not reset the Forced ACTIVE status.
Reset Status? Y or N	Possible values are Y (yes) or N (no). Yes specifies that TOM resets the status of the object, based on the defined schedule, to ACTIVE or STOPPED regardless of the current status of the object. For example this option can be used to clear LOCKED, BLOCKED or any FAILURE status. No specifies that TOM does not reset the status.
Reset Abend count? Y or N	Possible values are Y (yes) or N (no). Yes specifies that TOM resets the Abend count for the object. The Abend count represents the number of times the Abnormal Termination Event occurred since the last time the count was reset. No specifies that TOM does not reset the Abend count.
Reset Suspend Control? Y or N	Possible values are Y (yes) or N (no). Yes specifies that the object returns to TOM's control and SUSPEND is changed to ACTIVE state. TOM controls the objects' availability according to their schedule. No specifies that TOM does not take the object out of SUSPEND state.
Reset Start command count? Y or N	Possible values are Y (yes) or N (no). Yes specifies that TOM resets the Start command count for the object. Use this option when an object has reached the Max start count limit. No specifies that TOM does not reset the Start command count.

## Suspending or Activating an Object

The Suspend and Activate line commands have the following effects:

- Suspend temporarily takes an object out of TOM's control. TOM continues to monitor the status of the object but does not start or stop the object based on its schedule or definitions.
- Activate reinstates the suspended object within TOM's control. TOM resumes starting and stopping the object based on its schedule and definitions.

## Suspending Objects

To take an object temporarily out of TOM's control:

- Step 1** On the Started Task Overview panel, enter **U** (for Suspend) in the **LC** field next to the name of the object.

**Figure 6-10 Started Task Overview: Suspending an Object**

```

                Started Task Overview
                Row 1 to 6 of 6
Command ==>          Scroll ==> CSR

Primary Commands: ADD  CMDSHOW  CUST  V-evaluate

    Line Commands: B-browse D-delete  DD-delete block E-edit   R-repeat
                  : O-bounce P-stop   S-start   CH-change status
                  : L-lock   UL-unlock T-reset   U-suspend A-activate
                  : C-status K-block  UK-unblock M-move   J-propagate
                  : BT-browse real time

Definition Base: DEFAULT

LC Description          Except Control  Type STC
-----
>-----
_ test.objects.one      NO     SUSPEND  NORM  AAOCSMN1
_U CICSPROD.SYSA       NO     SUSPEND           AAOCSMN5
_                       NO     SUSPEND           AAOCSMN4
_                       NO     SUSPEND           AAOCSMN3
_                       NO     SUSPEND           AAOCSMN2
_                       NO     SUSPEND           AAOCSMN1
***** End of Data *****
    
```

The confirmation pop-up panel is displayed (Figure 6-11).

**Figure 6-11 Confirm Suspend Pop-Up Panel**

```

Command ==>

        Proceed?  N ( Y or N )

Press ENTER to continue or END to cancel
    
```

- Step 2** In the **Proceed** field, enter **Y**.

## Activating Objects

To return control of the object to TOM, you can perform any of the following actions:

- Issue the **A** (Activate) line command on the Started Task Overview panel next to the object name.

- Issue the **T** (Reset) line command on the Started Task Overview panel next to the object name and specify **Y** (yes) for the Reset Suspend Control option on the Confirm pop-up panel.
- Edit the **Control** field of the object's definition on the Start Task Definition panel and enter **ACTIVE**.

Use the Activate line command to return all objects to TOM's control:

**Step 1** On the Started Task Overview panel, enter **A** (for Activate) in the **LC** field next to the name of the object.

**Figure 6-12 Activating an Object**

```

Started Task Overview
Row 1 to 6 of 6
Command ==> Scroll ==> CSR

Primary Commands: ADD CMDSHOW CUST V-evaluate

Line Commands: B-browse D-delete DD-delete block E-edit R-repeat
                : O-bounce P-stop S-start CH-change status
                : L-lock UL-unlock T-reset U-suspend A-activate
                : C-status K-block UK-unblock M-move J-propagate
                : BT-browse real time

Definition Base: DEFAULT

LC Description          Except Control Type STC
-----
>-----
_ test.objects.one     NO    SUSPEND  NORM  AAOCSMN1
_A CICSProd.SYSA      NO    SUSPEND  AAOCSMN5
_                      NO    SUSPEND  AAOCSMN4
_                      NO    SUSPEND  AAOCSMN3
_                      NO    SUSPEND  AAOCSMN2
_                      NO    SUSPEND  AAOCSMN1
***** End of Data *****

```

The confirmation pop-up panel is displayed (Figure 6-13).

**Figure 6-13 Confirm Activate Pop-Up Panel**

```

Command ==>

Proceed? N ( Y or N )

Press ENTER to continue or END to cancel

```

**Step 2** In the **Proceed** field, enter **Y**.

## Changing the Status of an Object

**Note:** The CH line command affects only objects that do not have the STC name defined.

You can issue the CH line command to change the status of an object on a specific system.

For example, use the CH command against an object with a status of ACTIVE or ACTIVE-ERR to change the object status to STOPPED.

When this command is issued against an object with a status other than ACTIVE or ACTIVE-ERR, the status changes to ACTIVE.

To issue the CH line command:

- Step 1** On the Started Task Overview panel, enter **CH** in the **LC** field next to the name of the object.

**Figure 6-14 Changing the Status of an Object**

```

Started Task Overview
Command ==> Row 1 to 6 of 6
Scroll ==> CSR

Primary Commands: ADD  CMDSHOW  CUST  V-evaluate

Line Commands: B-browse D-delete DD-delete block E-edit R-repeat
               : O-bounce P-stop S-start CH-change status
               : L-lock UL-unlock T-reset U-suspend A-activate
               : C-status K-block UK-unblock M-move J-propagate
               : BT-browse real time

Definition Base: DEFAULT

LC Description          Except Control Type STC
-----
>-----
__ test.objects.one    NO    SUSPEND  NORM AAOCSMN1
CH CICSPROD.SYSA      NO    SUSPEND          AAOCSMN5
__                     NO    SUSPEND          AAOCSMN4
__                     NO    SUSPEND          AAOCSMN3
__                     NO    SUSPEND          AAOCSMN2
__                     NO    SUSPEND          AAOCSMN1
***** End of Data *****
    
```

The confirmation pop-up panel is displayed.

**Figure 6-15 Confirm Change Status Pop-Up Panel**

```

Command ==>

Proceed? N ( Y or N )

Press ENTER to continue or END to cancel
    
```

- Step 2** In the **Proceed** field, enter **Y**.

# Managing Definition Databases

Definition databases are where TOM stores objects, sets, calendar specifications, and user-created definitions. When TOM is implemented for the first time, a default definition database is created.

You might create more than one database for different situations, such as

- having a temporary database to use when making a large number of changes before moving the changes over into a production environment
- creating a disaster-recovery database to quickly establish a functioning system
- developing alternative databases that you can use for weekend activity or to meet other recurring system requirements

When more than one definition database has been created, you can use the Definition Base hyperlink on the Started Task Overview panel to activate a database or switch to another definition database. When you switch to another database, you can use the TOM applications to create, edit and save new objects to a new database while the system continues to use the current database.

Refer to the middle portion of Figure 6-2 on page 6-3 to see the Definition Base field.

**Figure 6-16 Definition Base Field**

```

Started Task Overview
Row 1 to 6 of 6
Command ==> Scroll ==> CSR
Primary Commands: ADD CMDSHOW CUST V-evaluate

Line Commands: B-browse D-delete DD-delete block E-edit R-repeat
                : O-bounce P-stop S-start CH-change status
                : L-lock UL-unlock T-reset U-suspend A-activate
                : C-status K-block UK-unblock M-move J-propagate
                : BT-browse real time

Definition Base: DEFAULT

LC Description          Except Control Type STC
-----
>-----
__ test.objects.one    NO    SUSPEND  NORM AAOCSMN1
__ CICSPROD.SYSA      NO    SUSPEND          AAOCSMN5
__                    NO    SUSPEND          AAOCSMN4
__                    NO    SUSPEND          AAOCSMN3
    
```

To activate a different database or switch to another definition database:

**Step 1** Perform one of the following:

- Select option **2** (Definitions) from the Administration Options Menu.
- Access the Started Task Overview panel, place the cursor on the **Definition Base** hyperlink, and then press **Enter**.

The Definition databases panel is displayed (Figure 6-17).

**Figure 6-17 Definition databases Panel**

```

Definition databases
Row 1 to 1 of 1
Command ==>          Scroll ==> CSR

Sysplex: BBPLEX01

Primary commands: CMDSHOW CUST SORT
Line commands: A-activate S-switch to

Database
LC Status Database Name
-----
>-----
___ ACTIVE  DEFAULT
___ INACTIVE BACKUP
___ INACTIVE WEEKEND PRODUCTION
___ INACTIVE WEEKDAY BACKUP
***** End of Data *****
    
```

**Step 2** Enter **A** (for Activate) in the **LC** field next to the database to which you want to change.

The **Database Status** column shows **ACTIVE** for the database that you selected.

Any changes that you make to objects for the active database take effect immediately.

For example, Figure 6-18 shows the panel when you select the WEEKDAY BACKUP database with the Activate line command:

**Figure 6-18 Changing Definition Databases Panel: Activate**

```

          Definition databases
          Row 1 to 1 of 1
Command ==>          Scroll ==> CSR

Sysplex: BBPLEX01

Primary commands: CMDSHOW  CUST  SORT
Line commands:  A-activate  S-switch to

Database
LC Status  Database Name
-----
>-----
__ INACTIVE DEFAULT
__ INACTIVE BACKUP
__ INACTIVE WEEKEND PRODUCTION
__ ACTIVE  WEEKDAY BACKUP
***** End of Data *****
    
```

**Step 3** To switch to another database, enter **S** in the **LC** field next to the target database.

You can add, delete, or otherwise make edits to the database, but TOM *continues* to use the ACTIVE definition database.

The **Database Status** column shows ACTIVE for the database that TOM is currently using. CURRENT appears next to the database that you selected with the S line command.

For example, Figure 6-19 shows the panel when you select the WEEKDAY BACKUP database with the Switch to line command:

**Figure 6-19 Changing Definition Databases Panel: Switch To**

```

          Definition databases
                Row 1 to 1 of 1
Command ==>                Scroll ==> CSR

Sysplex: BBPLEX01

Primary commands: CMDSHOW  CUST  SORT
Line commands: A-activate  S-switch to

Database
LC Status  Database Name
-----
>-----
__ ACTIVE  DEFAULT
__ INACTIVE BACKUP
__ INACTIVE WEEKEND PRODUCTION
__ CURRENT WEEKDAY BACKUP
***** End of Data *****

```

**Note:** On the Started Task Overview panel, the name of the ACTIVE database continues to appear in the **Definition Base** field.



---

---

# Chapter 7 Administrative Tasks

This chapter describes how TOM operates and describes administrative tasks that you might need to perform. This chapter includes the following topics:

How TOM Communicates with Other TOM Systems in a Sysplex . . . . .	7-2
Overview of the Registry . . . . .	7-6
Shutting Down a TOM System . . . . .	7-12
Shutting Down a Set . . . . .	7-18
Creating Multiple Definition Databases . . . . .	7-23
Reviewing Messages in the TOM Log . . . . .	7-26

## How TOM Communicates with Other TOM Systems in a Sysplex

You can have one TOM system running for each system within a sysplex. This configuration is referred to as a TOM-plex (refer to “TOM-Plex” on page 1-7). For all object dependencies, affinities, and statuses in the TOM-plex to stay updated and synchronized, all TOM systems in the TOM-plex must operate from identical data.

This data is maintained in the TOM Registry data set, which is specified by the REGISTRY DD name in BBSAMP member TOMALLOC. For each TOM system within a TOM-plex, when the system starts, it receives its own copy of this Registry data set through a process called *cloning*. For more information about the Registry, refer to “Overview of the Registry” on page 7-6.

### Overview of Cloning

When a TOM system is started, it automatically checks for any other active TOM systems on the same XCF group (see “Grouping Multiple TOM Systems in a Sysplex” on page 2-4 for more information about TOM systems and XCF groups).

The first TOM system that is started in a TOM-plex is *authoritative* which means that as additional TOM systems are started, they consider the object definitions and information from the already active TOM system’s registry as the definitive repository. When additional TOM systems are added to the TOM-plex, the new systems receive a copy of the Registry data set from the older TOM systems. This process is known as *cloning*.

**Warning!** You must ensure that the TOM system with the most updated object definitions is always started *before* any other TOM systems. As a result, all TOM systems that are started after this up-to-date TOM system will receive the latest information.

For example, suppose three TOM systems exist in a TOM-plex: TOM1, TOM2, and TOM3. These three systems are active and perfectly synchronized with each other until the TOM1 system is stopped.

During the time that TOM1 is not active, you can still add objects or make changes to systems TOM2 and TOM3. For this example, you add a new object named CICS.PROD3. Then TOM2 and TOM3 are stopped, TOM1 is restarted, and someone adds another object named CICS.PROD3 to TOM1. A discrepancy now exists between TOM1, TOM2, and TOM3 for the definition of an object named CICS.PROD3.

Eventually, TOM2 and TOM3 are restarted. To determine which of the CICS.PROD3 definitions to use for all TOM systems, TOM uses the following policy: The first TOM system that was started is the authoritative system. Additional TOM systems automatically receive a clone of the Registry data set from the TOM system(s) that is already started. In this example, the first TOM system that was started is TOM1. TOM2 and TOM3 will receive the CICS.PROD3 object definition from TOM1.

In summary, when TOM systems start after one has already started, they copy (clone) the information in the Registry from the TOM system(s) started earlier. This helps to keep all the TOM systems synchronized even when they are started at different times.

### **Scenario: One TOM System Starts before Other TOM Systems**

In another example, suppose a sysplex of three systems (SYS1, SYS2, and SYS3) where three TOM systems (TOM1, TOM2, and TOM3) are started. TOM1 is started first on SYS1.

When started on SYS1, TOM1 immediately begins managing all objects that are defined to run on SYS1. In the Registry, objects might also be defined to run on SYS2 and SYS3 (where TOM2 and TOM3 are not started yet). Until TOM2 and TOM3 are started, TOM1 displays a STOPPED status for the objects that are defined for SYS2 and SYS3, *even though* the objects might actually be up and running on SYS2 and SYS3.

Without being able to communicate with TOM2 and TOM3, TOM1 does not know the status of the objects on SYS2 and SYS3 and cannot manage those objects. When TOM2 and TOM3 are started, they receive clones of the Registry from TOM1, and the status of all objects within the sysplex can be accurately managed.

### **Scenario: A TOM System Loses Contact with Other TOM Systems in a TOM-plex**

Suppose that a sysplex contains three systems (SYS1, SYS2, and SYS3) and each system has a TOM system (TOM1, TOM2, and TOM3) started. If TOM1 and TOM2 lose contact with TOM3, they use the Cross-System Coupling (XCF) services to determine the status of SYS3, where TOM3 was running. There are two situations that might have caused the loss of contact:

- The TOM3 system has experienced a failure and stopped running, but SYS3 and all of its tasks are still active.

In this case, TOM1 and TOM2 do not change the state of the objects that are running on SYS3.

- TOM3 and SYS3 have both stopped running.

In this case, TOM1 and TOM2 set the status of all SYS3 objects to STOPPED. If the SYS3 objects had definitions in the **Valid Systems** field where they can run on systems other than SYS3, TOM1 and TOM2 can start those objects on other systems.

### Objects with a Status of BLOCKED

Objects that have a status of BLOCKED *always* retain that status until you issue the UNBLOCK command from the TOM panels or from an EXEC that uses the UNBLOCK function.

## Troubleshooting Cloning Failures

In another scenario, suppose the TOM2 system stops unexpectedly while cloning with TOM1.

During cloning, the authoritative TOM system—in this case, TOM1—can detect most problems that occur during cloning. In this scenario, the RY5000E error message is issued, and the TOM2 system abends with an abend code of 999.

If cloning to TOM3 is not interrupted, TOM3 receives a copy of the TOM1 Registry. If cloning to TOM3 is also interrupted, another RY5000E error message is issued, and TOM3 ends with abend code 999.

**Warning!** If cloning does not complete successfully, do not immediately restart the failed TOM system(s). In the case of a cloning failure, perform the following steps. Failure to follow the steps in this section can cause serious data loss.

**Step 1** Determine what caused the TOM systems to fail.

**Step 2** Perform one of the following actions:

- If you can correct the situation, restart the TOM systems and specify the VERIFY keyword on the OS parameter of the JCL in BBSAMP member TOMALLOC.
- If you cannot correct the failure, use the REPRO utility to copy the Registry data set of the authoritative TOM system to the Registry of the failed TOM system.

Registries *must* be the same physical size.

- Step 3** Restart the authoritative TOM system with the VERIFY keyword specified on the OS parameter of the JCL in BBSAMP member TOMALLOC.

## Backing Up the Registry

If you do not perform periodic backups of the Registry, you could lose all object definitions for a TOM-plex.

For example, suppose systems TOM1, TOM2, and TOM3 are stopped. The TOM1 system was a test system, and you decide to create a new (empty) Registry for TOM1 and start TOM1.

Because TOM1 was started first, it is the authoritative TOM system. If you start systems TOM2 and TOM3 after TOM1, the *empty* registry from TOM1 is cloned to TOM2 and TOM3, and the TOM-plex now has no object definitions at all.

**Tip:** BMC Software strongly recommends that you create periodic backups of the TOM Registry data set. You can do this by issuing the TOM console command REGISTRY DUMP DD (refer to “List of TOM Console Commands” on page 9-46).

In addition, BMC Software recommends that you back up the Registry before attempting to add a new TOM system to the TOM-plex. By creating this backup, you can prevent the loss of all TOM-plex data if errors occur when a new TOM system is added.

## Overview of the Registry

The Registry is a data space that is backed by a VSAM linear data set. As mentioned in “How TOM Communicates with Other TOM Systems in a Sysplex” on page 7-2, the TOM Registry contains all object definition information, provides quick access to the information, and maintains the information across system restarts.

Every TOM system that is running within the same sysplex needs to be constantly updated about the status of all objects in the TOM-plex. Therefore, every modification to the information in the Registry is communicated across the sysplex by way of XCF messages.

If one of these systems fails, the system can be automatically recovered when the system re-enters the TOM-plex (refer to “Overview of Cloning” on page 7-2).

## Handling Registry Errors

The Registry is backed up to disk every 30 seconds. In addition, each TOM system can automatically detect and save all object status changes with the auto-discovery process (refer to “How TOM Starts and Stops Objects” on page 1-10).

However, there are situations where the TOM Registry might become corrupted. For example, if the system has a power outage, or if the system is IPLed and the TOM systems were not shut down cleanly prior to the system IPL, internal pointers in the Registry can become corrupted. Registry corruption can cause subsequent abends and failures in TOM.

To help prevent TOM systems from failing, TOM provides some safeguards:

- When a TOM system starts, cloning begins immediately if another TOM system is available. During the cloning process, the TOM system can detect potential Registry corruption. If corruption is detected, VSAM checks the integrity of the Registry linear data set and VSAM gives you the option of running a VSAM VERIFY process against the data set.
- If a TOM system starts and another TOM system is not available (for example, if the entire sysplex has had a failure and ended), the TOM system can initiate its own VSAM VERIFY processing.

Although you have to wait for TOM to complete the VSAM VERIFY processing, this processing allows the Registry to be recovered as much as possible, and the TOM should not have any remaining internal problems.

**Note:** If you are aware that a TOM system abended or was shut down improperly, you can specify the VERIFY keyword on the OS parameter of the TOM Started Task procedure to ensure that the VERIFY processing occurs when TOM restarts.

After VERIFY processing is complete, you will receive a message if storage space is reclaimed. This message, means that the VERIFY processing has optimized the Registry's storage space usage.

## Restoring a Registry from a Backup

If the TOM Registry cannot be recovered from the VSAM VERIFY process, you must restore the Registry from a backup copy. "Backing Up the Registry" on page 7-5 describes the recommendations for when a Registry should be backed up.

To create a backup of the Registry:

**Step 1** using the following JCL:

---

```
//BAORAER JOB (RE62),UR.USERID,CLASS=F,MSGCLASS=R,NOTIFY=&SYSUID
//IDCAMS EXEC PGM=IDCAMS
//INPUT DD DISP=SHR,DSN=BAORAE.TM63.REGISTRY
//OUTPUT DD DISP=(,CATLG,DELETE),
// DCB=(RECFM=FB,LRECL=4096),SPACE=(CYL,20),
// DSN=BAORAE.TM63.REGISTRY.REPRO
//SYSPRINT DD SYSOUT=*
REPRO INFILE(INPUT) OUTFILE(OUTPUT)
```

---

Modify the jobholder settings for your user ID and site requirements.

**Warning!** It is important to ensure that INFILE and OUTFILE data set settings are the same size. If they are not the same size, you may experience an incomplete backup. If the OUTFILE is smaller than the INFILE, the backup process may complete but you might also encounter a user abend code U878.

## Monitoring Registry Size

If the Registry becomes full, TOM systems cannot function properly. You will receive a user abend code U878. However, you should not allocate a Registry that is larger than you need because that can impair the performance of certain of functions.

Therefore, it is important to pick the correct Registry size. When you are just beginning to create objects or you are converting several hundred objects from another application, BMC Software recommends a Registry size of 20 MB.

In addition, specify a percentage value on the WARNPCT() parameter in BBPARM member REGSTY00. BMC Software recommends using a value of 40. With this specification, when 40% (or more) of the Registry's data space is used, you receive a warning message. The message is issued every 30 seconds until usage drops below this threshold.

You can also use the following console command to see available storage, utilized storage, and other information:

```
F TOM,REGISTRY DISPLAY
```

Following is an example of the output from this command:

---

```
RY3050I Registry ALET is 01010042, origin at 00000000, pages: 3599
RY3060I Registry version is 01, originally created by TOMRE
RY3070I Registry was opened 1 and closed 0 times
RY3080I Registry was DIV saved 1 times
RY3082I The logically allocated storage is 11792 / 11792 bytes
RY3085I Free registry storage is 14725616 bytes
RY3090I Registry DIV save interval is about 60 seconds
RY3100I Last update timestamp is 13:25:24 - 2003.313 local time
```

---

After making substantial changes, you can also use the IDCAMS facility to make a backup of the Registry that can be restored in case of a failure.

## Maintaining Identical Registry Sizes

In general, all registries within a TOM-plex contain a similar amount of data, and you should ensure that all TOM systems in a TOM-plex have allocated Registries that are of a similar size.

For example, if you have a TOM system with a Registry that uses 50 cylinders of space and you attempt to start a new TOM system that has a 20-cylinder Registry, the cloning process (attempting to copy a 50-cylinder data set into a 20-cylinder data set) will fail.

Another benefit of having similar size Registries is that you can use the IDCAMS facility to copy the data set from one TOM system to another TOM system. When the Registry data sets are exactly the same size, one Registry can be substituted directly for another with IDCAMS. If one Registry is larger or smaller, internal corruption of the Registry might occur when using IDCAMS.

## Changing the Registry Size

After you begin working with a TOM system you might find that the size of the Registry that you allocated is not large enough. The following procedure describes how to change the size of the Registry when you are working in a TOM-plex (where other TOM systems are running and available):

- Step 1** Stop the TOM system where the Registry size needs to be changed.
- Step 2** Save that Registry under a new name.
- Step 3** Allocate a new Registry data set with the correct size.

**Step 4** Restart the TOM system.

The cloning process populates the new Registry with the data from the other TOM systems that are running in the TOM-plex.

**Step 5** Repeat this process for all the other TOM systems, ensuring that all the systems within the TOM-plex are running with the same Registry size.

You can also use the console command described in Table 9-22 on page 47 to off load the Registry into a sequential data set:

```
F TOM,REGISTRY DUMP DD( )
```

The data set can subsequently be used in conjunction with the PRIMEDD() statement in BBPARM member REGSTY00 to restore the contents of the Registry.

## TOM System Shutdown Considerations

While TOM systems can recover from an abnormal system termination, this process might take a long time to complete. To avoid this situation, issue the SHUTSYS line command and stop a TOM system before resetting the system. This way, TOM can properly save its Registry data sets, ensuring that all data structures are accurate. Refer to “Shutting Down a TOM System” on page 7-12 for more information about using the SHUTSYS line command.

Another situation is that TOM uses VSAM linear data sets and VSAM might require recovery if the system is turned off. Therefore, BMC Software recommends that you include a VSAM VERIFY for TOM VSAM data sets *before* the TOM execution step in the TOM startup JCL.

If you do not properly shut down a TOM system or an error occurs where you must reset the system, the following can occur:

- When the TOM restarts, it queries the Coupling Facility to find other TOM systems in the sysplex. If another TOM system is found, the Registry is cloned from the system that is already running.
- If no other TOM system is available when TOM restarts, TOM initiates its own VERIFY processing of the Registry, which might take some time to complete.

**Note:** Whenever possible, ensure that a TOM system undergoes a proper shutdown so that you can quickly restart the TOM system.

## Difference between Stopping a TOM System and Stopping its Host System

A TOM system runs as part of an OS/390 system. If (or when) you have to recycle the TOM system, it can be normally done without much impact on the other systems.

When a TOM system detects (through XCF) that another TOM system has dropped from the TOM-plex, the active TOM system continues to query the Coupling Facility about the status of the OS/390 system. As long as the OS/390 system is still active, TOM assumes that the status of objects on this LPAR has not changed, and the TOM system does not take any actions to prepare for a recovery situation. When the other TOM system is restarted and rejoins the TOM-plex, the status of the objects on both systems is synchronized through cloning.

Review the section “Overview of Cloning” on page 7-2 for a detailed description of how TOM clones the Registry from one TOM system to another TOM system. The order in which you restart TOM systems is very important, especially after you restart a sysplex. The first TOM system that is started contains the Registry that all other TOM systems receive as a clone copy. If that first TOM system’s Registry has errors, all TOM systems will receive copies of the errors.

## Shutting Down a TOM System

At times, the system where TOM is running will need to be stopped for maintenance or other tasks.

Although the primary purpose of TOM is to maintain the availability of the objects, there are times when the system where TOM is running will need to be stopped for maintenance or other tasks. System shut down can be an exacting and tedious process but TOM makes it simple and precise.

The following sections describe the tasks for shutting down a TOM system.

- From the TOM Managed Systems panel, use the **H** line command (for Shutsys) to try to stop all the objects under TOM's control.

TOM attempts to stop objects based on their dependency structure; TOM checks each object to see if another object depends on it. If the dependent object or objects are still active, the object is not stopped. If on the other hand, an object has no dependents (or those dependents are no longer active) TOM stops the object. For information about how TOM stops objects in sets, refer to "Shutting Down a Set" on page 7-18.

**Note:** Objects that are in SUSPEND mode are not stopped. They remain in SUSPEND mode unless you manually stop the objects.

All of the other objects defined to TOM are stopped.

For information about how to stop objects from the TOM Managed Systems panel, refer to "Stopping Objects on a Specific TOM System" on page 7-13.

- In the event that not all objects are stopped, you will need to determine why the object(s) did not stop and then stop the object(s) individually.

For information about how to handle objects that are not automatically stopped, refer to "Manually Shutting Down Remaining Objects" on page 7-15.

- When all the objects are stopped, issue the MVS stop command to stop the TOM address space.

For information about how to stop the TOM address space, refer to "Stopping the TOM Address Space" on page 7-17.

## Stopping Objects on a Specific TOM System

To shut down a TOM system:

**Step 1** On the TOM Primary Options Menu, enter **A** on the **Option** line.

The Administration Options Menu is displayed (Figure 7-1)

**Figure 7-1 TOM Administration Options Menu: Shutting Down a TOM System**

```

Administration Options Menu
Option ===> 2

1 Systems      - Systems mode operations
2 Definitions   - Define, display and manage definitions
3 Conversion    - Convert CSM V6 Repository
4 Layers       - Layer Management

X Exit         - Exit

```

**Step 2** On the **Option** line, enter **2** (for Definitions).

The TOM Managed Systems panel is displayed (Figure 7-2)

**Figure 7-2 TOM Managed Systems Panel: Shut Down a TOM System**

```

TOM Managed Systems                               Row 1 to 1 of 1
Command ===>                                     Scroll ===> CSR

Sysplex: BBPLEX01 - DEFAULT

Primary commands: CMDSHOW CUST SORT
Line commands: H-shutsys HL-shutsys/lock T-Tasks overview

  TOM      AO  System  IPL      IPL      OS      Clone
LC ID     SSID      Date    Time    Information  ID
-----
>-----
__ TOMKZ  KMZ2 SJSC    2003/10/16 00:09:55 z/OS 03.01.00    SC
***** End of Data *****

```

**Step 3** Enter **H** in the **LC** field next to the name of the TOM system that you want to shut down.

The confirmation pop-up panel is displayed (Figure 7-3).

**Figure 7-3 Please confirm shutsys Pop-Up Panel**

```
      Please confirm shutsys
Command ==>
      Proceed?  N      ( Y or N )
Target system  SJSD
Press ENTER to continue or CANCEL to exit
```

**Step 4** In the **Proceed?** field, enter **Y**.

The following message is displayed, indicating that the shutdown process has been triggered for all of the objects on the selected system:

```
OD9012I All TOM managed objects in shutdown
```

The objects are stopped and left in a status of LOCKED. When shutdown processing is complete, the following message is issued:

```
IC1003I TOM Shutdown Processing Complete
```

## How TOM Stops Objects Defined with a Shutdown Mode of QUICK

For objects that are managed by another scheduling product or that are manually started but monitored by TOM, you can specify shutdown mode of QUICK for the object (refer to Step 3 in “Defining How to Start an Object” on page 3-23).

When a QUICK shutdown mode is specified for an object, the following process occurs: During shutdown (of either the TOM system or of a set), TOM changes the status of the object immediately to STOPPED without

- performing any Pre-stop or Post-stop automation that might be associated with the object
- waiting for the object’s termination process to complete on the system

Specifying a QUICK shutdown mode for an object can significantly expedite shutdown processing or produce error messages during shutdown, depending on the situation.

For example, suppose an object named STCA depends on object named STCB. In this case, if STCA is defined with a shutdown mode of QUICK, the following problem could occur during shutdown:

1. TOM determines that STCB cannot be shut down (because STCA depends on it).
2. TOM determines that STCA can be stopped (because no other object depends on STCA).

Because a QUICK shutdown is specified for STCA, TOM immediately treats STCA as if its status is STOPPED. However, the stopping process is actually *still in progress* on the system and, therefore, STCA is not actually stopped.

3. TOM determines that object STCB can be stopped (because TOM considers STCA stopped and concludes that stopping STCB does not affect other objects).

However, if STCA has not yet actually stopped, you receive an error.

Therefore, you might have situations where an object does not shut down properly because an object has been defined with a shutdown mode of QUICK and another object was defined with a dependency on it. When this situation occurs, you must manually stop both objects (refer to “Manually Shutting Down Remaining Objects”).

## Manually Shutting Down Remaining Objects

Even when TOM reports that shutdown processing is complete, some objects might not be stopped.

Failure to stop an object can occur for any of the following reasons:

- An object is in SUSPEND mode. For these objects, you must determine whether these objects should be shut down and then manually stop the object from the console.
- An object was defined with a dependency and that object has not been shut down.

- Other errors were encountered during the shutdown process, such as one of the following conditions:
  - A Verify Stop condition was defined for an object, and a write-to-operator (WTOR) message was issued, but nobody replied to it.
  - A non-zero return code was returned from a pre-stop EXEC that is associated with an object.
  - An invalid shutdown validation message, command, or ALERT ID was specified in the object’s definition.
  - Invalid or unresolvable variables were specified in the Stop command that was defined for the object.
  - The Stop command timeout value was exceeded.

To determine whether any objects were not stopped during the shutdown process:

- Step 1** Display the Started Task Overview panel from the TOM Primary Options Menu.
- Step 2** Look for objects that display YES in the **Except** column (see Figure 7-4 for an example).

**Figure 7-4 Verifying That Objects Are Stopped**

Started Task Overview		Row 13 to 16 of 50	
Command ==>		Scroll ==> CSR	
Primary Commands: ADD CMDSHOW CUST EVAL SORT			
Line Commands: A-activate U-suspend O-bounce P-stop S-start			
: B-browse E-edit D-delete DD-delete block			
: BT-browse real-time C-status CH-change status			
: L-lock UL-unlock K-block UK-unblock			
: J-propagate M-move R-repeat T-reset			
Definition Base: DEFAULT			
LC Description	STC	Active Except System	Status
>-----			
__ AAOKMZ87 - CICS Object	AAOKMZ01	<b>YES</b>	OTHER
__ AAOKMZ86 - CICS2 Object	AAOKMZ01	<b>YES</b>	OTHER
__ AAOKMZ85 - Test1 Object	AAOKMZ01	NO	OTHER
__ AAOKMZ84 - Test2 Object	AAOKMZ01	NO	OTHER

- Step 3** After identifying and correcting the exception, enter **T** in the **LC** field next to each object to take the object out of Exception mode.
- Step 4** Enter **P** in the **LC** field next to each object to manually stop the object.

## Stopping the TOM Address Space

When all of the objects are no longer active, you can stop the TOM address space with an MVS stop command. When the TOM address space is restarted, all of the objects are restarted.

## Shutting Down a Set

You can use TOM to group objects into sets and specify dependency structures where an object is dependent on a set. Refer to “Creating and Managing Sets” on page 4-1 for more information about sets and their uses.

The following sections describe the tasks for shutting down a set.

- From the Started Task Sets panel, use the **H** line command (for Shutset) to try to stop all the objects with a set.

TOM attempts to stop objects based on their dependency structure; TOM checks each object to see if another object depends on it. If the dependent object or objects are still active, the object is not stopped. If on the other hand, an object has no dependents (or those dependents are no longer active) TOM stops the objects within the set.

**Note:** Objects that are in SUSPEND mode are not stopped. They remain in SUSPEND mode unless you manually stop the objects.

All of the other objects defined in the set are stopped.

For information about how to stop objects from the Started Task Sets panel, refer to “Stopping Objects within a Set” on page 7-19.

- In the event that not all objects are stopped, you will need to determine why the object(s) did not stop and then stop the object(s) individually.

For information about how to handle objects that are not automatically stopped, refer to “Manually Shutting Down Remaining Objects” on page 7-15.

## Stopping Objects within a Set

To stop all the objects within a set, use the SHUTSET line command from the Started Task Sets panel:

To stop all objects within a set:

**Step 1** On the TOM Primary Options Menu, enter **2** on the **Option** line.

The Started Task Sets panel is displayed (Figure 7-5)

**Figure 7-5 Started Task Sets Panel**

Started Task Sets		Row 2 to 8 of 9
Command ==>		Scroll ==> CSR
Primary commands: ADD CMDSHOW CUST		
Line commands: A-activate U-suspend O-bounce P-stop S-start		
: B-browse E-edit D-delete DD-delete block		
: L-lock UL-unlock K-block UK-unblock H-shutset		
: C-status J-propagate M-move R-repeat T-reset		
LC Description	Members Number	Set Status
-----		
>-----		
__ Mirror Test 2_____	3	ACTIVE
__ Mirror Test_____	0	OTHER
__ Copy of Test Set 01_____	0	OTHER
__ Test Set 05_____	10	STOPPED
__ Test Set 04_____	4	STOPPED
__ Test Set 03_____	2	STOPPED

**Step 2** Enter **H** in the LC field next to the name of the set that you want to shut down.

The confirmation pop-up panel is displayed (Figure 7-6).

**Figure 7-6 Confirm Shutset Pop-up Panel**

<p>Please confirm shutset</p> <p>Command ==&gt;</p> <p>Proceed with shutset? N ( Y or N )</p> <p>Press ENTER to continue or CANCEL to exit</p>
--

**Note:** When you issue the H line command (for SHUTSET), any automation that is defined for the objects within the set (such as Pre-stop or Post-stop EXECs) is scheduled as part of shutting down the set.

**Step 3** In the **Proceed with shutset?** field, enter **Y**.

The following message is displayed, indicating that the shutdown process has been triggered for all objects in the set:

```
OD7501D Set shutdown for setID commencing
```

When each object is ready to be stopped, the following messages are issued:

```
OD2709I Stop initiated for object AAOKMZ23  
OD9011I Shutting down AAOKMZ23
```

The objects are stopped and left in a status of **LOCKED**. When processing is complete, the following message is issued:

```
OD9013I All objects for set setID stopped
```

When all objects in the set have stopped, the set status is set to **STOPPED**. If objects within the set did not stop properly, the set's state is set to **OTHER**.

**Note:** If messages OD2709I and OD9011I are not issued for an object in the set, it is likely that the object has a dependent that is active. SHUTSET does not issue stop commands to stop the dependent object. You must manually find and stop the dependent objects. When the dependent object has stopped then TOM triggers the stop of the appropriate object in the set.

Refer to “How TOM Stops Objects Defined with a Shutdown Mode of QUICK” on page 7-14 for information about what might happen during the shutdown of a set when objects are defined with a shutdown mode of QUICK.

## Manually Shutting Down Remaining Objects

Even when TOM has reported that set shutdown processing is complete, some objects might not be stopped.

Failure to stop an object can occur for any of the following reasons:

- An object within the set has a dependency on another object that is not in the set.
- An object was defined with a dependency on this set and that object has not been shut down.
- An object is in SUSPEND mode. For these objects, you must determine whether these objects should be shut down and then manually stop the object from the console.
- Errors were encountered during the shutdown process, such as one of the following conditions:
  - A Verify Stop condition was defined for an object, and a write-to-operator (WTOR) message was issued, but nobody replied to it.
  - A non-zero return code was returned from a Pre-stop EXEC that is associated with an object.
  - An invalid shutdown validation message, command, or ALERT ID was specified in the object's definition.
  - Invalid or unresolvable variables were specified in the Stop command that was defined for the object.
  - The Stop command timeout value was exceeded.
  - An object's dependents were not stopped.

To determine whether any objects were not stopped during the shutdown process:

- Step 1** Display the Started Task Overview panel from the TOM Primary Options Menu.

**Step 2** Look for objects that display **YES** in the **Except** column (see Figure 7-4 for an example).

**Figure 7-7** Verifying That Objects Are Stopped

```

Started Task Overview                               Row 13 to 16 of 50
Command ==>>                                       Scroll ==>> CSR

Primary Commands: ADD  CMDSHOW  CUST  EVAL  SORT

Line Commands: A-activate  U-suspend  O-bounce  P-stop    S-start
                : B-browse   E-edit     D-delete  DD-delete  block
                : BT-browse  real-time C-status  CH-change  status
                : L-lock     UL-unlock K-block   UK-unblock
                : J-propagate M-move     R-repeat  T-reset

Definition Base: DEFAULT

LC Description                                     STC      Active
                                                Except System  Status
-----
>-----
__ AAOKMZ87 - CICS Object                         AAOKMZ01 YES          OTHER
__ AAOKMZ86 - CICS2 Object                        AAOKMZ01 YES          OTHER
__ AAOKMZ85 - Test1 Object                        AAOKMZ01 NO           OTHER
    
```

**Step 3** After identifying and correcting the exception, enter **T** in the **LC** field next to each object to take the object out of Exception mode.

**Step 4** Enter **P** in the **LC** field next to each object to manually stop the object.

## Creating Multiple Definition Databases

A Definition Database is a part of the Registry where TOM collects and organizes information about objects (and their definitions), sets, and calendar definitions.

You might create multiple Definition Databases for different situations, such as

- having a test database that you can use temporarily when making a large number of changes
- creating a disaster recovery database that you can use to quickly establish a functioning system
- developing alternative databases that you can use for weekend activity or to meet other recurring system requirements

To create additional Definition Databases:

**Step 1** On the TOM Primary Options Menu (Figure 7-8), enter **A** on the **Option** line.

**Figure 7-8** TOM Primary Options Menu

```

BMC Software                TOM Primary Options Menu

Option  ==> A

1 Started Task  - Define, display and manage objects      User ID: BAOMXY1
2 Sets         - Define, display and manage sets          Server Id: TOMN1
3 Status       - Define, display and manage systems        Release: 1.1D
4 Calendar     - Define, display and manage calendars      System: SJSD

A Administration - System Administration
M Messages      - List messages
L Log           - Display Log
X Exit          - Exit

                                     Copyright BMC Software, Inc. 2003

```

The Administration Options Menu is displayed (Figure 7-9).

**Figure 7-9 TOM Administration Options Menu**

```

Administration Options Menu
Option ==> 2

1 Systems      - Systems mode operations
2 Definitions   - Define, display and manage definitions
3 Conversion    - Convert CSM V6 Repository
4 Layers        - Layer Management

X Exit         - Exit
    
```

**Step 2** On the **Option** line, enter **2**.

The Definition Databases panel is displayed (Figure 7-10).

**Figure 7-10 Definition Databases Panel**

```

Definition Databases                               Row 1 to 1 of 1
Command ==>                                       Scroll ==> CSR

Sysplex: BBPLEX01

Primary commands: ADD CMDSHOW CUST SORT
Line commands: D-delete R-repeat
                A-activate S-switch to

Database
LC Status Database Name Description
-----
__ ACTIVE  DEFAULT
***** End of Data *****
    
```

The active Definition Database is displayed.

**Step 3** To add a new database, enter **ADD** on the **Command** line.

The New database name pop-up panel is displayed (Figure 7-11).

**Figure 7-11 New database name Pop-Up Panel**

```

New database name
Command ==>

=>

Description:

Press END to continue or CANCEL to exit
    
```

- Step 4** At the => prompt, enter the 1- to 64-character name of a new Definition Database.
- Step 5** (*optional*) In the **Description** field, enter a 1- to 40-character description of the new database.
- Step 6** Press **PF3** to save.

The Definition Databases panel is redisplayed, showing the new database with a status of CURRENT. Refer to Figure 7-12.

**Figure 7-12 Definition Databases Panel: New Database Added**

```

                                     Definition Databases
                                     Row 1 to 1 of 1
Command ==>                               Scroll ==> CSR

Sysplex: BBPLEX01

Primary commands: ADD  CMDSHOW  CUST  SORT
Line commands:  D-delete  R-repeat
                A-activate  S-switch to

Database
LC Status  Database Name          Description
-----
__ CURRENT  BACKUPDATABASE         Use to back up original database _____
__ ACTIVE   DEFAULT
***** End of Data *****
    
```

- Step 7** To create objects, sets, calendar entries, and other definitions for a new database, enter **S** (for Switch) in the **LC** field next to the name of the new database.

The short message Database switched appears in the upper right corner.

You can now use the TOM applications to create, edit, and delete objects for the new database. However, TOM is still managing the sysplex based on the definitions in the active database.

- Step 8** To change the current database to a different database that you have created, enter **A** (for Activate) in the **LC** field next to the name of the database.

TOM immediately begins to use the object definitions in the activated database.

## Reviewing Messages in the TOM Log

The TOM messages log collects and displays messages from the systems.

**Note:** You can view the messages in the TOM Message Facility currently only by selecting the **M** (for messages) option from the TOM Primary Option Menu.

To review the messages that are captured by the TOM log:

- Step 1** On the TOM Primary Options Menu (Figure 7-13), enter **L** on the **Option** line.

**Figure 7-13** TOM Primary Options Menu

```

BMC Software                TOM Primary Options Menu

Option ==>

1 Started Task    - Define, display and manage objects      User ID: BAOMXY1
2 Sets           - Define, display and manage sets           Server Id: TOMN1
3 Status         - Define, display and manage systems          Release: 1.1D
4 Calendar       - Define, display and manage calendars        System: SJSD

A Administration - System Administration
M Messages       - List messages
L Log            - Display Log

X Exit           - Exit

                                Copyright BMC Software, Inc. 2003
    
```

The TOM log is displayed (Figure 7-14).

**Figure 7-14** TOM Log

```

TOM Log Display
Command ==> Scroll ==> CSR
Line 9118 of 9151 Log# 1
>-----
13:13:29 2003/09/24 TOMN1 RY3070I Registry was opened 1 and closed 0 times
13:13:29 2003/09/24 TOMN1 RY3080I Registry was DIV saved 105 times
13:13:29 2003/09/24 TOMN1 RY3082I The logically allocated storage is 28976
13:13:29 2003/09/24 TOMN1 RY3085I Free registry storage is 14708432 bytes
13:13:29 2003/09/24 TOMN1 RY3090I Registry DIV save interval is about 60 s
13:13:29 2003/09/24 TOMN1 RY3100I Last update timestamp is 13:12:50 - 2003
13:13:29 2003/09/24 TOMN1 RY3200D Dumping registry now to DD REGDUMP
13:13:29 2003/09/24 TOMN1 RY3241I 113 records written for dump
13:18:02 2003/09/24 TOMN1 CX1021W Could not open TOMPARM DD statement, mem
13:18:02 2003/09/24 TOMN1 TX3000I TOM (Total Object Manager) TOMN1 version
13:18:02 2003/09/24 TOMN1 TX3001I TOM currently manages:
13:18:02 2003/09/24 TOMN1 TX3002I Address spaces (TSUs, STCs and jobs)
13:18:02 2003/09/24 TOMN1 TF2001D TOM will use XCF group BMCTOMNS
13:18:05 2003/09/24 TOMN1 RY3007I Definition Base DEFAULT is in use
13:18:05 2003/09/24 TOMN1 TX0110I TOM TOMN1 version 1.1.0m initialized
13:52:41 2003/09/24 TOMN1 CX1021W Could not open TOMPARM DD statement, mem
13:52:41 2003/09/24 TOMN1 TX3000I TOM (Total Object Manager) TOMN1 version
13:52:41 2003/09/24 TOMN1 TX3001I TOM currently manages:
13:52:41 2003/09/24 TOMN1 TX3002I Address spaces (TSUs, STCs and jobs)
13:52:41 2003/09/24 TOMN1 TF2001D TOM will use XCF group BMCTOMNS
13:52:49 2003/09/24 TOMN1 RY5200I Verifying registry - please wait
13:53:06 2003/09/24 TOMN1 RY5201I Registry verify complete - 0 errors four
13:53:06 2003/09/24 TOMN1 RY3007I Definition Base DEFAULT is in use
13:53:06 2003/09/24 TOMN1 TX0110I TOM TOMN1 version 1.1.0m initialized
16:46:30 2003/09/24 TOMN1 CX1021W Could not open TOMPARM DD statement, mem

```

**Step 2** Refer to the online Help for information about scrolling through this log and using the Find command to find specific messages.



---

---

# Chapter 8 Managing Objects across Sysplexes with Layering

This chapter includes the following topics:

Overview of Layering . . . . .	8-2
Layering Tasks . . . . .	8-5
Getting Started with Layering. . . . .	8-6
When to Create Layer Objects . . . . .	8-12
Creating Layer Sets. . . . .	8-21
Adding Layer Objects and Layer Sets to a System Layer . . . . .	8-25
Adding System Layers to a Sysplex Layer. . . . .	8-29
Attaching a System Layer to a System within the Same Sysplex . . . . .	8-32
Attaching a System Layer to a System in a Remote Sysplex . . . . .	8-35
Attaching a Sysplex Layer to a Sysplex . . . . .	8-38
Example Scenarios . . . . .	8-41
Updating Objects and Sets Created from Layering . . . . .	8-46
Copying the Registry . . . . .	8-47

## Overview of Layering

When you are defining objects within a single sysplex, TOM provides facilities such as Modeling that you can use to create identical or similar objects from a single Model definition.

However, you can use the features in the Layering option to define and manage many similar objects that are running on multiple sysplexes. Layering creates Layer objects and Layer sets that you can copy to other sysplexes. This process minimizes the number of times that you have to define or update objects and sets (and their associated definitions).

For example, almost all OS/390 systems need to have JES, VTAM, TSO, and other objects defined on each system in each sysplex. In addition, each OS/390 system might have multiple CICS and WebSphere objects defined. If you have 30 OS/390 systems spread among 12 sysplexes, defining and managing objects for all of these systems is a big task. To change the definition of the VTAM object, you can make the change in the Layer object and easily copy the updates to all other objects, on multiple sysplexes, which are derived from the Layer object.

## Layering Terms and Concepts

Table 8-1 describes important terms and concepts that are used in Layering.

**Table 8-1 Layering Terms and Concepts**

Term	Definition
Layer object	<p>A special type of object whose own state is not managed by TOM but the Layer object definitions, dependencies and schedule will be replicated (known as propagating) through the Layering application to other systems or sysplexes.</p> <p>When a Layer object is defined, TOM does not start or stop or manage the Layer object's state. Instead, the Layer object's definitions create non-Layer objects on another system (or on all the systems) in a sysplex.</p> <p>Layer objects are created on the Started Task Definition panel by specifying a Pattern name for a Layer object in the <b>Pattern</b> field in addition to a name in the <b>Name</b> field. The naming convention of the Pattern name uses a combination of mandatory variables and (optional) user-specified literal characters.</p> <p>The Pattern name is used by TOM during replication to create the names of the non-Layer objects that are created from the Layer object.</p> <p>For information about the naming convention for Layer objects, refer to "Layer Object Naming Conventions" on page 8-12.</p>
Layer set	<p>A special type of set that contains only Layer objects and Layer sets.</p> <p>When a Layer set is defined, TOM does not start or stop or manage the Layer set's state. Instead, Layer sets create non-Layer sets of objects on another system (or on all the systems) in a sysplex.</p> <p>Layer sets are created on the Started Task Sets panel with the ADD primary command. On the New Name pop-up panel, specify a name for the set and Layer Pattern set name in the <b>Layer Pattern</b> field in addition to a set name. The naming convention of the Layer Pattern name uses a combination of mandatory variables and (optional) user-specified literal characters.</p> <p>The Layer Pattern name is used by TOM during replication to create the names of the non-Layer sets that are created from the Layer set.</p> <p>For information about the naming convention for Layer sets, refer to "Layer Set Naming Conventions" on page 8-13.</p>

**Table 8-1 Layering Terms and Concepts**

Term	Definition
System layers	<p>Contains Layer objects and Layer sets.</p> <p>A system layer allows you to create a grouping of both Layer objects and Layer sets which can then be applied as a single entity to a sysplex (or another system). This collection prevents you from having to add each Layer object or Layer set separately to a sysplex.</p> <p>It also allows you to logically group together similar Layer objects. For example, during your Layer object planning phase, you may decide to group all the Layer objects that represent CICS objects into one system layer and all the WebSphere objects into a separate system layer. This will make it easier to maintain similar Layer objects and sets over time.</p> <p>Another instance of when you might create a system layer is if you want to group together a collection of the basic objects that are common to every system across multiple sysplexes. This system layer is used to quickly and easily populate new systems added to a sysplex and across to other sysplexes.</p>
Sysplex Layers	<p>Contains System Layers.</p> <p>By having well organized system layers, you can have more options when managing the sysplex layer. For example, you might make an update to all the Layer objects that represent CICS objects by collecting all those objects in a single system layer. Then, with these objects are collected in a system layer, you can include only that system layer in a sysplex layer and update all the CICS objects in a sysplex at once.</p> <p>In addition, you can include multiple system layers in a sysplex layer and by applying the sysplex layer, update many types of objects and sets with a single action.</p>
Attach	<p>Is the name of the process where after all the Layer objects and sets are created, collected in system layers and sysplex layers, and then activated on a sysplex.</p> <p>During an attach, the variables in the Layer objects's names get translated and become the names of non-Layer object on the "attached to" sysplex systems. All the object definitions, schedules, and dependencies are replicated to the new sysplex.</p>
Propagate	<p>Is the name of the action where changes to Layer objects and Layer sets are replicated to a new sysplex.</p> <p>Is also the name of a line command on the Started Task Definition panel and Started Task Sets panel where Layer objects and Layer sets definitions can be replicated from the these panels.</p>
Layer ID	<p>To make Layer object and Layer set definitions as flexible as possible, the Layering process includes assigning a Layer ID to each system within a sysplex.</p> <p>The format of the Layer ID is: <b>&amp;SYSnnnn</b> where nnnn is a number from 1 to 9999.</p> <p>The Layer ID is then used in Layer object definitions as a substitute for the actual system names such as in the Layer object's <b>Valid System</b> definition list or in the <b>On System</b> field in the Dependency Property definition.</p>
Clone ID	<p>The Clone ID is the 2-character MVS Clone ID of each system.</p> <p>Specifying a clone ID for each system in a sysplex is required so that the names in the Layer object Pattern name and the layer Set pattern name can be properly resolved.</p>

# Layering Tasks

Table 8-2 describes the recommended order of steps to complete the processes for Layering.

**Table 8-2 Overview to Layering Tasks**

Step	Task Description	Reference Page
1	Enter the names of sysplexes	"Entering Sysplex Names" on page 8-6
2	Enter system names and assign Layer IDs to the systems in each sysplex	"Entering System Names and Layer IDs for the Sysplex" on page 8-9
3	Create Layer objects	"Creating a Layer Object" on page 8-14
4	Create Layer sets	"Creating Layer Sets" on page 8-21
5	Add Layer objects and Layer sets to a system layer	"Adding Layer Objects and Layer Sets to a System Layer" on page 8-25
6	Add system layers to a sysplex layer	"Adding System Layers to a Sysplex Layer" on page 8-29
7	Attach system layers to systems in sysplexes	"Attaching a System Layer to a System within the Same Sysplex" on page 8-32 "Attaching a System Layer to a System in a Remote Sysplex" on page 8-35
8	Attach a sysplex layer to a sysplex	"Attaching a Sysplex Layer to a Sysplex" on page 8-38
9	Review how object and set names are resolved when created from Layer objects and Layer sets	"Example Scenarios" on page 8-41
10	Copy the Registry	"Copying the Registry" on page 8-47
11	Update objects and sets that were created from layering	"Updating Objects and Sets Created from Layering" on page 8-46

## Getting Started with Layering

When you determine that you want to use Layering to be able to replicate objects and sets to many systems running in a large number of sysplexes, the first three tasks you need to perform are as follows:

- Enter the names of the sysplexes.
- Enter the names of the systems within the sysplexes.
- Assign layer IDs to each system in each sysplex.

The following sections describe these processes.

### Entering Sysplex Names

By adding the names of the sysplexes and each system within each sysplex to the TOM Sysplex Matrix application, you can identify the number of sysplexes and systems within each sysplex.

Some systems require identical or similar configurations of objects. For example, each sysplex might have a Production system, a Test system, and a weekend system that runs backup jobs, and each of these systems has the same configuration requirements in each sysplex.

When you assign Layer IDs to these systems, you might want to use the same number to represent a type of system across sysplexes. For example, all of the weekend backup systems might have the same Layer ID of 2244, regardless of the sysplex to which they belong.

**Note:** All sysplex and system names for remote sysplexes must be added on the System Matrix panel. Information appears automatically only if you add the local sysplex.

To enter sysplex names:

- Step 1** On the TOM Primary Options Menu, enter **A** (for Administration) on the **Option** line (as shown in Figure 8-1).

**Figure 8-1 TOM Primary Options Menu**

```

BMC Software                TOM Primary Options Menu

Option  ==> A

1 Started Task  - Define, display and manage objects      User ID: BAOMXY1
2 Sets         - Define, display and manage sets          Server Id: TOMN1
3 Status       - Define, display and manage systems       Release: 1.1D
4 Calendar     - Define, display and manage calendars     System: SJSD

A Administration - System Administration
M Messages     - List messages
L Log          - Display Log

X Exit         - Exit

Copyright BMC Software, Inc. 2003

```

The Administration Options Menu is displayed (Figure 8-2).

**Figure 8-2 Administration Options Menu**

```

Administration Options Menu

Option  ==> 4

1 Systems      - Systems mode operations
2 Definitions  - Define, display and manage definitions
3 Conversion   - Convert CSM V6 Repository
4 Layers       - Layer Management

X Exit        - Exit

```

- Step 2** On the **Option** line, enter **4** (for Layers).

The Layer Management panel is displayed (Figure 8-3).

**Figure 8-3 Layer Management Panel**

```

Layer Management

Option  ==>

1 Sysplex Matrix - Manage system definitions
2 System Layers  - Define object groups
3 Sysplex Layers - Define system groups

X Exit         - Exit

```

**Step 3** On the **Option** line, enter **1** (for Sysplex Matrix).

The Sysplex Matrix panel is displayed (Figure 8-4).

**Figure 8-4 Sysplex Matrix Panel**

```

                                Sysplex Matrix                                Row 1 to 4 of 4
Command ==>

Primary Commands: ADD  CMDSHOW

      Line Commands: D-delete Sysplex definition
                    M-manage systems in Sysplex
                    AT-attach sysplex layer to Sysplex

      Sysplex  Number  Systems
LC Name      Systems Active  Description
-----
>-----
__ BBPLEX01      6      6
__ BBPLEX03      2      0 Production System
__ BBPLEX02      2      0 Weekend Backups
***** End of Data *****

```

**Note:** The first time you display the Sysplex Matrix panel, the panel shows no sysplexes because sysplexes have not been entered yet.

**Step 4** To add a sysplex, enter **ADD** on the **Command** line.

The Add Sysplex Definition pop-up panel is displayed (Figure 8-5).

**Figure 8-5 Add Sysplex Definition Pop-up Panel**

```

                                Add Sysplex definition
Command ==>

      Name: BBPLEX04
      Description: Remote system 4

Press ENTER to continue or CANCEL to exit

```

**Step 5** In the **Name** field, enter a 1- to 8-character sysplex name.

**Step 6** (*optional*) In the **Description** field, enter a 1- to 40-character description of the sysplex.

For this example, the sysplex name is BBPLEX04 with a description of *Remote system 4*.

**Step 7** Press **Enter** to save.

The Sysplex Matrix panel is redisplayed, and the new sysplex name is added (Figure 8-6).

**Figure 8-6 Sysplex Matrix Panel (New Sysplex Added)**

```

                                Sysplex Matrix                                Row 1 to 4 of 4
Command ==>

Primary Commands: ADD  CMDSHOW

    Line Commands: D-delete Sysplex definition
                  M-manage systems in Sysplex
                  AT-attach sysplex layer to Sysplex

    Sysplex  Number  Systems
    LC Name   Systems Active  Description
-----
>-----
__ BBPLEX01      6      6
__ BBPLEX04      0      0 Remote system 4
__ BBPLEX03      2      0 Production System
__ BBPLEX02      2      0 Weekend Backups
***** End of Data *****

```

**Step 8** Repeat Step 4 through Step 6 for each sysplex name that you need to add.

## Entering System Names and Layer IDs for the Sysplex

After the sysplex name is entered, you need to enter the names for all the systems within the sysplex and enter a 1- to 4-numeric Layer ID to each system where values can be 1 to 9999.

Assigning Layer IDs to all systems in the sysplexes simplifies the process of creating Layer object definitions. The Layer ID represents a virtual system name that is resolved during the attachment and propagation processes.

When you need to use the Layer ID in the definition for the Valid System or Dependency property, enter the Layer ID in the following format:

```
&SYSnnnn
```

nnnn represents the 1 to 4-character Layer ID of the system you defined in the Systems Matrix application.

Refer to “Specifying Valid Systems for a Layer Object” on page 8-18 and “Specifying Dependency Properties for a Layer Object” on page 8-20 for examples.

To enter system names and assign Layer IDs:

- Step 1** On the Sysplex Matrix panel, enter **M** (for manage) in the **LC** field next to the name of the sysplex for which you will specify systems (as shown in Figure 8-7).

**Figure 8-7 Entering System Names**

```

                                     Sysplex Matrix                               Row 1 to 4 of 4
Command ==>

Primary Commands: ADD  CMDSHOW

Line Commands: D-delete Sysplex definition
                M-manage systems in Sysplex
                AT-attach sysplex layer to Sysplex

Sysplex  Number  Systems
LC Name   Systems Active  Description
-----
>-----
__ BBPLEX01      6      6
_M BBPLEX04      0      0 Remote system 4
__ BBPLEX03      2      0 Production System
__ BBPLEX02      2      0 Weekend Backups
***** End of Data *****

```

The System Definitions panel is displayed (Figure 8-8).

**Figure 8-8 System Definitions Panel**

```

                                     System Definitions                               Row 0 to 0 of 0
Command ==>

Primary Commands: ADD  CMDSHOW
Line Commands: D-delete AT-attach system layer

Systems in Sysplex BBPLEX04    ( remote )

System  Layer
LC Name ID   Description                Status  Clone
-----
***** End of Data *****

```

The **Systems in Sysplex** display-only field shows the name of the selected sysplex. In this example, the sysplex name is **BBPLEX04**.

- Step 2** On the **Command** line, enter **ADD** to add a system name to the sysplex.

The Add Systems Definition panel is displayed (Figure 8-9).

**Figure 8-9 Add Systems Definition Panel**

```

Add Systems Definition
Command ==>

      Name:
      Layer ID:
      Clone ID:
      Description:

Press ENTER to continue or CANCEL to exit

```

**Step 3** Complete the fields of the Add Systems Definition panel:

- 3.A** In the **Name** field, enter the 1- to 8-character name of the system.
- 3.B** In the **Layer ID** field, enter the 1- to 9999-digit Layer ID that you choose for this system.
- 3.C** In the **Clone ID** field, enter the 1- to 2-character MVS clone ID.
- 3.D** (*optional*) In the **Description** field, enter a 1- to 40-character description of the system.

**Step 4** Press **Enter** to save.

The new system is added.

**Figure 8-10 System Definitions Panel**

```

System Definitions                               Row 0 to 0 of 0
Command ==>

Primary Commands: ADD  CMDSHOW
Line Commands: D-delete  AT-attach system layer

Systems in Sysplex BBPLEX04   ( remote )

      System   Layer
LC Name   ID   Description                Status   Clone
-----
SYSTEMD  2244  Testing system                N/A     XY
***** End of Data *****

```

**Step 5** Repeat Step 2 through Step 4 to enter the names of all systems in the sysplex.

**Note:** The **Status** column of the System Definitions panel shows N/A because this version of TOM does not support connections between sysplexes, so it cannot obtain the system status of remote sysplexes.

## When to Create Layer Objects

After all of the sysplexes and system names have been defined, you can start to create Layer objects. Some good candidates to be defined as Layer objects are VTAM, JES, TSO, and TCP/IP systems. In fact, any object that can be found on every system or is implemented many times on a single system (such as CICS or WebSphere) are also good candidates because by creating a Layer object, you can avoid defining each of the objects individually for each system in each sysplex.

### Layer Object Naming Conventions

Use one of the following formats to name a Layer object:

**&SYSPLEX.&SYSNAME.userSpecifiedName**

**&SYSPLEX.&SYSCLONE.userSpecifiedName**

This name is also referred to as a as a Layer object *pattern*.

**Note:** The Layer object pattern must contain at least 3 nodes and can be 64 characters, as described in the following table:

Node	Required?	Description
&SYSPLEX	Yes	resolves to the sysplex name  This name is required for every Layer object or Layer set.
&SYSNAME	No. However, if not used, you must specify &SYSCLONE.	resolves to the system name  This variable resolves to different values, depending on whether you attach <ul style="list-style-type: none"> <li>• a system layer to a system</li> <li>• a sysplex layer to a sysplex</li> </ul> Refer to "Example Scenarios" on page 8-41 for more information.
&SYSCLONE	No. If not used, must specify &SYSNAME.	resolves to the system clone ID
user-specified name	Yes. A Layer object (or Layer set) name must have a minimum of 3 nodes in the name.	1- to 46-character name for the object

**Note:** You can specify the three nodes of the Layer object pattern in any order in the **Pattern** field of the Started Task Definition panel.

When you enter the definitions for the Layer object, be sure also to enter a name for the object in the **Name** field in addition to a 3-node name in the **Pattern** field (see Step 3 and Step 4 on page 8-16).

This object name is very important because you will see and use this object name on the current system to edit and maintain this Layer object. When objects are created from Layer objects on other systems or sysplexes, the object names are derived from the 3-node name in the **Pattern** field. In addition, the objects cannot be edited and changed on the remote systems. You must edit the definitions for the Layer object on the system it was created and those changes can be copied to the remote systems and sysplexes.

## Layer Set Naming Conventions

You must use the following format to name a Layer set:

**&SYSPLEX.userSpecifiedName**

This name is also referred to as a layer set *pattern*. The Layer set pattern of a Layer set must contain at least 2 nodes and can be 64 characters, as described in the following table:

Node	Required?	Description
&SYSPLEX	Yes	resolves to the sysplex name  This name is required for every Layer object or Layer set.
User specified name	Yes. A Layer object (or Layer set) name must have a minimum of 3 nodes in the name.	1- to 54-character name for the set.

**Note:** You can specify the two nodes of the layer pattern name for a Layer set in any order in the **Layer Pattern** field of the New Name pop-up that is issued from Started Task Sets panel.

When you enter the definitions for the Layer set, be sure to also enter a name for the set in the > (Name) field in addition to a name in the **Layer Pattern** field (see Step 3 and Step 5 on page 8-23).

This set name is very important because you will see and use this set name on the current system to edit and maintain this Layer set. When sets are created from Layer sets on other systems or sysplexes, the set names are derived from the 2-node name in the **Layer Pattern** field. In addition, the sets cannot be edited and changed on the remote systems. You must edit the definitions for the Layer set on the system it was created and those changes can be copied to the remote systems and sysplexes.

## Creating a Layer Object

Layer objects are specified on the same Started Task Definition panel as non-Layer objects. The only differences between defining a non-Layer object and a Layer object are

- how to specify a name for a Layer object in the **Pattern** field
- how to specify the object definition in the Valid Systems pop-up panel
- how to enter the system name for a Dependency Property value

The following section describes these processes.

### Specifying a Pattern Name for a Layer Object

To specify a name for a Layer object on the Started Task Definition panel:

- Step 1** On the TOM Primary Options Menu, enter **1** on the **Option** line (for Started Task), as shown in Figure 8-11.

**Figure 8-11** Creating a Layer Object

```

BMC Software          TOM Primary Options Menu

Option ==> 1

1 Started Task      - Define, display and manage objects      User ID: BAOMXY1
2 Sets             - Define, display and manage sets          Server Id: TOMN1
3 Status           - Define, display and manage systems        Release: 1.1D
4 Calendar         - Define, display and manage calendars      System: SJSD

A Administration   - System Administration
M Messages         - List messages
L Log              - Display Log

X Exit             - Exit

Copyright BMC Software, Inc. 2003
    
```

The Started Task Overview panel is displayed (Figure 8-12).

**Figure 8-12 Started Task Overview Panel**

```

Started Task Overview
Row 1 to 3 of 3
Command ==> Scroll ==> CSR

Primary Commands: ADD  CMDSHOW  CUST

Line Commands: B-browse D-delete DD-delete block E-edit R-repeat
               : O-bounce P-stop S-start CH-change status
               : L-lock UL-unlock T-reset U-suspend A-activate
               : C-status K-block UK-unblock M-move J-propagate
               : BT-browse real time

Definition Base: DEFAULT

LC Description Except Control Type STC
-----
>-----
__ This is a Model object for BPO6607 NO NORM AAOCSMN3
__ TSO Address Space NO SUSPEND NORM TEST
__ non-Layer Object 1 NO ACTIVE NORM AAOCSMN2
***** End of Data *****

```

**Step 2** On the **Command** line, enter **ADD** to add a new Layer object.

The Started Task Definition panel is displayed. Figure 8-13 shows an example of the upper portion of the Started Task Definition panel.

**Figure 8-13 Defining a Layer Object**

```

Started Task Definition
Command ==>

More: +

General information

Name: CICS.SYSTEM1

Description: CICS on SYSTEM1 Product STM
Last modified: 2003/07/17 15:31:21 BAORMB3

STC name: CICS0001 Step name: Type: N ( TRAN or NORM )

This is a Model only: ( Y or N ) Control: ( Active or Suspend )
Modeled from:
Pattern:
.
.
.

```

**Step 3** Enter all object definitions for the Layer object as explained for non-Layer objects in “Creating and Managing Objects” on page 3-1.

**Note:** When you enter the definitions for the Layer object, be sure to enter a name for the object in the **Name** field. You will use this object name on the current system to edit and maintain this Layer object. When objects are created from Layer objects on other systems or sysplexes, the object names are derived from the **Pattern** name and the names cannot be edited and changed on those systems.

**Step 4** In the **Pattern** field, enter the pattern for the Layer object in one of the following formats:

**&SYSPLEX.&SYSNAME.userSpecifiedName**

**&SYSPLEX.&SYSCLONE.userSpecifiedName**

Refer to “Layer Object Naming Conventions” on page 8-12 for information about name formats of the Layer object.

Figure 8-14 shows an example of the Pattern field looks like with a example Layer object name filled in.

**Figure 8-14 Defining a Layer Object**

```

Started Task Definition
Command ==>
General information
Name: CICS.SYSTEM1
Description: CICS on SYSTEM1
Last modified: 2003/07/17 15:31:21
STC name: CICS0001 Step name:
Type: N ( TRAN or NORM )
This is a Model only: ( Y or N ) Control: ( Active or Suspend )
Modeled from:
Pattern: &SYSPLEX.&SYSNAME.CICS object
.
.
.
    
```

**Step 5** Enter the other definitions for the Layer object on the Started Task Definition panel just as if you were creating a non-Layer object (except for the Valid Systems field and the Dependency Property fields).

For information about how to define Valid Systems for a Layer object, refer to “Specifying Valid Systems for a Layer Object” on page 8-18.

For information about how to define Dependency Property for a Layer object, refer to “Specifying Dependency Properties for a Layer Object” on page 8-20.

## Specifying Valid Systems for a Layer Object

To correlate the Layer ID for a system in a sysplex with the variables that are required for the **Valid Systems** list:

- Step 1** On the Started Task Definition panel, place the cursor on the **Valid Systems** hyperlink and press **Enter**.

The Valid Systems pop-up panel is displayed (Figure 8-15) where you define the list of valid systems on which this Layer object can run.

**Figure 8-15** Defining a Layer Object: Valid Systems Pop-Up Panel

```

Valid Systems
Row 0 to 0 of 0
Command===>          Scroll ===> CSR

Primary commands: ALL-all systems  A-add system
Line commands:  I-insert  R-repeat
                D-delete

LC System   Object
Name       Status
-----
***** End of Data *****

```

The system name must be specified with the `&SYSnnnn` variable that correlates to the Layer ID of the system.

*nnnn* is a number from 1 to 9999 and corresponds to the Layer ID of the system defined on the Systems Definition panel (refer to “Entering System Names and Layer IDs for the Sysplex” on page 8-9).

For this example, suppose you have the following systems and Layer IDs defined for sysplex BBPLEX04 on the System Definitions panel:

- SYS\_A, with Layer ID 1
- SYS\_B, with Layer ID 2
- SYS\_C, with Layer ID 3
- SYS\_D, with Layer ID 4

See Figure 8-16.

**Figure 8-16 System Definitions Panel**

```

System Definitions                               Row 0 to 0 of 0
Command ===>

Primary Commands: ADD  CMDSHOW
Line Commands: D-delete  AT-attach system layer

Systems in Sysplex BBPLEX04   ( remote )

  System   Layer
  LC Name  ID   Description                Status   Clone
-----
  SYS_A   1   Side A system: Production only   N/A     SA
  SYS-B   2   Side B system: Pre-Production   N/A     SB
  SYS-C   3   Side C system: Testing OK       N/A     SC
  SYS_D   4   Side D system: Testing OK       NA      SD
-----
***** End of Data *****

```

**Step 2** To run the Layer object on SYS\_B and SYS\_D, enter **&SYS2** and **&SYS4** on the Valid Systems pop-up panel, as shown in Figure 8-17.

**Figure 8-17 Defining a Layer Object: Valid Systems Pop-Up Panel (2)**

```

Valid Systems                               Row 0 to 0 of 0
Command===>                               Scroll ===> CSR

Primary commands: ALL-all systems  A-add system
Line commands: I-insert  R-repeat
                D-delete

LC System   Object
Name       Status
-----
&SYS2     UNKNOWN
&SYS4     UNKNOWN
-----
***** End of Data *****

```

When the system layer is attached to BBPLEX04, the Valid Systems definitions listed for the non-Layer object that is created from this Layer object will resolve to SYS\_B and SYS\_D.



## Creating Layer Sets

Layer sets are defined on the same Started Task Sets panel as non-Layer sets. The only differences between defining a non-Layer set and a Layer set are as follows:

- A different method exists for specifying a Layer set name in the **Layer Pattern** field.
- Layer sets can contain *only* Layer objects and other Layer sets.

Similar to a non-Layer set, a Layer set can contain up to 16 “nested” levels. For example, Layer Set1 can contain Layer Set2, which can contain Layer Set3 (and so on) until Layer Set16.

To specify a name for a Layer set:

- Step 1** On the TOM Primary Options Menu, enter **2** on the **Option** line (for Sets), as shown in Figure 8-19.

**Figure 8-19 Creating a Layer Set**

```

BMC Software                TOM Primary Options Menu

Option ==> 2

1 Started Task  - Define, display and manage objects      User ID: BAOMXY1
2 Sets         - Define, display and manage sets          Server Id: TOMN1
3 Status       - Define, display and manage systems      Release: 1.1D
4 Calendar     - Define, display and manage calendars    System: SJSD

A Administration - System Administration
M Messages      - List messages
L Log           - Display Log

X Exit         - Exit

Copyright BMC Software, Inc. 2003
    
```

The Started Task Sets panel is displayed (Figure 8-20).

**Figure 8-20 Started Task Sets Panel**

```

                Started Task Sets                Row 1 to 5 of 5
Command ==>                Scroll ==> CSR

Primary commands: ADD  CMDSHOW  CUST  SORT

Line commands: B-browse  E-edit  D-delete  DD-delete block  R-repeat
                : S-start  P-stop  O-bounce  L-lock  UL-unlock  J-propagate
                : C-status  H-shutset  HL-shutset/lock  U-suspend  A-activate
                : T-reset  M-move  K-block  UK-unblock

LC Description                Members Set
                Type Number Status
-----
>-----
***** End of Data *****
    
```

- Step 2** On the **Command** line, enter **ADD** to add a new Layer set.

A pop-up panel is displayed for adding a new set name (Figure 8-21).

**Figure 8-21 Pop-Up Panel for Defining a Layer Set**

```

Command ==>

>

Description:

Layer Pattern:

Press ENTER to continue or CANCEL to exit
    
```

- Step 3** At the >, enter the name of the set.
- Step 4** (*optional*) In the **Description** field, enter a 1- to 40-character description.
- Step 5** In the **Layer Pattern** field, enter the name of the Layer set pattern in the following format:

```
&SYSPLEX.userSpecifiedName
```

Refer to “Layer Set Naming Conventions” on page 8-13 for information about name formats for a Layer set.

Figure 8-22 shows an example of the **Layer Pattern** field looks like with a example Layer Pattern set name filled in

**Figure 8-22** Defining a Layer Set

```
Command ==>>
> CICS.PROD.SET
Description: Set of CICS Layer objects
Layer Pattern: &SYSPLEX.CICS
Press ENTER to continue or CANCEL to exit
```

- Step 6** Press **Enter**.

The Members of Set pop-up panel is displayed (Figure 8-23).

**Figure 8-23** Members of Set Pop-Up Panel

```

Members of Set
Row 0 to 0 of 0
Scroll ==> CSR
Command ==>>
Primary commands: AO-add object AS-add set
Line commands: D-delete

Short   Member   Member
LC NAME Type    Name
-----
***** End of Data *****
```

- Step 7** To add Layer objects to the set, enter **AO** on the **Command** line.

**Note:** You can also enter the **AS** primary command to add Layer objects or Layer sets to the Layer set.

An Objects pop-up panel is displayed (Figure 8-24).

**Figure 8-24 Objects Pop-Up Panel**

```

                                Objects
                                Row 1 to 2 of 2
Command ==>                               Scroll =

Primary commands: CMDSHOW CUST SORT
Line commands: S-select

LC Sel Object Name
-----
>-----
__ Layer Object CICS
__ Layer object TSO
__ AAOCSMN2
__ AAOCSMN3
***** End of Data *****
    
```

**Step 8** In the LC field, enter S (for Select) next to the Layer objects that you want to add to the Layer set.

**Step 9** Press **Enter** to save.

The Members of Set pop-up panel is redisplayed.

**Figure 8-25 Members of Set Pop-Up Panel with Object Selected**

```

                                Members of Set
                                Row 0 to 0 of 0
Command ==>                               Scroll ==> CSR

Primary commands: AO-add object AS-add set
Line commands: D-delete

Short   Member   Member
LC NAME Type     Name
-----
__ _____ OBJECT Layer Object CICS
__ _____ OBJECT Layer object TSO
***** End of Data *****
    
```

**Step 10** (*optional*) In the **Short NAME** field, enter a 1- to 8-character name next to the set.

**Step 11** Press **END** to save.

**Step 12** Repeat Step 2 through Step 11 to create additional Layer sets.

## Adding Layer Objects and Layer Sets to a System Layer

You can use a system layer to create a group of both Layer objects and Layer sets, and then you can apply the group to a sysplex (or another system) as a single entity. This collection prevents you from having to add each Layer object or Layer set separately to a sysplex.

You can also use a layer to logically group similar Layer objects together. For example, during your planning phase, you might decide to use one system layer to group all Layer objects that represent CICS objects and use a separate system layer for all WebSphere objects.

You can also create a system layer if you want to group together objects that are common to every system in your all your sysplexes. This system layer can be used to quickly and easily populate new systems that are added to this sysplex and across many sysplexes.

To add Layer objects or Layer sets to a system layer:

- Step 1** On the Administration Options Menu, enter **4** on the **Option** line (for Layers).

The Layer Management menu is displayed (Figure 8-26).

**Figure 8-26** Adding Layer Objects and Sets to a System Layer

```
                                Layer Management
Option  ==>

1 Sysplex Matrix - Manage system definitions
2 System Layers  - Define object groups
3 Sysplex Layers - Define system groups

X Exit          - Exit
```

- Step 2** On the **Option** line, enter **2** (for System Layers).

The System Layers panel is displayed (Figure 8-27).

**Figure 8-27 System Layers Panel**

```

                                System Layers                                Row 1 to 1 of 1
Command ==>

Primary Commands: ADD  CMDSHOW

    Line Commands: D-delete layer  O-layer objects  S-layer sets

LC Layer Name          Count Description                                Updated
                        _____                                Date
>-----
***** End of Data *****
    
```

**Step 3** On the **Command** line, enter **ADD** to add a new system layer.

The Add System Layer pop-up panel is displayed (Figure 8-28).

**Figure 8-28 Add System Layer Pop-Up Panel**

```

                                Add System Layer
Command ==>

    Name:
    Description:

Press ENTER to continue or CANCEL to exit
    
```

**Step 4** In the **Name** field, enter the 1- to 16-character name of the new system layer.

**Step 5** (*optional*) In the **Description** field, enter a 1- to 40-character description of the system layer.

**Step 6** Press **Enter** to add the new system layer.

The name of the new system layer is displayed on the System Layers panel (Figure 8-29).

**Figure 8-29 System Layers Panel: New System Layer Added**

```

                                System Layers                                Row 1 to 1 of 1
Command ==>

Primary Commands: ADD  CMDSHOW

    Line Commands: D-delete layer  O-layer objects  S-layer sets

LC Layer Name          Count Description                                Updated
                        _____                                Date
>-----
__ System Layer 1      0 Example of a system layer_____ 2003/08/26
***** End of Data *****

```

**Step 7** To add Layer objects to the new system layer, enter **O** in the **LC** field next to the system layer name.

The System Layer Objects panel is displayed (Figure 8-30).

**Figure 8-30 System Layer Objects Panel**

```

                                System Layer Objects                                Row 0 to 0 of 0
Command ==>

Define objects for layer: System Layer 1

Line Commands: R-remove  S-select

LC Sel Layer Object Name
_____
-----
__  &sysplex.&sysname.TSO
__  &sysplex.&sysname.JES2
__  &sysplex.&sysname.CICSPROD
__  &sysplex.&sysnmae.CICSTEST
***** End of Data *****

```

The System Layer Objects panel lists the names of all the Layer objects you created.

**Step 8** In the **LC** field, enter **S** next to the Layer objects that you want to add.

The word *Yes* appears in the **Sel** column next to your selection.

**Step 9** Press **END** to save all selected Layer objects.

The System Layers panel is redisplayed.

**Step 10** To add Layer sets to the system layer, enter **S** in the **LC** field next to the system layer name.

The System Layer Sets panel is displayed (Figure 8-31).

**Figure 8-31 System Layer Sets Panel**

```

                                System Layer Sets                                Row 1 to 4 of 4
Command ==>
Define sets for layer: System Layer 1
Line Commands: R-remove  S-select
LC Sel Layer Set Name
-----
___  &sysplex.CICS objects
___  &sysplex.VTAM
___  &sysplex.DB2 objects
___  &sysplex.Netview objects
***** Data *****

```

The System Layer Sets panel lists the names of all of the Layer sets that you created.

**Step 11** In the **LC** field, enter **S** next to the Layer sets that you want to add.

The word **Yes** appears in the **Sel** column next to your selection.

**Step 12** Press **END** to save all selected Layer sets.

The System Layers panel is redisplayed.

**Step 13** Perform the appropriate steps as necessary:

- Repeat Step 3 through Step 6 to add additional system layers.
- Repeat Step 8 through Step 11 to add Layer objects to a system layer.
- Repeat Step 12 through Step 15 to add Layer sets to a system layer.

## Adding System Layers to a Sysplex Layer

A sysplex layer is a superset that contains system layers. By having organized system layers, you have more options when managing the sysplex layer. For example, you might collect all CICS objects in a single system layer. Then you can include only that system layer in a sysplex layer and update all CICS objects in a sysplex at once.

In addition, you can include multiple system layers in a sysplex layer and update many types of objects and sets with a single action.

To add system layers to a sysplex layer:

- Step 1** On the Administration Options Menu, enter **4** on the **Option** line (for Layers).

The Layer Management menu is displayed (Figure 8-32).

**Figure 8-32 Adding System Layers to a Sysplex Layer**

```

                                     Layer Management
Option ==>

1 Sysplex Matrix - Manage system definitions
2 System Layers  - Define object groups
3 Sysplex Layers - Define system groups

X Exit          - Exit

```

- Step 2** On the **Option** line, enter **3** (for Sysplex Layers).

The Sysplex Layers panel is displayed (Figure 8-33).

**Figure 8-33 Sysplex Layers Panel: Adding System Layers**

```

                                     Sysplex Layers
Command ==>

Primary Commands: ADD  CMDSHOW

      Line Commands: D-delete  M-manage

LC Layer Name      Count Description                                Updated
                   |         |                                     Date
-----|-----|-----|-----|-----|-----|-----|-----|-----|
>-----|-----|-----|-----|-----|-----|-----|-----|
***** End of Data *****

```

- Step 3** On the **Command** line, enter **ADD** to add a new sysplex layer.

The Add Sysplex Layer pop-up panel is displayed (Figure 8-34).

**Figure 8-34 Add Sysplex Layer Pop-Up Panel**

```

                                Add Sysplex Layer
Command ==>

                                Name:
                                Description:

Press ENTER to continue or CANCEL to exit
    
```

**Step 4** In the **Name** field, enter the 1- to 16-character name of the new sysplex layer.

**Step 5** (*optional*) In the **Description** field, enter a 1- to 40-character description of the sysplex layer.

**Step 6** Press **Enter** to add the new sysplex layer.

The name of the new sysplex layer is displayed on the Sysplex Layers panel (Figure 8-35).

**Figure 8-35 Sysplex Layers Panel: New Sysplex Layer Added**

```

                                Sysplex Layers                                Row 1 to 1 of 1
Command ==>

Primary Commands: ADD  CMDSHOW

    Line Commands: D-delete  M-manage

LC Layer Name      Count  Description                                Updated
-----
>-----
__ Test Sysplex    0    Sysplex layer for test systems            2003/08/26
***** End of Data *****
    
```

**Step 7** To add system layers to the new sysplex layer, enter **M** (for Manage) in the **LC** field next to the sysplex layer name.

The Sysplex Layer Definitions panel is displayed (Figure 8-36).

**Figure 8-36 Sysplex Layer Definitions Panel**

```

                                Sysplex Layer Definitions                                Row 1 to 1 of 1
Command ==>

Define System layers for Sysplex layer: Test Sysplex

Line Commands: R-remove  S-select

LC Sel System Layer Name Description
-----
_      System Layer 1
***** End of Data *****

```

The Sysplex Layer Definitions panel lists the names of all system layers that you created.

**Step 8** In the **LC** field, enter **S** next to each system layer that you want to add.

The word *Yes* appears in the **Sel** column next to your selection.

**Step 9** Press **END** to save all selected system layers.

The Sysplex Layer Definitions panel is redisplayed.

**Step 10** Repeat this process for all sysplex layers to which you want to add system layers.

## Attaching a System Layer to a System within the Same Sysplex

When you are managing many systems within a sysplex, you can use a system layer to quickly create non-Layer objects and sets on the system within the same sysplex.

To create non-Layer objects and sets from layer objects and sets, you must attach the system layer (or sysplex layer) to another system (or sysplex).

To attach a system layer to a system within the same sysplex where you created the Layer objects and sets:

**Step 1** On the Administration Options Menu, enter **4** on the **Option** line (for Layers).

The Layer Management menu is displayed (Figure 8-37).

**Figure 8-37 Attaching a System Layer to a Local System**

```

                                     Layer Management
Option ==>

1 Sysplex Matrix - Manage system definitions
2 System Layers  - Define object groups
3 Sysplex Layers - Define system groups

X Exit          - Exit

```

**Step 2** On the **Option** line, enter **1** (for Sysplex Matrix).

The Sysplex Matrix panel is displayed (Figure 8-38).

**Figure 8-38 Sysplex Matrix Panel: Attaching a System Layer to a Local System**

```

                                     Sysplex Matrix
                                     Row 1 to 4 of 4
Command ==>

Primary Commands: ADD  CMDSHOW

Line Commands: D-delete Sysplex definition
               M-manage systems in Sysplex
               AT-attach sysplex layer to Sysplex

Sysplex  Number  Systems
LC Name   Systems  Active  Description
-----
>-----
__ BBPLEX01      6      6
__ BBPLEX04      0      0 Remote system 4
__ BBPLEX03      2      0 Production System
__ BBPLEX02      2      0 Weekend Backups

```

**Step 3** In the LC field, enter M (for Manage) next to the name of the local sysplex.

**Figure 8-39 Sysplex Matrix Panel: Attaching a System Layer to a Local System**

```

                                Sysplex Matrix                                Row 1 to 4 of 4
Command ==>

Primary Commands: ADD  CMDSHOW

    Line Commands: D-delete Sysplex definition
                  M-manage systems in Sysplex
                  AT-attach sysplex layer to Sysplex

    Sysplex  Number  Systems
LC Name      Systems Active  Description
-----
>-----
_M BBPLEX01      6      6
__ BBPLEX04      0      0 Remote system 4
__ BBPLEX03      2      0 Production System
__ BBPLEX02      2      0 Weekend Backups
***** End of Data *****

```

The System Definitions panel is displayed (Figure 8-40). The individual systems within the local sysplex are listed.

**Figure 8-40 System Definitions Panel: Attaching a System Layer to a Local System**

```

                                System Definitions                                Row 1 to 6 of 6
Command ==>

Primary Commands: ADD  CMDSHOW
    Line Commands: D-delete  AT-attach system layer

Systems in Sysplex BBPLEX01  ( local )

    System  Layer
LC Name    ID   Description          Status  Clone
-----
-----
__ SJSB    __1  _____  ACTIVE  SB
__ SJSC    __2  _____  ACTIVE  SC
__ SJSD    __3  _____  ACTIVE  SD
__ SJSE    __4  _____  ACTIVE  SE
__ SJSG    __5  _____  ACTIVE  SG
__ SJSH    __6  _____  ACTIVE  SH
***** End of Data *****

```

**Step 4** In the LC field, enter AT (for attach system layer) next to the name of the system to which you want to attach a system layer.

The Attach System Layer pop-up panel is displayed (Figure 8-41).

**Figure 8-41 Attach System Layer Pop-Up Panel**

```

          Attach System Layer
Command ==>

S-select Layer to attach

LC Sel System Layer Name Description
-----
_      System Layer 1
***** End of Data *****

```

**Step 5** In the LC field, enter S next to the name of the system layer that you want to attach.

**Step 6** Press END to save.

Attach Submitted appears in the upper right corner of the panel.

After you attach a system layer to a local system, the non-Layer objects and sets that are created from the Layer objects and Layer sets are immediately available in the active TOM registry. If you specified Active in the **Control** field when the Layer objects were defined, TOM immediately begins managing the non-Layer objects.

**Note:** You can also display the Started Task Overview panel and the Started Task Sets panel to see the new non-Layer objects and sets that were created from the Layer objects and Layer sets in the system layer that you just attached.

## Attaching a System Layer to a System in a Remote Sysplex

For each Layer Set that is being attached, TOM creates a non-Layer Set but does not create the members of this Set if they do not previously exist on the remote sysplex. Therefore when attaching a Layer Set with a system or sysplex Layer, all Layer objects and Sets that are members of this Set have to be included in the same system or sysplex layer.

To attach a system layer to a system in a remote sysplex:

**Step 1** On the Administration Options Menu, enter **4** on the **Option** line (for Layers).

The Layer Management menu is displayed (Figure 8-42).

**Figure 8-42 Attaching a System Layer to a Remote System**

```

                                Layer Management
Option  ===>

1 Sysplex Matrix - Manage system definitions
2 System Layers  - Define object groups
3 Sysplex Layers - Define system groups

X Exit          - Exit

```

**Step 2** On the **Option** line, enter **1** (for Sysplex Matrix).

The Sysplex Matrix panel is displayed (Figure 8-43).

**Figure 8-43 Sysplex Matrix Panel: Attaching a System Layer to a Remote System**

```

                                Sysplex Matrix                                Row 1 to 4 of 4
Command ===>

Primary Commands: ADD  CMDSHOW

Line Commands: D-delete Sysplex definition
                M-manage systems in Sysplex
                AT-attach sysplex layer to Sysplex

Sysplex  Number  Systems
LC Name   Systems Active  Description
-----
>-----
__ BBPLEX01      6      6 _____
__ BBPLEX04      0      0 Remote system 4_____
__ BBPLEX03      2      0 Production System _____
__ BBPLEX02      2      0 Weekend Backups_____

```

**Step 3** In the LC field, enter **M** (for Manage) next to the name of the remote sysplex.

**Figure 8-44 Sysplex Matrix Panel: Attaching a System Layer to a Remote System**

```

                                Sysplex Matrix                                Row 1 to 4 of 4
Command ==>

Primary Commands: ADD  CMDSHOW

    Line Commands: D-delete Sysplex definition
                  M-manage systems in Sysplex
                  AT-attach sysplex layer to Sysplex

    Sysplex  Number  Systems
LC Name     Systems Active  Description
-----
>-----
__ BBPLEX01      6      6
__ M BBPLEX04      0      0 Remote system 4
__ BBPLEX03      2      0 Production System
__ BBPLEX02      2      0 Weekend Backups
***** End of Data *****
    
```

The System Definitions panel is displayed (Figure 8-45). The individual systems within the remote sysplex are listed.

**Figure 8-45 System Definitions Panel**

```

                                System Definitions                                Row 1 to 6 of 6
Command ==>

Primary Commands: ADD  CMDSHOW
    Line Commands: D-delete  AT-attach system layer

Systems in Sysplex BBPLEX04 ( remote )

    System  Layer
LC Name    ID   Description          Status  Clone
-----
-----
__ SJSB    __1  _____  ACTIVE  JB
__ SJSC    __2  _____  ACTIVE  JC
__ SJSD    __3  _____  ACTIVE  JD
__ SJSE    __4  _____  ACTIVE  JE
__ SJSG    __5  _____  ACTIVE  JG
__ SJSH    __6  _____  ACTIVE  JH
***** End of Data *****
    
```

**Step 4** In the LC field, enter **AT** (for attach system layer) next to the name of the system to which you want to attach a system layer.

The Attach System Layer pop-up panel is displayed (Figure 8-46).

**Figure 8-46 Attach System Layer Pop-Up Panel**

```

          Attach System Layer
Command ===>

S-select Layer to attach

LC Sel System Layer Name Description
-----
_ System Layer 1
***** End of Data *****

```

**Step 5** In the LC field, enter S next to the name of the system layer that you want to attach.

**Step 6** Press END to save.

Attach Submitted appears in the upper right corner of the panel.

Unlike attaching a system layer to a local system, the non-Layer objects and sets that are created for a remote system are not immediately available at this point. You must complete the process.

**Step 7** Manually copy the current TOM registry to the registry that is used in the remote sysplex.

For information about how to copy the registry, refer to “Copying the Registry” on page 8-47.

After the TOM registry is copied to the new sysplex, the newly generated non-Layer objects will run on systems within the sysplex, as defined by two factors:

- the `&SYSnnnn` variable in the Valid Systems definition for the Layer object (where `nnnn` represents the Layer ID of a system)

This variable is described in “Specifying Valid Systems for a Layer Object” on page 8-18.

- the system name that is associated with the Layer IDs (described in “Entering System Names and Layer IDs for the Sysplex” on page 8-9)

For more information about how non-Layer object names are resolved and what systems they are running on after an attach, refer to “Example Scenarios” on page 8-41.

## Attaching a Sysplex Layer to a Sysplex

When you attach a sysplex layer to another sysplex, you are propagating new non-Layer objects and sets from the Layer objects and Layer sets that are contained in the sysplex layer.

For each Layer Set that is being attached, TOM creates a non-layer Set but does not create the members of this Set if they do not previously exist on the remote sysplex. Therefore when attaching a Layer Set with a system or sysplex Layer, all Layer objects and Sets that are members of this Set have to be included in the same system or sysplex layer.

To attach a sysplex layer to a sysplex:

- Step 1** On the Administration Options Menu, enter **4** on the **Option** line (for Layers).

The Layer Management menu is displayed (Figure 8-47).

**Figure 8-47 Attaching a Sysplex Layer to a Remote Sysplex**

```

                                Layer Management
Option ==>

1 Sysplex Matrix - Manage system definitions
2 System Layers  - Define object groups
3 Sysplex Layers - Define system groups

X Exit          - Exit

```

- Step 2** On the **Option** line, enter **1** (for Sysplex Matrix).

The Sysplex Matrix panel is displayed (Figure 8-48).

**Figure 8-48 Sysplex Matrix Panel: Attaching a Sysplex Layer to a Remote Sysplex**

```

                                Sysplex Matrix                                Row 1 to 4 of 4
Command ==>

Primary Commands: ADD  CMDSHOW

Line Commands: D-delete Sysplex definition
                M-manage systems in Sysplex
                AT-attach sysplex layer to Sysplex

Sysplex Number Systems
LC Name   Systems Active Description
-----
>-----
__ BBPLEX01    6      6
__ BBPLEX04    0      0 Remote system 4
__ BBPLEX03    2      0 Production System
__ BBPLEX02    2      0 Weekend Backups

```

- Step 3** In the LC field, enter **AT** (for attach sysplex layer) next to the name of the remote sysplex.

**Figure 8-49 Sysplex Matrix Panel: Attaching a Sysplex Layer to a Remote Sysplex**

```

                                Sysplex Matrix                                Row 1 to 4 of 4
Command ==>

Primary Commands: ADD  CMDSHOW

      Line Commands: D-delete Sysplex definition
                    M-manage systems in Sysplex
                    AT-attach sysplex layer to Sysplex

      Sysplex  Number  Systems
LC Name      Systems Active  Description
-----
>-
__ BBPLEX01      6      6
__ BBPLEX04      0      0 Remote system 4
__ T BBPLEX03    2      0 Production System
__ BBPLEX02      2      0 Weekend Backups
***** End of Data *****

```

The Attach Sysplex Layer pop-up panel is displayed (Figure 8-50). The sysplex layer names are listed.

**Figure 8-50 Attach Sysplex Layer Pop-Up Panel**

```

                                Attach Sysplex Layer                                Row 1 to 4 of 4
Command ==>

S-select Layer to attach

LC Sel Layer Name      Description
-----
__ Test Sysplex        Sysplex layer for test systems
__ NEW SPL LAYER       Test for new Sysplex Layer
__ SYSPLLAYER          Prod Sysplex Layer
__ BBLAYER 123         Test Sysplex Layer
***** End of Data *****

```

- Step 4** In the LC field, enter **S** next to the name of the sysplex layer that you want to attach to the remote sysplex.

- Step 5** Press **END** to save.

Attach Submitted appears in the upper right corner of the panel.

All Layer objects and sets in all system layers of a sysplex layer are propagated into a sysplex during the attach process. The generated non-Layer objects will run on systems within a sysplex, as defined by two factors:

- the `&SYSnnnn` variable in the Valid Systems definition for the Layer object (where *nnnn* represents the Layer ID of a system)

This variable is described in “Specifying Valid Systems for a Layer Object” on page 8-18.

- the system name that is associated with the Layer IDs (described in “Entering System Names and Layer IDs for the Sysplex” on page 8-9)

For more information about how non-Layer object names are resolved and what systems they are running on after an attach, refer to “Example Scenarios” on page 8-41.

## Example Scenarios

This section has two sections:

- “Example Scenarios” shows how a non-Layer object name is derived differently when you attach to a system or to another sysplex.
- “Example of Creating a Layer Object for Multiple Systems” shows an example scenario for using a Layer object to create the non-Layer object TSO on multiple systems in two sysplexes.

### How Object Names Are Resolved after an Attach

The following scenario provides an example of how a Layer object’s name can be resolved to two different non-Layer object names depending on whether the non-Layer object is created by attaching a system layer or a sysplex layer.

Table 8-3 describes the individual components of the scenario in the left column and where you can find additional documentation about each component in the right column.

To use Table 8-3, read the left hand column and then refer to the right-hand column when you want to go read the documentation and see examples.

**Table 8-3**      **How Object Names Resolve after an Attach**

Scenario Steps	See Section for More Information
You are planning to use Layer objects to populate a sysplex named BBPLEX01	“Entering Sysplex Names” on page 8-6
Within sysplex BBPLEX01, there are 2 systems named: SYSA SYSB	“Entering System Names and Layer IDs for the Sysplex” on page 8-9
You assign Layer IDs to the systems as follows: <ul style="list-style-type: none"> <li>• SYSA, Layer ID = 1,</li> <li>• SYSB, Layer ID = 2</li> </ul>	“Entering System Names and Layer IDs for the Sysplex” on page 8-9
Suppose you have a Layer object name is: &SYSPLEX . &SYSNAME . TEST . TSO	“Layer Object Naming Conventions” on page 8-12 “Creating a Layer Object” on page 8-14 “Specifying a Pattern Name for a Layer Object” on page 8-14

**Table 8-3 How Object Names Resolve after an Attach**

Scenario Steps	See Section for More Information
<p>In the Layer object definition, in the Valid Systems pop-up panels, you specify that the system the object will run on in BBPLEX01 is the system named SYSB.</p> <p>To enter the system name for SYSB, you must enter the variable:</p> <p>&amp;SYS2</p> <p>Where 2 represents the Layer ID you assigned SYSB.</p>	<p>"Specifying Valid Systems for a Layer Object" on page 8-18</p>
<p>Create a system layer and add the Layer object to the system layer. In this example, the system layer name is SYSLAYER01.</p>	<p>"Adding Layer Objects and Layer Sets to a System Layer" on page 8-25</p>
<p>Attach the system layer SYSLAYER01 to the system named SYSA in BBPLEX01.</p>	<p>"Attaching a System Layer to a System in a Remote Sysplex" on page 8-35</p>
<p><b>Results:</b> When you attach the system layer SYSLAYER01 to SYSA in sysplex BBPLEX01, the name of the Layer object, &amp;SYSPLEX.&amp;SYSNAME.TEST.TSO resolves to the non-Layer object name of: BBPLEX01.SYSA.TEST.TSO where:</p> <ul style="list-style-type: none"> <li>• &amp;SYSPLEX resolves to the name of the sysplex, BBPLEX01</li> <li>• &amp;SYSNAME resolves to the name of the system you attached the system layer to, SYSA</li> <li>• TEST.TSO remains the same as the user-defined, literal portion of the object name</li> </ul> <p>If you browse the object's definition for BBPLEX01.SYSA.TEST.TSO, the Valid System is listed as SYSB and the object is running on the system named SYSB.</p>	
<p>Create a sysplex layer called SYSPLEX LAYER 1 and add SYSLAYER01 to it.</p>	<p>"Adding System Layers to a Sysplex Layer" on page 8-29</p>
<p>Attach the sysplex layer named SYSPLEX LAYER 1 to BBPLEX01.</p>	<p>"Attaching a Sysplex Layer to a Sysplex" on page 8-38</p>
<p><b>Results:</b> When you attach SYSPLEX LAYER 1 to BBPLEX01, the name of the layer object, &amp;SYSPLEX.&amp;SYSNAME.TEST.TSO resolves to the non-Layer object name of: BBPLEX01.SYSB.TEST.TSO where:</p> <ul style="list-style-type: none"> <li>• &amp;SYSPLEX resolves to the name of the sysplex, BBPLEX01</li> <li>• &amp;SYSNAME resolves to the name of the system that the non-Layer object is actually running on, SYSB, as defined in the Valid System pop-up panels</li> <li>• TEST.TSO remains the same as the user-defined, literal portion of the object name</li> </ul> <p>If you browse the object's definition for BBPLEX01.SYSB.TEST.TSO, the Valid System is listed as SYSB and the object is running on the system named SYSB.</p>	

## Example of Creating a Layer Object for Multiple Systems

The following example shows how to create a single Layer object to create the non-Layer object TSO in two different sysplexes, BBPLEX01 and BBPLEX02 (each of which has four systems).

**Note:** BMC Software recommends using Layering when you have many systems in many sysplexes. For brevity, this scenario uses a small number of systems and sysplexes.

**Step 1** Define and assign Layer IDs to the systems in the two sysplexes, BBPLEX01 and BBPLEX02.

This task is described in “Entering System Names and Layer IDs for the Sysplex” on page 8-9.

In this sample, the results are as follows:

- BBPLEX01 has the following four systems and Layer IDs assigned:

```
SYSA  Layer ID=1
SYSB  Layer ID=2
SYSC  Layer ID=3
SYSD  Layer ID=4
```

- BBPLEX02 has the following four systems and Layer IDs assigned:

```
SYSE  Layer ID=1
SYSF  Layer ID=2
SYSG  Layer ID=3
SYSH  Layer ID=4
```

**Step 2** On the Started Task Definition panel, create four Layer objects to represent TSO.

The following table lists the values that you must enter to create the four Layer objects for this sample scenario:

Layer Object	Values for the Started Task Definition Panel
Layer object 1	<b>Name</b> field: TSO . LAYER . SYS1 <b>Pattern</b> field: &SYSPLEX . &SYSNAME . TSO <b>Valid Systems</b> list: &SYS1 Plus other object definitions for the TSO object
Layer object 2	<b>Name</b> field: TSO . LAYER . SYS2 <b>Pattern</b> field: &SYSPLEX . &SYSNAME . TSO <b>Valid Systems</b> list: &SYS2 Plus other object definitions for the TSO object

Layer Object	Values for the Started Task Definition Panel
Layer object 3	<b>Name</b> field: TSO . LAYER . SYS3 <b>Pattern</b> field: &SYSPLEX . &SYSNAME . TSO <b>Valid Systems</b> list: &SYS3 Plus other object definitions for the TSO object
Layer object 4	<b>Name</b> field: TSO . LAYER . SYS4 <b>Pattern</b> field: &SYSPLEX . &SYSNAME . TSO <b>Valid Systems</b> list: &SYS4 Plus other object definitions for the TSO object

**Step 3** Create a system layer named **TSO objects System Layer**, and include the four TSO Layer objects in this layer.

This task is described in “Adding Layer Objects and Layer Sets to a System Layer” on page 8-25.

**Step 4** Create a sysplex layer named **Basic Sysplex Layer**, and include **TSO objects System Layer** in this sysplex layer.

This task is described in “Adding System Layers to a Sysplex Layer” on page 8-29.

**Step 5** Attach **Basic Sysplex Layer** to sysplex BBPLEX01.

**Step 6** Attach **Basic Sysplex Layer** to the second sysplex, BBPLEX02.

Attaching is described in “Attaching a Sysplex Layer to a Sysplex” on page 8-38.

---

As a result, the following eight non-Layer objects are created in the current TOM registry (along with the valid systems on which they will run):

- On BBPLEX01, the following objects are created:
  - object BBPLEX01.SYSA.TSO, running on system SYSA
  - object BBPLEX01.SYSB.TSO, running on system SYSB
  - object BBPLEX01.SYSC.TSO, running on system SYSC
  - object BBPLEX01.SYSD.TSO, running on system SYSD

When the registry is copied to the registry that is running on sysplex BBPLEX01, these four non-Layer objects will be managed by TOM.

If BBPLEX01 is the current, local sysplex, there is no need to copy the registry.

- On BBPLEX02, the following objects are created:
  - object BBPLEX02.SYSE.TSO, running on system SYSE
  - object BBPLEX02.SYSF.TSO, running on system SYSF
  - object BBPLEX02.SYSG.TSO, running on system SYSG
  - object BBPLEX02.SYSH.TSO, running on system SYSH

When the registry is copied to the registry that is running on sysplex BBPLEX02, these four objects will be managed by TOM.

From this simple example, you can see that while Layering can prevent you from having to create each TSO object on each system in each sysplex, you have to create one Layer object for *each system*.

## Updating Objects and Sets Created from Layering

After non-Layer objects and non-Layer sets are created on other systems and sysplexes from Layer objects and Layer sets, you cannot edit the definitions of those objects or sets on those systems.

You can only edit the definition of the Layer object or Layer set from the original definitions on the original system on which they were created.

### Updating a Layer Object

To edit a Layer object:

- Step 1** From the TOM Primary Options Menu, enter **1** on the **Option** line to display the Started Task Overview panel.
- Step 2** Enter **S** next to the Layer object to select it.
- Step 3** Edit the definitions on the Started Task Definition panel.
- Step 4** Press **END** to save the changes.
- Step 5** Copy the Registry to the sysplex to effect new changes. (Refer to “Copying the Registry” on page 8-47.)

### Updating a Layer Set

To edit a Layer set:

- Step 1** From the TOM Primary Options Menu, enter **2** on the **Option** line to display the Started Task Sets panel.
- Step 2** Enter **S** next to the Layer set to select it.
- Step 3** Edit the members of the Layer set on the Members of Set pop-up panel.
- Step 4** Press **END** to save the changes.
- Step 5** Copy the Registry to the sysplex to effect new changes. (Refer to “Copying the Registry” on page 8-47.)

## Copying the Registry

When a Layer object or Layer set is updated or deleted, or when a system layer or sysplex layer is attached, all changes are made in the current TOM Registry.

To activate the changes in another sysplex, you must copy the Registry to that sysplex.

The following sections describe the methods for copying the Registry.

### Using the TOM Offload and Upload Utility

You can easily copy (offload) the TOM registry to another data set and then copy (upload) it to another TOM system. You might perform this process for one of the following reasons:

- You have created Layer objects and Layer sets that you have attached, and you need to propagate the non-Layer objects and sets to other sysplexes.
- You have edited the definitions of Layer objects or Layer sets, and you are trying to replicate those changes in the non-Layer objects and sets that are running on other sysplexes.
- You have determined that the current Registry is too small, and you want to create a larger Registry into which you will copy the current Registry.

To offload the Registry by using the MODIFY command:

**Step 1** At a console, enter the following command:

**F *TOMAddressSpace*,REGISTRY DUMP DD(*DDname*)**

The following table describes the parts of the preceding command:

Term	Definition
F	abbreviation for the MODIFY console command

Term	Definition
<i>TOMAddressSpace</i>	Is the name of the TOM address space containing the Registry that you are trying to copy to another TOM address space
<i>DDname</i>	name of a DD card that must refer to an existing, predefined data set with the following attributes:  Organization: PS Record format: VB Record length: 8192 Block size: 27998

**Step 2** During the TOM startup phase, upload the Registry to another TOM by updating TOMPARM member REGSTY00 with a PRIMEDD parameter that contains a valid DD name.

For example, you could specify:

```
PRIMEDD( 'REGDUMP' )
```

where REGDUMP is a name of DD card in the TOM started task JCL that refers to the data set *DDname* from Step 1.

## Using the IDCAMS REPRO Utility

You can also use the IDCAMS REPRO utility to copy the Registry data set from one system to another system.

To use the IDCAMS REPRO utility:

Customize the following JCL and submit it to copy the current Registry data set to another Registry data set.

**Note:** If you choose this method, the two Registry data sets must be identical in size, and TOM must *not* be active when the JCL is submitted.

---

```
//BAORAE JOB (RE62),R.ERNST,CLASS=F,MSGCLASS=R,NOTIFY=&SYSUID
//IDCAMS EXEC PGM=IDCAMS
//INPUT DD DISP=SHR,DSN=BAORAE.TM63.REGISTRY
//OUTPUT DD DISP=(,CATLG,DELETE),
// DCB=(RECFM=FB,LRECL=4096),SPACE=(CYL,20),
// DSN=BAORAE.TM63.REGISTRY.REPRO
//SYSPRINT DD SYSOUT=*
REPRO INFILE(INPUT) OUTFILE(OUTPUT)
```

---

---

---

# Chapter 9 Using AOAnywhere TOM Functions and Console Commands

This chapter describes how to use the AOAnywhere TOM functions and console commands. The chapter includes the following topics:

AOAnywhere TOM Functions .....	9-2
Coding Conventions .....	9-2
AOEXEC TOM Function List .....	9-5
Command Syntax for AOEXEC TOM Functions .....	9-5
TOM Console Commands .....	9-45
Coding Conventions .....	9-45

## AOAnywhere TOM Functions

You might want to manage objects through EXECs. AOAnywhere allows automation functions to be invoked from EXECs that are running both inside and outside of the MAINVIEW AutoOPERATOR BBI-SS PAS. With the AOEXEC TOM functions, you can perform TOM object-management actions from your EXECs, such as

- starting, stopping, or bouncing an object (or set of objects)
- setting a user property state for an object (or set)
- changing an object status back and forth between SUSPEND and ACTIVE

Using the AOEXEC TOM functions from an EXEC has the same effect as issuing line commands against an object (or set) from within the TOM user interface.

## Coding Conventions

The following sections describe restrictions for using the TOMID and XCFGROUP parameters; the sections also discuss other syntax limitations.

## Creating EXECs That Can Run on Multiple TOM Systems

You will most likely want to write EXECs that you can schedule on all systems that are running TOM without having to manually edit and customize the EXEC for each system.

**Tip:** BMC Software strongly recommends that if you have only one TOM system implemented for each MVS system, *omit specifying* the TOMID and XCFGROUP parameters from the AOEXEC TOM functions, thereby making the EXEC generic enough to run on other MVS systems.

If you do not know how many TOM systems are implemented on a system, use the AOEXEC TOM (SYSINFO) command. This command identifies all of the TOM systems on the OS/390 system on which you ran the EXEC.

If multiple TOM systems are implemented on a single OS/390 system, and the AOEXEC TOM function omits the TOMID and XCFGROUP parameters, a random TOM system is selected as the target.

---

## How to Use the TOMID and XCFGROUP Parameters

The optional TOMID and XCFGROUP parameters control which TOM system receives the AOEXEC TOM function. Do not specify both parameters.

The default value of the XCFGROUP parameter is an asterisk (\*), which specifies any or all XCF group names (as appropriate for the requested function).

**Note:** You can only issue AOEXEC TOM functions to a TOM system that is on the OS/390 system that is issuing the EXEC.

If you omit the TOMID and the XCFGROUP parameters, a search is performed to find an active TOM on the system on which the EXEC is scheduled. This is the recommended method.

**Note:** If multiple qualifying TOMs are found on a single OS/390 system, it is not predictable which TOM system is selected.

When the XCFGROUP parameter is specified, for example XCFGROUP(BBEX01), a search is performed to find an active TOM belonging to the XCF Group Name BBEX01 on system that the EXEC is scheduled on.

Examples:

```
AOEXEC TOM FUNCTION(xxx)
```

```
AOEXEC TOM FUNCTION(xxx) TOMID(yyy)
```

```
AOEXEC TOM FUNCTION(xxx) XCFGROUP(zzz)
```

In addition, all TOM repositories within the same XCF group are mirrors of each other so it is not necessary to read or write to remote repositories. Review the parameters settings in BBPARM member XCFDEF00 to determine the XCF group name.

## Object and Set Naming Restrictions

Following are some restrictions for naming objects and sets:

- Object names and set names cannot contain a slash (/) and can be up to 64 characters.

- When you specify an object or set name as part of your syntax for an AOEXEC TOM function, you must enclose the name within single quotation marks (') if it contains lowercase characters, spaces, or special characters.

For example, if the object name is CICS.testsystem.2, you must specify:

**OBJECT('CICS.testsystem.2')**

If the object name has spaces such as CICS.system F, you must specify:

**OBJECT('CICS.system f')**

**Note:** Values that do not use single quotation marks are resolved to uppercase characters. If you have two objects with the same name (but with different combinations of lowercase and uppercase characters), you must apply correct formatting.

For example, OBJECT(CICS.systemD.test) represents the same object as OBJECT(CICS.SYSTEMD.TEST). However, OBJECT('cics.a') and OBJECT('cics.A') are two different objects.

## How to Use Stem Variables

CLIST EXECs are not supported because some AOEXEC TOM functions only return stem variables. Note that CLIST EXECs can invoke these functions by using an intermediary REXX EXEC.

## Return Codes

All AOEXEC functions produce return codes, as follows:

- When an AOEXEC function is used from an EXEC within the MAINVIEW AutoOPERATOR subsystem, the return code is returned in the IMFRC variable (as is normal when calling TSO command processors).
- When an AOEXEC function is used outside of the MAINVIEW AutoOPERATOR subsystem, the return code is returned in the RC special variable.

## Command Syntax for AOEXEC TOM Functions

The AOEXEC TOM functions provide automation services that you can use in an EXEC to perform TOM functions such as starting or stopping an object. Many of the line commands from the Started Task Overview panel have equivalent functions with AOEXEC TOM functions.

The command syntax is the keywords AOEXEC TOM, followed by the function name (such as ACTIVATE, BLOCK) and parameters; for example:

AOEXEC TOM FUNCTION(*functionName*) [*optionalParameters*]

## AOEXEC TOM Function List

Table 9-1 contains a list of all AOEXEC TOM functions and the page where each function is described.

**Table 9-1 List of AOEXEC TOM Functions**

Function Name	Reference Page
ACTIVATE	9-6
BLOCK	9-8
BOUNCE	9-10
CHSTATUS	9-12
GET	9-14
LIST	9-18
LOCK	9-20
MOVE	9-22
RESET	9-25
SETPROPERTY	9-27
SHUTSET	9-29
SHUTSYS	9-30
START	9-31
STOP	9-34
SUSPEND	9-37
SYSINFO	9-39
UNBLOCK	9-41
UNLOCK	9-43

## ACTIVATE Function

The ACTIVATE function corresponds to the Activate line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(ACTIVATE)	OBJECT('name') SET('name') [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
TOMID	Optional. Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Optional. Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

### Examples

Following are some examples of the syntax for the ACTIVATE function:

```
AOEXEC TOM FUNCTION(ACTIVATE) OBJECT('cicsprod.sysa')
```

```
AOEXEC TOM FUNCTION(ACTIVATE) SET(CICSPROD.SYSA)
TOMID(tmek1)
```

## Return Codes

Table 9-2 shows the possible return codes and what they mean.

**Table 9-2 Return Codes for the AOEXEC TOM ACTIVATE Function**

<b>Return Code</b>	<b>Explanation</b>
0	Command successfully completed
4	OBJECT(name) already Active
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
118	Security violation occurred
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

## BLOCK Function

The BLOCK function corresponds to the Block line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(BLOCK)	OBJECT('name') SET('name') HOST(hostname) [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
HOST	Specify the name of an OS/390 system that is specified in the object definition's Valid Systems list.	Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

## Examples

Following are some examples of the syntax for the BLOCK function:

```
AOEXEC TOM FUNCTION(BLOCK) SET('set.cics.sysa')
HOST(SYSA)
```

```
AOEXEC TOM FUNCTION(BLOCK) OBJECT(CICSPROD.SYSB)
HOST(sysb) XCFGROUP(group1)
```

## Return Codes

Table 9-3 shows the possible return codes and what they mean.

**Table 9-3 Return Codes for the AOEXEC TOM BLOCK Function**

Return Code	Explanation
0	Command successfully completed
4	OBJECT(name) is already Blocked
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
116	TOM hostname is inactive or does not exist
118	Security violation occurred
121	HOST(hostname) is not in the Valid Systems list
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

## BOUNCE Function

The BOUNCE function corresponds to the Bounce line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(BOUNCE)	OBJECT('name') SET('name') HOST(hostname) [FULLAUTO] [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
HOST	Specify the name of an OS/390 system that is specified in the object definition's Valid Systems list.	The HOST() keyword is required when using OBJECT(). Do not use the Host() parameter when using SET().  Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
FULLAUTO	Specify this keyword when you use full automation. This means that any of the object's defined Pre or Post-Start EXECs and Pre or Post-Stop EXECs will be scheduled as part of the BOUNCE function.	When this key word is omitted, the Stop and Start occurs without scheduling these EXECs.

Parameter	Function	Notes
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

## Examples

Following are some examples of the syntax for the BOUNCE function:

```
AOEXEC TOM FUNCTION(BOUNCE) OBJECT(CICSPROD.SYSA)
HOST(syssa)
```

```
AOEXEC TOM FUNCTION(BOUNCE) SET('Set.CICS.objects')
FULLAUTO
```

## Return Codes

Table 9-4 shows the possible return codes and what they mean.

**Table 9-4 Return Codes for the AOEXEC TOM BOUNCE Function**

Return Code	Explanation
0	Command successfully completed
4	Object(name) is not Active
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
116	TOM on hostname is inactive or does not exist
118	Security violation occurred
120	Product STM is absent
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

## CHSTATUS Function

The CHSTATUS function corresponds to the CH line command, which can be issued for objects from the Started Task Overview panel.

**Note:** When the @STATUS on the specified host is ACTIVE or ACTIVE-ERR for the object, the CHSTATUS function changes the status to STOPPED and sets a return code of 2.

Otherwise, the CHSTATUS function sets the @STATUS on the host to ACTIVE for the object and sets a return code of 1.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(CHSTATUS)	OBJECT('name') HOST(hostname) [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	CHSTATUS cannot be used with sets. If you specify a set name, you will receive a RC 114.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
HOST	Specify the name of an OS/390 system that is specified in the object definition's Valid Systems list.	Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

### Examples

Following are some examples of the syntax for the CHSTATUS function:

```
AOEXEC TOM FUNCTION(CHSTATUS) OBJECT(CICSPROD.SYSA)
      XCF(group1)
```

## Return Codes

Table 9-5 shows the possible return codes and what they mean.

**Table 9-5 Return Codes for the AOEXEC TOM CHSTATUS Function**

Return Code	Explanation
1	Command successfully completed, new status is Active
2	Command successfully completed, new status is Stopped
8	Object(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
114	A SET was specified and CHSTATUS does not apply to sets, only objects.
116	TOM hostname is inactive or does not exist
118	Security violation occurred
121	HOST(hostname) is not in the Valid Systems list
136	OBJECT(name) was missing
142	Internal error occurred

## GET Function

Use the GET function to request and receive information about an object or set. The information is returned in REXX stem variables.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(GET)	OBJECT('name') SET('name') [STEM(LINE)] [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
STEM	The STEM keyword is optional and specifies the prefix of the REXX stem variables that contain the properties of the object (or set)	The default REXX stem value is LINE.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

See Table 9-6 and Table 9-7 on page 9-16 for a list of variables and example values that the AOEXEC TOM GET function returns.

Table 9-6 shows the variables and example values when AOEXEC TOM GET is used for an object.

**Table 9-6 REXX Stem Variables and Sample Returned Values: Object (Part 1 of 2)**

REXX Stem Variable	Examples of Values Returned
line.affinity	LOW
line.altrnte	NO
line.cntl	SUSPEND
line.confirm	NO
line.descr	a4
line.except	NO
line.forced	NO
line.iplStart	NO
line.lockipl	NO
line.mdlType	NO
line.model	name of model object
line.pattern	pattern name
line.product	STM
line.reinst	MANUAL
line.reinst#	3
line.resetCnt	YES
line.restart	NO
line.resTime	0
line.shutdown	NORMAL
line.type	NORM
line.stm.name	STC proc name
line.stm.step	STC proc step name
line.dep.1.op	IQ
line.dep.1.name	name of object or set
line.dep.1.prop	@STATUS or user-defined property name
line.dep.1.sysname	MVS system name of system TOM checks the property on which is applicable for @STATUS only
line.dep.1.value	value of line.dep.1.prop Property
line.hosts.0	1
line.hosts.1.sysname	SJSD
line.hosts.1.status	STOPPED

**Table 9-6 REXX Stem Variables and Sample Returned Values: Object (Part 2 of 2)**

REXX Stem Variable	Examples of Values Returned
line.hosts.1.strtswtc	Y
line.hosts.1.stopswtc	Y
line.hosts.1.abncount	0000
line.start.0	1
line.start.1.type	MVS
line.start.1.max	50
line.start.1.timeout	5
line.start.1.cmd	S AAOCSME4
line.userprop.0	1
line.userprop.1.name	UP5
line.userprop.1.value	User defined property such as: my property at \$ 14 Mar 2003 \$ 13:50:38
line.verify.start	NO
line.verify.stop	NO

Table 9-7 shows an example of the variables and example values that the AOEXEC TOM GET command returns for a set.

**Table 9-7 REXX Stem Variables and Sample Values: Set**

REXX Stem Variable	Value Returned
line.desc	set1
line.status	STOPPED
line.member.0	2
line.member.1.name	Test Obj4
line.member.1.type	OBJECT
line.member.1.short	OBJ4
line.member.2.name	a1
line.member.2.type	OBJECT
line.member.2.short	A1

## Examples

Following are some examples of the syntax for the GET function:

```
drop line. /* drop all stem variables first */

AOEXEC TOM FUNCTION(GET) OBJECT('cicsprod.sysa')

AOEXEC TOM FUNCTION(GET) SET(SET.CICS.OBJS) STEM(MyStem)
      XCF(group1)
```

## Return Codes

Table 9-8 shows the possible return codes and what they mean.

**Table 9-8 Return Codes for the AOEXEC TOM GET Function**

Return Code	Explanation
0	Command successfully completed
8	Object(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
118	Security violation occurred
120	Product STM is absent
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing

## LIST Function

The LIST function returns a list of REXX stem variables for all existing objects or sets. You can use the optional STEM keyword to specify the REXX stem name. The default stem name is LINE.

The variables are returned as:

- LINE.0 which contains the number of objects or sets
- LINE.n which contains the name of each object or set

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(LIST)	TYPE(object set) [STEM(LINE)] [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
TYPE	Specify object or set name	
STEM	The STEM keyword is optional and specifies the prefix of the REXX stem variables that contain the properties of the object (or set)	The default REXX stem value is LINE
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

### Examples

Following are some examples of the syntax for the LIST function:

```
drop line. /* drop all Stem variables first */

AOEXEC TOM FUNCTION(LIST) TYPE(CICS) STEM(line)
  say 'line.0 = 'line.0

do i = 1 to line.0
  say 'line.'i '=' line.i '!'
end i
```

## Return Codes

Table 9-9 shows the possible return codes and what they mean.

**Table 9-9** Return Codes for the AOEXEC TOM LIST Function

<b>Return Code</b>	<b>Explanation</b>
0	Command successfully completed
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
118	Security violation occurred

## LOCK Function

The LOCK function corresponds to the Lock line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(LOCK)	OBJECT('name') SET('name') HOST(hostname) [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
HOST	Specify the name of an OS/390 system that is specified in the object definition's Valid Systems list.	Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

### Examples

Following are some examples of the syntax for the LOCK function:

```
AOEXEC TOM FUNCTION(LOCK) OBJECT('CICSPROD.sysg') HOST(sysa)
```

## Return Codes

Table 9-10 shows the possible return codes and what they mean.

**Table 9-10 Return Codes for the AOEXEC TOM LOCK Function**

<b>Return Code</b>	<b>Explanation</b>
0	Command successfully completed
4	OBJECT(name) is already Locked
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
116	HOST(hostname) is inactive or does not exist
118	Security violation occurred
121	HOST(hostname) is not in the Valid Systems list
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

## MOVE Function

The MOVE function corresponds to the Move line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The MOVE function has slightly different sets of parameters when you specify an object or a set.

The following table shows the command and parameters when you use the MOVE function with an object.

Command (with OBJECT specified)	Parameters
AOEXEC TOM FUNCTION(MOVE)	OBJECT('name') HOST(hostname) TOHOST(hostname2) [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table shows the command and parameters when you use the MOVE function with a set.

Command (with SET specified)	Parameters
AOEXEC TOM FUNCTION(MOVE)	SET('name') TOHOST(hostname2) [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.

Parameter	Function	Notes
HOST	Specify the name of an OS/390 system that is specified in the object definition's Valid Systems list to move the object from.	Do not use the Host() parameter when using SET().  Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
TOHOST	Specify the name of an OS/390 system that is specified in the object definition's Valid Systems list to move the object (or set) to.	Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

## Examples

Following are some examples of the syntax for the MOVE function:

```
AOEXEC TOM FUNCTION(MOVE) OBJECT('cicsprod.SYSB') HOST(sysa)
      TOHOST(sysb)
```

```
AOEXEC TOM FUNCTION(MOVE) SET(CICS.SET) TOHOST(sysb)
```

## Return Codes

Table 9-11 shows the possible return codes and what they mean.

**Table 9-11 Return Codes for the AOEXEC TOM MOVE Function (Part 1 of 2)**

Return Code	Explanation
0	Command successfully completed
4	HOST(hostname) and TOHOST(hostname) are the same
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
111	HOST(hostname) keyword is missing

**Table 9-11 Return Codes for the AOEXEC TOM MOVE Function  
(Part 2 of 2)**

<b>Return Code</b>	<b>Explanation</b>
112	TOHOST(hostname) keyword is missing
116	TOM hostname is inactive or does not exist
118	Security violation occurred
120	Product STM is absent
121	HOST(hostname) is not in the Valid Systems list
125	OBJECT(name) is not Active on HOST(hostname)
130	TOHOST(hostname) is not in the Valid Systems list
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

## RESET Function

The RESET function corresponds to the Reset line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(RESET)	OBJECT('name') SET('name') RESETOPT (ALL   [ABENDCT] [FORCED] [STARTCMDCT] [STATUS] [SUSPEND]) [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
RESETOPT	Specify either RESETOPT ALL or RESETOPT with one (or more) of the following: <ul style="list-style-type: none"> <li>• ABENDCNT</li> <li>• FORCED</li> <li>• STARTCMDCT</li> <li>• STATUS</li> <li>• SUSPEND</li> </ul> Enclose multiple keywords in parentheses ( ).	Specifying RESETOPT ALL includes the all of the other keywords. Otherwise, you can specify which definitions to reset. Specifying: <ul style="list-style-type: none"> <li>• ABENDCNT resets the number of abnormal termination events specified</li> <li>• FORCED resets the FORCED ACTIVE status</li> <li>• STARTCMDCT resets the start command count</li> <li>• STATUS resets the status</li> <li>• SUSPEND takes an object in or out of TOM's control</li> </ul>
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

## Examples

Following are some examples of the syntax for the RESET function:

```
AOEXEC TOM FUNCTION(RESET) OBJECT('mytest cics object') RESETOPT(ALL)
```

```
AOEXEC TOM FUNCTION(RESET) SET(SET.BACKUP) RESETOPT(ABENDCT STATUS)
```

## Return Codes

Table 9-12 shows the possible return codes and what they mean.

**Table 9-12 Return Codes for the AOEXEC TOM RESET Function**

Return Code	Explanation
0	Command successfully completed
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
113	ResetOpt() keyword is missing
118	Security violation occurred
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

## SETPROPERTY Function

You can use the SETPROPERTY function to create a property to associate with an object or set. You can also use SETPROPRRTY to delete a user-defined property. The value of the user-defined property can be up to 1024 bytes.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(SETPROPERTY)	OBJECT('name') SET('name') NAME(user_defined_property) DELETE VARIABLE(REXX_variable_name [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
NAME	Specify a 1 to 8-character user-defined name of a property.	This property can be used on the Started task Definition panel in the Dependency Property Value definition. The name of the property is not case sensitive.
VARIABLE	Specify the name of a REXX variable that contains the user property value.	The value can be up to 1024 bytes long.  Mutually exclusive with use of DELETE parameter. Do not specify the VARIABLE parameter with the DELETE parameter.
DELETE	Specify this parameter when you want to delete the user property you previously created.	Mutually exclusive with use of VARIABLE parameter. Do not specify the DELETE parameter with the VARIABLE parameter.

Parameter	Function	Notes
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

## Examples

Following are some examples of the syntax for the SETPROPERTY function:

```
AOEXEC TOM FUNCTION(SETPROPERTY) OBJECT('cicsprod.sysa') NAME(myProp)
    VARIABLE(variableName)
AOEXEC TOM FUNCTION(SETPROPERTY) SET(SET.BACKUP) NAME(myProp2) DELETE
```

## Return Codes

Table 9-13 shows the possible return codes and what they mean.

**Table 9-13 Return Codes for the AOEXEC TOM SETPROPERTY Function**

Return Code	Explanation
0	Command successfully completed
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
118	Security violation occurred
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
137	Both DELETE and VARIABLE() were specified
138	NAME(name) was missing
139	Both DELETE and VARIABLE() were missing
140	Variable v1 is invalid or non-existent
141	Variable v1 contains a value that is too long

## SHUTSET Function

The SHUTSET function corresponds to the H line command which can be issued from the Started Task Sets panel. This command causes an immediate and orderly shutdown of all objects within a set.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(SHUTSET)	SET('name') [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
SET	Specify the set name	If the set name includes lower case letters or blanks, be sure to include the set name in single quotation marks.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

### Examples

Following are some examples of the syntax for the SHUTSET function:

```
AOEXEC TOM FUNCTION(SHUTSET) Set (SET.BACKUP.OBJECTS)
```

### Return Codes

Table 9-14 shows the possible return codes and what they mean.

**Table 9-14 Return Codes for the AOEXEC TOM SHUTSET Function**

Return Code	Explanation
0	Command successfully completed
115	SET(name) does not exist
118	Security violation occurred
143	The specified SET does not exist

## SHUTSYS Function

The SHUTSYS function corresponds to the H line command which can be issued from the TOM Managed Systems panel. This command causes an immediate and orderly shutdown of all objects on the system that is running TOM.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(SHUTSYS)	HOST(hostname) [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
HOST	Specify the name of an OS/390 system where all the objects should be shut down.	Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

### Examples

Following are some examples of the syntax for the SHUTSYS function:

```
AOEXEC TOM FUNCTION(SHUTSYS) HOST(sysA) TOM(sysA)
```

```
AOEXEC TOM FUNCTION(SHUTSYS) HOST(sysA)
```

### Return Codes

Table 9-15 shows the possible return codes and what they mean.

**Table 9-15 Return Codes for the AOEXEC TOM SHUTSYS Function**

Return Code	Explanation
0	Command successfully completed
116	TOM hostname is inactive or does not exist
118	Security violation occurred

## START Function

The START function corresponds to the Start line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(START)	OBJECT('name') SET('name') HOST(hostname) [FULLAUTO] [RESNEXTIPL] [BYPASSCMDCT] [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
HOST	Specify the name of an OS/390 system that is specified in the object definition's Valid Systems list to start the object on.	Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
FULLAUTO	Specifies to start the object with full automation where any Pre or Post-Start EXECs will also be scheduled as part of starting the object.	This is an optional parameter. When this parameter is not specified, these EXECs are not scheduled.
RESNEXTIPL	This parameter is equivalent to the Reset at Next IPL option for starting an object. Specifying this parameter takes the object out of Forced state after the next IPL and resumes control based on the object's defined schedule and other object definitions.	This is an optional parameter. When this parameter is not specified, the object remains in this Forced state through subsequent system IPLs.

Parameter	Function	Notes
BYPASSCMDNT	This parameter is equivalent to the Bypass Command Count Check option for starting an object. Specifying this parameters means that TOM bypasses checking whether the command count specified in the object's definition (on the <b>Maximum Start Count</b> field of the Started Task Definition panel) has been exceeded.	This is an optional parameter.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

## Examples

Following are some examples of the syntax for the START function:

```
AOEXEC TOM FUNCTION(START) OBJECT('object 2') HOST(sysA)
FULLAUTO RESNEXTIPL
```

```
AOEXEC TOM FUNCTION(START) SET(SET.CICS) HOST(sysA)
FULLAUTO RESNEXTIPL
```

## Return Codes

Table 9-16 shows the possible return codes and what they mean.

**Table 9-16 Return Codes for the AOEXEC TOM START Function (Part 1 of 2)**

Return Code	Explanation
0	Command successfully completed
4	OBJECT(name) is already Active on HOST(hostname)
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred

**Table 9-16 Return Codes for the AOEXEC TOM START Function  
(Part 2 of 2)**

<b>Return Code</b>	<b>Explanation</b>
111	HOST(hostname) keyword is missing
116	TOM hostname is inactive or does not exist
118	Security violation occurred
120	Product STM is absent
121	HOST(hostname) is not in the Valid Systems list
122	Name(hostname) is a model, cannot start a model
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

## STOP Function

The STOP function corresponds to the Stop line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(STOP)	OBJECT('name') SET('name') HOST(hostname) [FULLAUTO] [RESNEXTIPL] [[TOMID(tom id)   XCFGROUP(XCF group name)]]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
HOST	Specify the name of an OS/390 system that is specified in the object definition's Valid Systems list to stop the object on.	Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
FULLAUTO	Specifies to start the object with full automation where any Pre or Post-Start EXECs will also be scheduled as part of starting the object.	This is an optional parameter. When this parameter is not specified, these EXECs are not scheduled.
RESNEXTIPL	This parameter is equivalent to the Reset at Next IPL option for starting an object. Specifying this parameter takes the object out of Forced state after the next IPL and resumes control based on the object's defined schedule and other object definitions.	This is an optional parameter. When this parameter is not specified, the object remains in this Forced state through subsequent system IPLs.

Parameter	Function	Notes
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

## Examples

Following are some examples of the syntax for the STOP function:

```
AOEXEC TOM FUNCTION(STOP) OBJECT('cicsprod.sysC)
HOST(sysC)FULLAUTO
```

```
AOEXEC TOM FUNCTION(STOP) SET(BACKUP.SET) RESNEXTIPL
XCFGROUP(group1)
```

## Return Codes

Table 9-17 shows the possible return codes and what they mean.

**Table 9-17 Return Codes for the AOEXEC TOM STOP Function (Part 1 of 2)**

Return Code	Explanation
0	Command successfully completed
4	OBJECT(name) is not Active on HOST(hostname)
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
111	HOST(hostname) keyword is missing
116	TOM hostname is inactive or does not exist
118	Security violation occurred
120	Product STM is absent
121	HOST(hostname) is not in the Valid Systems list
135	Both OBJECT(name) and SET(name) were specified

**Table 9-17**      **Return Codes for the AOEXEC TOM STOP Function  
(Part 2 of 2)**

<b>Return Code</b>	<b>Explanation</b>
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

## SUSPEND Function

The SUSPEND function corresponds to the Suspend line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(SUSPEND)	OBJECT('name') SET('name') [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

### Examples

Following are some examples of the syntax for the SUSPEND function:

```
AOEXEC TOM FUNCTION(SUSPEND) OBJECT('CICSprod.sysA')
AOEXEC TOM FUNCTION(SUSPEND) SET('set.production')
```

**Return Codes**

Table 9-18 shows the possible return codes and what they mean.

**Table 9-18 Return Codes for the AOEXEC TOM SUSPEND Function**

<b>Return Code</b>	<b>Explanation</b>
0	Command successfully completed
4	OBJECT(name) is already Suspended
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
118	Security violation occurred
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

## SYSINFO Function

You can use the SYSINFO function to request and receive information (in variables) about all TOM systems that are running on an OS/390 system, regardless of the XCF group to which the OS/390 system belongs.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(SYSINFO)	XCFGROUP(XCF group name)]

When the AOEXEC TOM SYSINFO function is specified, the following variables are returned:

- TOM.0 represents the number of TOM systems that were found.
- TOM.x represents the ID of the TOM system that was found.
- TOMXCF.x represents the name of the XCF group to which this TOM belongs.

The following table describes the parameters:

Parameter	Function	Notes
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Use the XCFGROUP parameter option when you want to restrict the search to a specific group name. For example, if you want to find only TOMs belonging to the default BMCTOM group name, specify: XCFGROUP(BMCTOM).

## Examples

Following are some examples of the syntax for the SYSINFO function:

```
AOEXEC TOM FUNCTION(SYSINFO)
```

```
AOEXEC TOM FUNCTION(SYSINFO) XCFGROUP(BMCTOM)
if rc = 0 then
  do
    do i = 1 to tom.0
      say 'tom' i '=' tom.i 'xcfg =' tomxcf.i
    end i
  end
end
```

**Return Codes**

Table 9-19 shows the possible return codes and what they mean.

**Table 9-19 Return Codes for the AOEXEC TOM SYSINFO Function**

<b>Return Code</b>	<b>Explanation</b>
0	Command successfully completed
4	No Active TOMs available on this MVS
12	Internal error occurred
16	Syntax error occurred
20	Internal error occurred
24	Internal error occurred
28	Name(name) was absent

## UNBLOCK Function

The UNBLOCK function corresponds to the Unblock line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(UNBLOCK)	OBJECT('name') SET('name') HOST(hostname) [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
HOST	Specify the name of an OS/390 system that is specified in the object definition's Valid Systems list.	Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

## Examples

Following are some examples of the syntax for the UNBLOCK function:

```
AOEXEC TOM FUNCTION(UNBLOCK) OBJECT('DB2 object')  
HOST(sysa)
```

```
AOEXEC TOM FUNCTION(UNBLOCK) SET(SET.TEST.OBJECTS)  
HOST(sysb) XCF(group3)
```

## Return Codes

Table 9-20 shows the possible return codes and what they mean.

**Table 9-20 Return Codes for the AOEXEC TOM UNBLOCK Function**

Return Code	Explanation
0	Command successfully completed
4	OBJECT(name) currently not Blocked
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
118	Security violation occurred
121	HOST(hostname) is not in the Valid Systems list
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

## UNLOCK Function

The UNLOCK function corresponds to the Unlock line command, which can be issued for objects from the Started Task Overview panel and for sets from the Started Task Sets panel.

The following table shows the command and parameters:

Command	Parameters
AOEXEC TOM FUNCTION(UNLOCK)	OBJECT('name') SET('name') HOST(hostname) [TOMID(tom id)   XCFGROUP(XCF group name)]

The following table describes the parameters:

Parameter	Function	Notes
OBJECT	Specify the object name	Mutually exclusive with use of SET parameter.  For important information describing specifying the object name, refer to "Object and Set Naming Restrictions" on page 9-3.
SET	Specify the set name	Mutually exclusive with use of OBJECT parameter.  For important information describing specifying the set name, refer to "Object and Set Naming Restrictions" on page 9-3.
HOST	Specify the name of an OS/390 system that is specified in the object definition's Valid Systems list.	Refer to "Defining Valid Systems for the Object to Start On" on page 3-16 for more information about the Valid Systems list.
TOMID	Specify the ID of the TOM system that you want the command issued on	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.
XCFGROUP	Specify the name of the XCFGROUP to which the TOM belongs to	Refer to "How to Use the TOMID and XCFGROUP Parameters" on page 9-3 for important usage information.

### Examples

Following are some examples of the syntax for the UNLOCK function:

```
AOEXEC TOM FUNCTION(UNLOCK) OBJECT('DB2 object') HOST(sysa)
```

```
AOEXEC TOM FUNCTION(UNLOCK) SET(SET.TEST.OBJECTS) HOST(sysa)
```

**Return Codes**

Table 9-21 shows the possible return codes and what they mean.

**Table 9-21 Return Codes for the AOEXEC TOM UNLOCK Function**

<b>Return Code</b>	<b>Explanation</b>
0	Command successfully completed
4	OBJECT(name) currently not Locked
8	OBJECT(name) or SET(name) does not exist
16	Syntax error occurred
17	FUNCTION keyword is missing
100	No active TOMs meet your criteria
101	Internal error occurred
102	TOM is not active
103	Internal error occurred
118	Security violation occurred
121	HOST(hostname) is not in the Valid Systems list
135	Both OBJECT(name) and SET(name) were specified
136	Both OBJECT(name) and SET(name) were missing
142	Internal error occurred

# TOM Console Commands

TOM provides console commands that you can use to perform object-management actions from an MVS console. The general format of the console command is

**F *tom*,*CMD commandOptions***

where *tom* is the name of your TOM procedure or identifier and *commandOptions* user-specified, optional parameters.

## Coding Conventions

Following are some general formatting rules:

- If an object or set name contains lowercase characters, you *must* enclose the name within single quotation marks in the syntax of your TOM console command. At an MVS console, for example, you can issue

```
F TOM,CMD activate object('cicsprod.sysa')
```

In the preceding command, `lower case object name` represents an object name that contains lowercase characters. When you do not use single quotation marks, the object or set name is translated to uppercase characters.

- To issue commands with a mixed-case object or set name using the Sysout Display and Search Facility (SDSF), enter a single forward slash (/) on the Command line to invoke the System Command Extension pop-up panel.

On this pop-up panel, you can issue commands with mixed characters, such as

```
F TOM,CMD activate object('cicsPROD.sysB')
```

In the preceding command, `mixed CASE object NAME`, entered within single quotation marks, is preserved and processed in mixed case. The other parts of the command are translated as uppercase characters.

**Note:** On OS/390 version 2.9 and earlier, the SDSF application does not allow you to issue MVS commands containing lowercase characters.

The message CK1005I echoes the command that is received by the TOM system. You can search for and view this message to see which portions have been processed as uppercase characters.

To issue commands from a program or EXEC, use the TOM functions as described in “Command Syntax for AOEXEC TOM Functions” on page 9-5. Do not use the console commands. The AOAnywhere TOM API provides return codes, which are much easier to process and more efficient than text responses.

## List of TOM Console Commands

The TOM console commands are as follows:

- **ACTIVATE** places an object (or set of objects) into TOM’s control.
- **BLOCK** makes the system unavailable for scheduling an object (or set) until the object or set status is manually reset. The status of the object is **BLOCKED** on the selected system.
- **BOUNCE** quickly brings down and re-starts an object.
- **CHSTATUS** changes the status of an object. Use the **CHSTATUS** command against an object with a status of **ACTIVE** or **ACTIVE-ERR** to change the object status to **STOPPED**. When the command is issued against an object with a status other than **ACTIVE** or **ACTIVE-ERR**, the object status changes to **ACTIVE**.
- **LOCK** makes the system unavailable for scheduling until the next system IPL. The status of the object is **LOCKED** on the selected system.
- **MOVE** moves an object (or set) from one system to another system.
- **RESET** resets the state of an object to **ACTIVE** or **STOPPED**.
- **SHUTSET** shuts down objects within sets in an orderly way.
- **SHUTSYS** shuts down all objects in an orderly way.
- **START** starts an object (or set).
- **STOP** stops an object (or set).
- **SUSPEND** resets an object (or set) to **SUSPEND** control.
- **UNBLOCK** reverses the **BLOCK** command.
- **UNLOCK** reverses the **LOCK** command.

Table 9-22 describes full command syntax and provides notes about the syntax notation.

**Table 9-22 List of TOM Console Commands**

<b>TOM Console Command</b>	<b>Format of the Console Command</b>	<b>Notes</b>
ACTIVATE	F tom,CMD ACTIVATE OBJECT('n1') SET('n1')	<ul style="list-style-type: none"> <li>n1 represents the name of an object or set</li> </ul>
BLOCK	F tom,CMD BLOCK OBJECT('n1') SET('n1') Host(h1)	<ul style="list-style-type: none"> <li>n1 represents the name of an object or set</li> <li>h1 represents the name of a system listed in the object's (or set's) Valid Systems list</li> </ul>
BOUNCE	F tom,CMD BOUNCE OBJECT('n1') Host(h1) [FullAuto] F tom,CMD BOUNCE SET('n1') [FullAuto]	<ul style="list-style-type: none"> <li>n1 represents the name of an object or set</li> <li>h1 represents the name of a system listed in the object's (or set's) Valid System list</li> <li>FULLAUTO is an optional keyword that indicates whether TOM should bounce the object and schedule Pre-start/Post-start EXECs and the Pre-stop/Post-stop EXECs with the object.</li> </ul>
CHSTATUS	F tom,CMD CHSTATUS OBJECT('n1') Host(h1)	<ul style="list-style-type: none"> <li>n1 represents the name of an object or set</li> <li>h1 represents the name of a system listed in the object's (or set's) Valid System list</li> </ul>
LOCK	F tom,CMD LOCK OBJECT('n1') SET('n1') Host(h1)	<ul style="list-style-type: none"> <li>n1 represents the name of an object or set</li> <li>h1 represents the name of a system listed in the object's (or set's) Valid System list</li> </ul>
MOVE	F tom,CMD MOVE OBJECT('n1') HOST(h1) TOHOST(h2) F tom,CMD MOVE SET('n1') TOHOST(h2)	<ul style="list-style-type: none"> <li>n1 represents the name of an object or set</li> <li>h1 represents the name of a system listed in the object's (or set's) Valid System list to move the object or set from</li> <li>h2 represents the name of a system listed in the object's (or set's) Valid System list to move the object or set to</li> </ul>

**Table 9-22 List of TOM Console Commands**

<b>TOM Console Command</b>	<b>Format of the Console Command</b>	<b>Notes</b>
RESET	F tom,CMD RESET OBJECT('n1') SET('n1') RESETOPT([ALL   [ABENDCT] [FORCED] [STARTCMDCT] [STATUS] [SUSPEND])	<ul style="list-style-type: none"> <li>n1 represents the name of an object or set</li> </ul> <p>Refer to the syntax description for the AOEXEC TOM REST command for information about the RESETOPT parameters in the section "RESET Function" on page 9-25.</p>
SHUTSET	F tom,CMD SHUTSYS SET(n1)	<ul style="list-style-type: none"> <li>n1 is the name of a set</li> </ul>
SHUTSYS	F tom,CMD SHUTSYS HOST(h1)	<ul style="list-style-type: none"> <li>h1 represents the name of an OS/390 system listed in the object's (or set's) Valid System list to move the object or set from</li> </ul>
START	F tom,CMD START OBJECT('n1') SET('n1') HOST(h1) [FULLAUTO] [RESNEXTIPL][BYPASSCMDCT]	<ul style="list-style-type: none"> <li>n1 represents the name of an object or set</li> <li>h1 represents the name of a system listed in the object's (or set's) Valid System list</li> <li>FULLAUTO is an optional keyword that indicates whether TOM should bounce the object and schedule the Pre, Post-start EXECs and the Pre, Post-Stop EXECs with the object.</li> <li>RESNEXTIPL is an optional keyword that takes the object out of Forced state after the next IPL and resumes control based on the object's defined schedule and other object definitions.</li> <li>BYPASSCMDCNT is an optional keyword where TOM bypasses checking whether the command count specified in the object's definition. In addition, an error does not occur when the command count has been exceeded. When the start command is issued, however, increments the command counter. Reset the command counter after the start command is issued, if indicated to do so in the object's definition.</li> </ul>

Table 9-22 List of TOM Console Commands

TOM Console Command	Format of the Console Command	Notes
STOP	F tom,CMD STOP OBJECT('n1') Host(h1) [FULLAUTO] [RESNEXTIPL] F tom,CMD STOP SET('n1') [FULLAUTO] [RESNEXTIPL]	<ul style="list-style-type: none"> <li>• n1 represents the name of an object or set</li> <li>• h1 represents the name of a system listed in the object's (or set's) Valid System list</li> <li>• FULLAUTO is an optional keyword that indicates whether TOM should bounce the object and schedule the Pre, Post-start EXECs and the Pre, Post-Stop EXECs with the object.</li> <li>• RESNEXTIPL is an optional keyword that takes the object out of Forced state after the next IPL and resumes control based on the object's defined schedule and other object definitions.</li> </ul>
SUSPEND	F tom,CMD SUSPEND OBJECT('n1') SET('n1')	<ul style="list-style-type: none"> <li>• n1 represents the name of an object or set</li> </ul>
UNBLOCK	F tom,CMD UNBLOCK OBJECT('n1') SET('n1') Host(h1)	<ul style="list-style-type: none"> <li>• n1 represents the name of an object or set</li> <li>• h1 represents the name of a system listed in the object's (or set's) Valid System list</li> </ul>
UNLOCK	F tom,CMD UNLOCK OBJECT('n1') SET('n1') Host(h1)	<ul style="list-style-type: none"> <li>• n1 represents the name of an object or set</li> <li>• h1 represents the name of a system listed in the object's (or set's) Valid System list</li> </ul>

Additional console commands are listed in Table 9-23.

**Warning!** Note that DEBUG parameter for any of the applicable console commands may produce high volumes of messages to either the CONSOLE, SYSLOG, or TOMLOG. You should also be sure to turn off the DEBUG feature after gathering sufficient documentation. Use this parameter ONLY upon instructions of BMC Software personnel.

**Table 9-23 List of Additional TOM Console Commands**

TOM Console Command	Valid Parameters	Notes
XCF	DEBUG[ON OFF START STOP] DISPLAY	<p>The DEBUG parameter turns intersystem communication debugging on or off.</p> <p>The DISPLAY parameter shows the status of intersystem communications such as the TOMs and MVS hosts that this TOM is connected with.</p>
REGISTRY	DEBUG[ON OFF START STOP] DISPLAY CLONETO DUMP DD()	<p>The DEBUG parameter turns registry debugging on or off.</p> <p>The DISPLAY parameters causes various pieces of information about the registry to be displayed.</p> <p>CLONETO allows you to clone one TOM registry to another registry. However, under normal circumstances, TOM is continuously updating all the TOM registries to ensure all the TOM are using the latest data. Use this command only upon the advice of BMC Software personnel because you risk overlaying valid data and corrupting the Registry for all TOMs.</p> <p>DUMP DD() writes an internal dump of the registry to the data set allocated to the TOM address space under the given DD name. Recommended DCB is VB, LRECL=8192,BLKSIZE=32760 (or anything reasonable). The command can be used to enlarge or decrease the size of the TOM registry. When first using the DUMP command and subsequently the PRIMEDD() keyword in REGSTY00 the registry can be reloaded into a fresh data set.</p> <p>Note that such a procedure is not necessary if the registry size was changed and then its TOM was brought up as second or later TOM in a sysplex. In this case the fresh (and empty) registry will automatically be updated from the running TOMs.</p>

Table 9-23 List of Additional TOM Console Commands

TOM Console Command	Valid Parameters	Notes
OBJECT	DEBUG[ON OFF START STOP] SHOW EVALUATE	<p>The DEBUG parameter turns object manager debugging on or off.</p> <p>The SHOW parameter dumps the status of all objects to the syslog</p> <p>The EVALUATE parameter forces an object evaluation cycle. This identifies any anomalies not covered by the normal processing. Normally this command is not necessary because TOM goes through this process every minute automatically. However, it may be beneficial during testing to drive the cycle instead of waiting for a minute to occur automatically. This is an equivalent of the EVAL command on the Started Task Overview panel.</p>
LOGGER	DEBUG[ON OFF START STOP] SWITCH DISPLAY	<p>The DEBUG parameter turns log processing debugging on or off.</p> <p>The SWITCH parameter forces a log switch between TOM log 1 and TOM log 2</p> <p>The DISPLAY parameter shows statistical information about log address space usage.</p>
SHUTDOWN	DISPLAY	<p>The DISPLAY parameter displays shutdown status. Shows active objects, objects to be shut down and whether any shutdown conflicts exists and which objects are involved.</p>
TOM	DUMPS[ON OFF START STOP] SHOWBLOCKS	<p>The DUMPS parameter turns on &amp; off the ability to automatically generate a diagnostic dump in the event of a failure.</p> <p>SHOWBLOCKS displays the location, size and PTF level of the following control blocks: CND, ICD, TMD, TOM. This information can be used to display the control blocks in real time for diagnostic purposes.</p>



---

---

# Chapter 10 Converting the Continuous State Manager Version 6 Repository

This chapter describes how to use the TOM conversion tool to convert a MAINVIEW AutoOPERATOR Continuous State Manager (CSM) version 6 object repository that can be used by TOM application.

This chapter includes the following topics:

Migration Considerations . . . . .	10-2
Accessing the Conversion Utility . . . . .	10-5
Exception Messages . . . . .	10-11

## Migration Considerations

If you currently use the MAINVIEW AutoOPERATOR CSM to manage objects, the TOM application provides a conversion utility.

The conversion utility works with only the following versions of CSM:

- 6.1.00
- 6.2.00
- 6.3.00
- 6.3.01

The utility attempts to convert the CSM object repository into TOM object definitions.

### Disabling CSM

BMC Software strongly recommends that you disable CSM before starting and making TOM active for a group of newly converted CSM objects.

### Using Test Mode

For the first time that you attempt to convert a CSM repository, BMC Software strongly recommends that you use the conversion utility to specify

Test Mode: YES

This specification allows the utility to attempt the conversion without creating TOM object definitions. Any exceptions to the conversion process are written to the Conversion Log.

Review all information in the Conversion Log. This log provides information about CSM functions that were not converted for TOM and provides advice about how you can manually handle exceptions to conversion processing.

---

## Defining a Valid System, Definition Base Name, and Initial Control Setting

During the conversion, you are asked, for each CSM group that you want to convert, to define settings for

- the name of a Valid System on which to run each converted CSM group of objects
- the name of a TOM Definition Base in which to store the newly converted objects
- the initial Control setting for the objects.

Possible choices are ACTIVE or SUSPEND. If you select ACTIVE, the newly converted objects are activated by TOM when the Definition Base is activated.

**Note:** BMC Software recommends that you select SUSPEND, where the objects are not active when the Definition Base is activated. You should then manually activate each object.

## Selecting a Name Pattern

Each CSM object that is converted is given a new object name. The conversion utility uses the following default naming pattern:

*objectName.sysName*

These naming nodes are defined as follows:

- *objectName* represents the CSM name of the object.
- *sysName* represents the name of the Valid System on which the object will run.

You can also use the node *sysplex* in the naming pattern, which resolves to the name of the sysplex in the converted object name. For example the following pattern resolves to the CSM object name as the first node and the sysplex name as the second node:

*objectName.sysplex*

The nodes for the object names can be entered in any order. For example the following configuration means that the Valid System name appears before the CSM object name when the name nodes are resolved by the conversion:

*sysName . objectName*

## Definitions That Are Not Converted

Not every object definition from CSM is automatically converted by the utility. Table 10-1 on page 11 and Table 10-2 on page 11 list all possible messages from the conversion process and provide brief descriptions of what the messages mean.

# Accessing the Conversion Utility

To access the conversion utility:

- Step 1** On the TOM Primary Options Menu, enter **A** (for Administration) on the **Option** line (Figure 10-1).

**Figure 10-1** Accessing the Conversion Utility

```

BMC Software                TOM Primary Options Menu

Option ==>  A

1 Started Task    - Define, display and manage objects      User ID: BAOMXY1
2 Sets           - Define, display and manage sets          Server Id: TOMN1
3 Calendar       - Define, display and manage calendars     System: SJSD

A Administration - System Administration
M Messages      - List messages
L Log           - Display Log

X Exit          - Exit

                                Copyright BMC Software, Inc. 2003

```

The Administration Options Menu is displayed (Figure 10-2).

**Figure 10-2** Administration Options Menu

```

                                Administration Options Menu

Option ==>

1 Systems        - Systems mode operations
2 Definitions    - Define, display and manage definitions
3 Conversion     - Convert CSM V6 Repository
4 Layers        - Layer Management

X Exit          - Exit

```

- Step 2** On the **Option** line, enter **3** (for Conversion).

The CSM v6 to TOM Conversion panel is displayed (Figure 10-3).

**Figure 10-3 CSM v6 to TOM Conversion Panel**

```

                                CSM v6 to TOM Conversion
Command ==>

Enter YES to test the repository without modifying it. Enter NO to convert.

    Test Mode: YES  ( YES or NO )

Specify the name of the CSM v6 repository data set containing the CSM objects
to be converted into TOM format and stored in the registry for TOMN1

Data Set Name:

Specify the name of the data set where the Conversion Log will be written.
If the data set does not already exist, it will be allocated dynamically.

Data Set Name:

Disposition:      ( SHR or MOD )

Press ENTER to continue or CANCEL to exit
    
```

**Step 3** In the **Test Mode** field, enter **YES**.

The conversion attempt does not actually produce any TOM objects from the CSM object repository. Test mode shows how many CSM objects can be converted to TOM objects. You can review the conversion log to see which CSM objects might require manual conversion.

**Step 4** In the first **Data Set Name** field, enter the name of the CSM repository.

**Step 5** In the second **Data Set Name** field, enter the name of the data set that you want the conversion utility to write the log to.

**Step 6** In the **Disposition** field, enter **SHR** to overwrite a pre-existing conversion log data set or enter **MOD** to append the log information to an existing log data set.

**Step 7** Press **Enter** to continue.

A panel is displayed (Figure 10-4), explaining that, for each CSM group that you want to convert, you must define

- the name of a Valid System on which the objects will run
- the name of a TOM Definition Base to use to store the newly converted objects

- the initial Control setting for each object (ACTIVE or SUSPEND)

**Note:** If you select ACTIVE, the newly converted objects are activated by TOM when the Definition Base is activated. BMC Software recommends that you select SUSPEND and then manually activate each object.

**Figure 10-4 CSM v6 to TOM Conversion Panel (Second Panel)**

```
CSM v6 to TOM Conversion

Command ==>

Please review the following information before proceeding:
The next panel will display the CSM Groups in the requested repository. For
each Group that you wish to convert, specify a Valid System, Definition Base
and Control setting.

Valid System: The MVS sysname where the CSM Group will be enabled.

Definition Base: The name of the Object Base where the objects from the CSM
group are to be stored. Since only one TOM Plex can be active at a time,
use the same Definition Base for all CSM Groups which will be managed by
the same TOM Plex. Specify a unique Definition database for any CSM Group
which is to be used an Alternative database.

Control: Specify whether the objects in the Group should be activated by TOM
as soon as the Definition Base becomes active. If you wish to activate the
objects in the Group immediately after the conversion, set Control active
for the currently active Definition Base.

Press ENTER to continue or CANCEL to exit
```

**Step 8** Press Enter to continue.

Another conversion panel is displayed (Figure 10-5).

**Figure 10-5 CSM v6 to TOM Conversion Panel (Third Panel)**

```

                                CSM v6 to TOM Conversion
Command ==>
      Server ID: TOMN1
Active Definition Base: DEFAULT

Enter a Valid System, Definition Base and Control for the CSM Groups
that you wish to convert.

      Valid
Group  System      Definition Base      Control
NS63   _____  _____  _____
_____  _____  _____  _____
_____  _____  _____  _____
_____  _____  _____  _____
_____  _____  _____  _____
_____  _____  _____  _____
_____  _____  _____  _____
_____  _____  _____  _____
_____  _____  _____  _____

Press ENTER to continue or CANCEL to exit
    
```

**Step 9** Enter the Valid System name, Definition Base name, and initial Control setting.

**Step 10** Press **Enter** to continue.

Another conversion panel is displayed (Figure 10-6).

**Figure 10-6 CSM v6 to TOM Conversion Panel (Fourth Panel)**

```

                                CSM v6 to TOM Conversion
Command ==>

TOM stores objects by default using the pattern: object name and system
name separated by a period e.g. TSO.SYS1

If you wish TOM to use a different pattern (such as object.sysname.sysplex)
enter it below. To accept the default pattern, leave the field blank and
press ENTER to continue.

The pattern can contain the following nodes: SYSPLEX, SYSNAME and OBJECT.

      Object name pattern:

Press ENTER to continue or CANCEL to exit
    
```

**Step 11** In the **Object name pattern** field, enter the pattern that the new TOM objects will have.

For information about entering a name pattern for newly converted TOM objects, refer to “Selecting a Name Pattern” on page 10-3.

**Step 12** Press **Enter** to continue.

During the conversion, a pop-up panel displays how many groups are being converted. When the conversion is complete, the TOM Conversion Log is displayed. Figure 10-7 shows an example of the TOM Conversion Log.

The TOM Conversion Log provides valuable information about the newly converted TOM objects names and about what some of the CSM object attributes are now as TOM objects.

**Figure 10-7 Conversion Log Panel**

```

BROWSE      BBTEST.CONVER.LOG                      Line 00000000 Col 001 079
Command ==>                                     Scroll ==> PAGE
***** Top of Data *****
BMC Software                      TOM Conversion Log

Input CSM repository data set: 'BBTEST.TEST.BBIREPS'
Converted by ..... TESTER1 at 12:43 on 08/29/2003
TOM Server..... TOMN1

Conversion status..... Test Mode!!! CSM Objects not converted.

Group: TN03 Valid System: SYSB Definition Base: default
=====

General Group exceptions:
- Enable Plex not converted. TOM to TOM communication can be enabled by
  specifying the same XCF Group in TOMPARM members XCFDEFxx.
- Alert Queues not converted. TOM issues exception messages to the TOM Log.
  These messages can be captured with JRNL Rule(s) and converted to Alerts.
- Statistics data set not converted. Feature not available in TOM.

Objects:  Exception Notes:
-----  -----
*BBISS   - Object renamed to BBISS.SYSB
*BBISS   - Cancel command converted to Stop Retry command.
*BBISS   - Start and Stop events converted to Startup and Shutdown validation
*BBISS   Rules. The TOM validation Rules currently do not support JOB Type.
-----
*APPC    - Object renamed to APPC.SYSB
*APPC    - Cancel command converted to Stop Retry command.
*APPC    - Start and Stop events converted to Startup and Shutdown validation
*APPC    Rules. The TOM validation Rules currently do not support JOB Type.
-----
*ASCH    - Object renamed to ASCH.SYSB
*ASCH    - Cancel command converted to Stop Retry command.
*ASCH    - Start and Stop events converted to Startup and Shutdown validation
*ASCH    Rules. The TOM validation Rules currently do not support JOB Type.
-----
*JES2    - Object renamed to JES2.SYSB
*JES2    - Cancel command converted to Stop Retry command.

```

Refer to Table 10-1 and Table 10-2 for all possible messages that are returned from the conversion process and a brief description of what the messages mean.

# Exception Messages

Table 10-1 and Table 10-2 list all possible exception messages that you might find in the Conversion Log after converting a CSM object repository to a TOM Definition Base.

**Table 10-1 Group Exception Messages**

General Group Exception Messages	Additional Explanation
Enable Plex not converted. Specifying the same XCF Group in TOMPARM members XCFDEFxx can enable TOM to TOM communication.	When you receive this message, you can edit the TOMPARM member XCFDEFxx in the data set specified on the TOMPARM DD statement listed in the TOM address space JCL. For example, in XCFDEFxx you can specify XCFGROUP named BMCTOM1 as: GROUP(BMCTOM1).
Alert Queues not converted. TOM issues exception messages to the TOM Log. These messages can be captured with JRNL Rule(s) and converted to Alerts.	There is no equivalent feature in the TOM application for the CSM Alert queues. As an alternative, you can create JRNL message-initiated Rules in MAINVIEW AutoOPERATOR where the Rule fires for various TOM exception messages and creates an Alert.
Statistics data set not converted. Feature not available in TOM.	There is no equivalent feature in the TOM application for the CSM statistics data set.

Table 10-2 lists all exception messages that you might receive after converting. These messages are specific to object definition differences between CSM and TOM objects.

**Table 10-2 Object Exception Messages**

Objects Exception Messages	Additional Explanation
Object renamed to <new_object_name> according to the selected Pattern.	This message appears for each object converted and shows the new name of the object based on the object name pattern that you selected (see Figure 10-6).
Object already exists. Renamed to <old_object_name>.1	It is possible that during the conversion, the utility discovers that more than CSM object would result in having the same object name in the TOM definition base. Because in TOM you may not have 2 objects with the same name, the utility names subsequent objects of the same name with a numeric appended (for example: TSO.SYSB.1, TSO.SYSB.2, and so on).
Grouping object detected. Object not converted. Create a Set and include all objects that you wish to start or stop with a single command.	TOM does not use Grouping objects. Instead, TOM uses sets where you can collect objects (and other sets of objects) together. You can create a set in TOM (and use the same name as your CSM Grouping object if you choose to) and add the desired objects into the new TOM set.
The Start (also for Stop/Cancel) command for this object contains CSM variable. CSM variables are not supported in TOM. Please update the object and replace all &CSM variables with shared variables.	There is no equivalent in the TOM application for CSM variables. You can replace the use of CSM variable with shared variables.

Table 10-2 Object Exception Messages

Objects Exception Messages	Additional Explanation
Cancel command converted to Stop Retry command.	There is no Cancel command for objects in the TOM application. The use of the CSM Cancel command can be replaced with the Stop Retry command in TOM because the TOM Stop Retry command performs the same function as the Cancel in CSM (the TOM Stop Retry command is issued only if the Stop command fails).
Start and Stop events converted to Startup and Shutdown validation Rules. The TOM validation Rules currently do not support JOB Type.	This is an informational message that explains that the CSM events are now called Validations in TOM. The CSM Rules that used the additional fields JOB type, Word in Message, Operand and Text string are not supported in the TOM Rules.
Abnormal Termination events converted to Abnormal Termination validation Rules. The TOM validation Rules currently do not support JOB Type, Word in message, Operand and Text string.	This is an informational message that explains that the CSM events are now called Validations in TOM. The CSM Rules that used the additional fields JOB type, Word in Message, Operand and Text string are not supported in the TOM Rules.
Rules not generated. Object will be considered active when it is detected that the address space exists. If Rules are desired update the object to include Startup and Shutdown validation.	This is an informational message that explains that the CSM events are now called Validations in TOM.
This object has a Post Start EXEC. The EXEC has been converted, however TOM does not pass internally parameters to the EXEC. You may need to modify the EXEC logic. The available variables to the EXEC are: TOM_OBJECT, TOM_OBJTYPE, TOM_SYSNAME and TOM_REQUEST.	Refer to “Where Variables Can Be Entered” on page 3-63 for information about using variables in EXECs associated with TOM objects.
This object has a Post Stop EXEC. The EXEC has been converted, however TOM does not pass internally parameters to the EXEC. You may need to modify the EXEC logic. The available variables to the EXEC are: TOM_OBJECT, TOM_OBJTYPE, TOM_SYSNAME and TOM_REQUEST.	Refer to “Where Variables Can Be Entered” on page 3-63 for information about using variables in EXECs associated with TOM objects.
Recovery command/EXEC not converted. If this function is desired create a JRNL Rule for TOM message IC1704I. The Rule can execute the Recovery command/EXEC.	There is no equivalent feature in the TOM application for the CSM <b>Recovery command/EXEC</b> field. As an alternative, you can create a JRNL message-initiated Rule in MAINVIEW AutoOPERATOR where the Rule fires for the IC1704I message and schedules an EXEC or issues a command.
TSO SEND not converted. Feature not available in TOM.	There is no equivalent feature in the TOM application for the CSM <b>TSO Send</b> field.
Contact not converted. Feature not available in TOM.	There is no equivalent feature in the TOM application for the CSM <b>Contact</b> field.
Info not converted. Feature not available in TOM.	There is no equivalent feature in the TOM application for the CSM <b>Info</b> field.
Multiple Calendar Schedules detected. TOM Calendar has different behavior than CSM. Please review object Schedules for correctness.	In certain cases, the CSM Schedules must be manually re-created for TOM. For information about how to create Calendar Dependencies for TOM objects, refer to “Examples of Entering Calendar Dependencies” on page 5-26.

Table 10-2 Object Exception Messages

Objects Exception Messages	Additional Explanation
Global Calendar Override converted to 'Exclude' schedule. Reason and Contact name not available in TOM.	Exclude time definitions in TOM represent periods of time when the object status should be STOPPED. Any CSM Global Calendar Override definitions were converted to TOM Exclude time definitions.  The CSM attributes for Reason and Contact for a Global Calendar Override are not converted because there is no equivalent function in the TOM application.
This object has a parent, which is a Grouping object. TOM does not support Grouping objects therefore the Grouping object is replaced with its parent(s).	TOM does not use Grouping objects as they are used in CSM. When the conversion process discovers a Grouping object, it is not converted. However the new TOM object has a dependency on the object that was the parent of the CSM grouping object.
This object contains multiple parents. All object parents have been defined as members of Set 'name'. The Set has been defined as a dependency for this object.	In the TOM application, an object may have a dependency only on one other object or a set of objects. For this object, all the parents of the CSM object have been added to a TOM set and the object has a dependency defined on this set.
This object has a server: '<server_name>'. To achieve the same functionality in TOM update object '<server_name>'. Include the connect command: '<connect_command>' in a Post-start EXEC and the disconnect command: '<disconnect_command>' in a Pre-stop EXEC.	There is no equivalent for CSM servers and connect commands in the TOM application. To achieve a similar effect, code the server connect and disconnect commands into Post-start and Pre-stop EXECs and define these EXECs as part of the new TOM object definition.
The Start/Stop/Cancel command for this object contains stem variables. The format was converted to the TOM stem variables format.	Stem variables used in the Start, Stop, or Cancel commands for the CSM object definitions were converted to TOM stem variables format.
This object is eligible for Move to another CSM Group. Please update the Valid systems for this object and include any MVS systems where you may want to move the object to.	N/A

## Additional Exceptions

If you had objects that were defined in CSM with **Generate Rules = NO** and issued the ACM901I and ACM902I messages to indicate that the objects are UP or DOWN, in TOM you will need to write Rules to capture the messages and schedule EXECs to issue TOM API commands to set the status of these objects to ACTIVE or STOPPED. Refer to “CHSTATUS Function” on page 9-12.

In addition, if you were using the CSMACT EXEC in CSM, you must replace the CSMACT EXEC with a new EXEC that issues the appropriate TOM API commands.



---

# Appendix A TOM Statuses

For information about possible TOM @STATUS values, refer to Table A-1.

**Table A-1 TOM @STATUS Values**

@STATUS Value	What Causes an Object to Have This Status
ACTIVE	Indicates the object is currently running and its status is being managed by TOM. TOM has successfully scheduled any Pre-start EXECs that were defined for the object and the return codes from the EXEC are zero (0). TOM issued the start command (or EXEC) and TOM received a startup validation event for the object.
ACTIVE-BOUNCE	Indicates that TOM is performing a bounce (a shut down and start up) of the object.
ACTIVE-ERR	Indicates that TOM has detected that the object was stopped outside of TOM's control and is in a failure situation that does not have a TOM status.
ACTIVE-NOAUTO	Indicates that TOM is performing a bounce (a shut down and start up) of the object and that no automation is being performed which means that no Pre-start, Post-start, Pre-stop, or Post-stop EXECs are being scheduled as part of the bounce.
ACTIVE-SD	Indicates that the object was brought stopped outside of TOM's control and TOM is in the process of restarting the object.
BLOCKED	Indicates that you have moved the object to another system. This makes the object unavailable to run on this system. An object can also have a BLOCKED status when the BLOCK line command is issued for an object (from the Started Task Overview panel, or from the AOEXEC TOM BLOCK function, or from the BLOCK console command) for an object or a set that was ACTIVE.
BLOCKED-AUTO	Indicates that you have issued a BLOCK command against an object. When you issue the BLOCK command and also specify that the EXECs defined as part of the object's definition are scheduled, the status is BLOCKED-AUTO. This status appears for a very short time every time an active object is blocked. A BLOCK command can be issued from the Started Task Overview panel with a line command, from the AOEXEC TOM BLOCK function, or from the BLOCK console command.

**Table A-1 TOM @STATUS Values**

<b>@STATUS Value</b>	<b>What Causes an Object to Have This Status</b>
BLOCKED-ERR	<p>Indicates that you have issued a BLOCK command for an object from running on a system but it is currently still running. This status appears for a very short time every time an active object is blocked.</p> <p>A BLOCK command can be issued from the Started Task Overview panel with a line command, from the AOEXEC TOM BLOCK function, or from the BLOCK console command.</p>
COMPLETE	<p>Occurs only for Transient objects and indicates that the transient object has completed processing and has ended.</p>
FAILURE-ABEND	<p>Indicates that TOM has received the abnormal termination event which is defined on the Abnormal Termination Validation field of the object's definition and the object has ended abnormally.</p>
FAILURE-AOLINK	<p>Indicates that TOM has lost communication with the MAINVIEW AutoOPERATOR BBI-SS PAS.</p>
FAILURE-CMD-INIT	<p>Indicates that TOM encountered an error while attempting to issue the start command or EXEC for the object.</p>
FAILURE-CMD-TERM	<p>Indicates that TOM encountered an error while attempting to issue the stop command or EXEC for the object.</p>
FAILURE-CMDCOUNT	<p>Indicates that the start command limit has been exceeded by the command counter. The command counter is incremented by one each time a start command is issued or a start retry command is issued while attempting to start the object.</p>
FAILURE-CONFIRM	<p>Indicates that for objects in this state, you have specified YES in either the Verify Start or the Verify Stop field of the object definition and you replied NO to the WTOR that is generated. Object start or stop commands are not issued.</p>
FAILURE-PRESTART	<p>Indicates that a non-zero return code was returned from when TOM attempted to schedule a Pre-start EXEC (or the EXEC did not exist) while attempting to start an object.</p>
FAILURE-PRESTOP	<p>Indicates that a non-zero return code was returned from when TOM attempted to schedule a Pre-stop EXEC (or the EXEC did not exist) while attempting to stop an object.</p>
FAILURE-PROPERTY	<p>Indicates that a property for an object was missing. This should be accompanied by a IC1202W or IC1203W message indicating the missing object definition.</p>
FAILURE-REC-INIT	<p>Indicates that TOM encountered an error while attempting to issue the start command or EXEC for the object.</p>
FAILURE-REC-TERM	<p>Indicates that TOM encountered an error while attempting to issue the stop command or EXEC for the object.</p>
LOCKED	<p>Indicates that you have moved the object to another system. This makes the object unavailable to run on this system. This status also appears briefly when you stop the object outside of its defined schedule.</p> <p>Upon the next IPL the status will be reset.</p> <p>An object can also be LOCKED from the UI with the Lock line command when it is issued against an object or a set that was ACTIVE.</p> <p>In addition, if the object is defined with the field <b>Locked at IPL: YES</b>, that also causes an object status to appear as LOCKED.</p>

**Table A-1 TOM @STATUS Values**

<b>@STATUS Value</b>	<b>What Causes an Object to Have This Status</b>
LOCKED-ERR	<p>Indicates that you have issued a LOCK command for an object from running on a system but it is currently still running. This status appears for a very short time every time an active object is locked.</p> <p>A LOCK command can be issued from the Started Task Overview panel with the line command, from the AOEXEC TOM LOCK function, or from the LOCK console command.</p>
PENDING	<p>Indicates that TOM is in the process of moving an object. TOM cannot start an object on another system before stopping it on the current system. and this period of time is denoted by PENDING. The host shows the status of PENDING for an object as does the target system where the object will be started when shutdown is complete.</p> <p>Note that the object might remain in this state if TOM, for some reason, cannot stop the object on the current system.</p>
SCHEDULED	<p>Indicates that TOM has issued a command to the initialization and termination controller and the controller has not yet begun to process the command. This status will be visible for a only very short period of time.</p>
SCHEDULED-LOCKED	<p>Indicates the object is in the process of being stopped.</p>
SCHEDULED-BLOCKED	<p>Indicates the object is in the process of being stopped and when it has stopped, will have the status of BLOCKED.</p>
STARTING	<p>Indicates TOM is starting the object, and any Pre-Start Execs have been executed, the return codes from the EXECs are zero (they ended successfully) and the start command has been issued. TOM is waiting for the start validation event to arrive. An object can also be in STARTING if TOM is processing the Start Retry commands.</p>
STOPPED	<p>Indicates that TOM is stopping the object, the stop command has been issued and the stop validation event has been received. Any and all Pre-stop EXECs have been scheduled and all the return codes indicate that the EXECs ended successfully. Any and all Post-stop EXECs have been scheduled.</p>
UNKNOWN	<p>The status UNKNOWN can indicate more than one situation:</p> <ul style="list-style-type: none"> <li>• A Reset command for an object was issued from the Reset line command (or from an EXEC or a console command). In this case, this status appears for a very short amount of time because the object should quickly be changed to another status (based on its definitions).</li> <li>• UNKNOWN can also indicate that a system was started without a status or the object is defined for a system that is not active. It can also occur when a Reset command was issued for an object that can only run on a system other than the one you are on and you are not currently connected to it. In these cases, the status UNKNOWN will last until the situation is resolved.</li> </ul>
WAIT-START	<p>This status occurs during the IPL of the first system in a sysplex is IPL'd and it encounters a RESTART-ONLY object. A RESTART-ONLY object is an object that was defined to have its status monitored by TOM but that it would be started outside of TOM's control. When TOM encounters a RESTART-ONLY object, it sets that object's status to WAIT-START on all systems which prevents TOM from starting the object.</p> <p>Once a RESTART-ONLY object has been either been discovered as ACTIVE or is manually started, its state changes to STOPPED on all systems except the system it is active on. This happens only for the first IPL in the system.</p>



---

# Index

## Symbols

&SYSCLONE node 8-12  
&SYSNAME node 8-12  
&SYSPLEX node 8-12, 8-13  
@STATUS  
    ACTIVE A-1  
    ACTIVE-BOUNCE A-1  
    ACTIVE-ERR A-1  
    ACTIVE-NOAUTO A-1  
    ACTIVE-SD A-1  
    BLOCKED A-1  
    BLOCKED-AUTO A-1  
    BLOCKED-ERR A-2  
    COMPLETE A-2  
    FAILURE-ABEND A-2  
    FAILURE-AOLINK A-2  
    FAILURE-CMDCOUNT A-2  
    FAILURE-CMD-INIT A-2  
    FAILURE-CMD-TERM A-2  
    FAILURE-CONFIRM A-2  
    FAILURE-PRESTART A-2  
    FAILURE-PRESTOP A-2  
    FAILURE-PROPERTY A-2  
    FAILURE-REC-INIT A-2  
    FAILURE-REC-TERM A-2  
    LOCKED A-2  
    LOCKED-ERR A-3  
    PENDING A-3  
    SCHEDULED A-3  
    SCHEDULED-BLOCKED A-3  
    SCHEDULED-LOCKED A-3  
    STARTING A-3

STOPPED A-3  
UNKNOWN A-3  
WAIT-START A-3

## A

A (Activate) line command  
    ACTIVATE function 9-6  
A line command  
    Started Task Overview panel 6-5  
    Started Task Sets panel 4-9  
ABEND validation panel 3-47  
ABENDEVT#.ID variable 3-67  
ABENDEVT#.ORIGINSTC variable 3-67  
ABENDEVT#.STEP variable 3-67  
ABENDEVT#.TEXT variable 3-67  
ABENDEVT#.TYPE variable 3-67  
abends when using online Help 2-2  
abnormal termination  
    events 3-46 to 3-48  
    TOM response 1-13  
    validation field 3-48  
    validation panel 3-46  
ACM901I messages 10-14  
ACM902I messages 10-14  
Activate (A) line command  
    ACTIVATE function 9-6  
    Started Task Overview panel 6-5  
    Started Task Sets panel 4-9  
ACTIVATE function  
    command 9-6  
    description 9-6  
    examples 9-6

---

ACTIVATE function (*continued*)  
 parameters 9-6  
 return codes 9-7

ACTIVATE TOM console command 9-46, 9-47

activating  
 definition databases 6-19  
 objects 6-13 to 6-15  
 sets 4-16, 4-17

ACTIVE @STATUS A-1

ACTIVE-BOUNCE @STATUS A-1

ACTIVE-ERR @STATUS A-1

ACTIVE-NOAUTO @STATUS A-1

ACTIVE-SD @STATUS A-1

ADD command  
 Started Task Overview panel 8-15  
 Started Task Sets panel 4-3  
 System Definitions panel 8-10

add object (AO) command, Members of Set panel 8-23

add set (AS) command, Members of Set panel 4-7

Add Sysplex Definition panel 8-8

Add Sysplex Layer panel 8-30

Add System Layer panel 8-26

Add Systems Definition panel 8-11

adding  
 abnormal termination events 3-46 to 3-48  
 calendar entries to a Set 5-20  
 Day entries 5-8 to 5-10  
 Layer IDs 8-9 to 8-11  
 Layer objects to a system layer 8-25 to 8-27  
 Layer sets to a system layer 8-25 to 8-28  
 objects 4-3 to 4-6  
 Period entries 5-12 to 5-14  
 Set entries to Calendar Base 5-18  
 sets 4-3 to 4-8  
 sysplex names 8-6 to 8-9  
 system layers to a sysplex layer 8-29 to 8-31  
 system names 8-9 to 8-11  
 Time entries 5-15 to 5-17

address spaces, TOM  
 starting 2-2  
 stopping 7-17

Administration Options Menu 8-7

affinities, system 3-51

Affinity field 3-54

affinity levels  
 High 3-51  
 Low 3-53  
 Med (Medium) 3-52  
 setting 3-53

AFFINITY variable 3-65

alternate systems, starting objects on 3-49

AO (add object) command, Members of Set panel 8-23

AOAnywhere TOM functions 9-2

AOEXEC TOM  
 functions 9-5  
 keywords 9-5  
 SYSINFO command 9-2

APARs  
 OA03333 2-2  
 OW56486 2-2

AS (add set) command, Members of Set panel 4-7

AT (attach sysplex layer) line command, Sysplex Matrix panel 8-39

attach sysplex layer (AT) line command, Sysplex Matrix panel 8-39

Attach Sysplex Layer panel 8-39

Attach System Layer panel 8-34

attaching  
 definition 8-4  
 sysplex layer to sysplex 8-38 to 8-40  
 system layer to system (remote sysplex) 8-35 to 8-37  
 system layer to system (same sysplex) 8-32 to 8-34

authoritative systems 7-2

AUTOMATION Menu 2-7

## B

backing up registry data sets 7-5

Base Calendar Entries panel 5-9

BBPARM members 2-7

BLOCK function  
 command 9-8  
 description 9-8  
 examples 9-9  
 parameters 9-8  
 return codes 9-9

BLOCK TOM console command 9-46, 9-47

BLOCKED @STATUS A-1

BLOCKED-AUTO @STATUS A-1

---

BLOCKED-ERR @STATUS A-2  
BMC Software, contacting ii  
Bounce (O) line command, BOUNCE function 9-10  
BOUNCE function  
    Bounce (O) line command 9-10  
    command 9-10  
    description 9-10  
    examples 9-11  
    parameters 9-10  
    return codes 9-11  
BOUNCE TOM console command 9-46, 9-47  
browsing set associations 3-13 to 3-15  
BYPASSCMD CNT parameter, START function 9-32

## C

Calendar Base application  
    Activate (A) line command 5-4  
    adding calendar entries to a Set 5-20  
    adding Day entries 5-8 to 5-10  
    adding Period entries 5-12 to 5-14  
    adding Set entries 5-18  
    adding Time entries 5-15 to 5-17  
    benefits 5-3  
    Calendar Dependency 5-5  
    creating 5-6, 5-7  
    Day 5-3  
    default name 5-3  
    description 5-1  
    editing default 5-6  
    Exclude 5-5  
    Include 5-5  
    Period 5-4  
    Set 5-4  
    terms 5-3 to 5-5  
    Time 5-4  
    Validate (V) line command 5-4  
    when to use 5-2  
Calendar Bases panel 5-6  
Calendar Day panel 5-9  
calendar dependencies  
    description 1-6, 5-2  
    object definitions 5-22  
Calendar Dependency panel  
    AND logic 5-32 to 5-34

    example 5-25  
    fields 5-25  
    OR logic 5-29 to 5-31  
    stopping an object 5-35  
    using Exclude 5-27  
    using Exclude for a Period of Time 5-36 to 5-38  
    using Include 5-28  
Calendar Period Entry panel 5-13  
Calendar Set panel 5-19  
Calendar Time panel 5-16  
CH (Change status) line command  
    CHSTATUS function 9-12  
CH line command  
    Started Task Overview panel 6-5  
Change status (CH) line command  
    CHSTATUS function 9-12  
    Started Task Overview panel 6-5  
changing objects 6-16  
CHSTATUS function  
    CH (Change status) line command 9-12  
    command 9-12  
    description 9-12  
    examples 9-12  
    parameters 9-12  
    return codes 9-13  
CHSTATUS TOM console command 9-46, 9-47  
CLIST EXECs 9-4  
clone ID 8-4  
cloning  
    example 7-2  
    failures 7-4  
    overview 7-2  
command failures 1-12  
commands  
    *See also* line commands  
    ADD 4-3, 8-10, 8-15  
    AO (add object) 8-23  
    AOEXEC TOM (SYSINFO) 9-2  
    AS (add set) 4-7  
communication, TOM systems 7-2 to 7-5  
COMPLETE @STATUS A-2  
concepts  
    Calendar Base 5-3 to 5-5  
    Layering 8-3 to 8-4  
Condition field 3-57

---

- Confirm panels
  - Activate 6-15
  - Change Status 6-17
  - Reset 6-11
  - Start 6-6
  - Stop 6-9
  - Suspend 6-14
- console commands
  - MODIFY (F) 8-47
  - TOM 9-45 to 9-51
- Control field 3-11
- CONTROL variable 3-65
- conventions, document and typographical xxiii
- conversion utility
  - accessing 10-5 to 10-10
  - considerations 10-2
- copying TOM Registry 8-47 to 8-48
- Count field 3-60
- creating
  - Calendar Bases 5-6, 5-7
  - EXECs on multiple systems 9-2
  - Layer objects 3-12, 8-14
  - Layer objects for multiple systems 8-43 to 8-45
  - Layer sets 8-21 to 8-24
- CSM (Continuous State Manager) migration 10-2
- CSM v6 to TOM Conversion panels 10-6 to 10-8
- CSMACT EXEC 10-14
- customer support iii

## D

- databases, managing definition 6-16, 6-18
- defining
  - alternate systems 3-49
  - Definition Base name 10-3
  - dependencies between objects or sets 3-55
  - how objects start 3-24 to 3-26
  - how objects stop 3-30
  - initial Control setting for objects 10-3
  - Layer IDs 8-9 to 8-11
  - models 3-8
  - object schedules 3-20 to 3-22
  - objects 3-4
  - periods of time 5-22 to 5-24
  - shutdown validation events 3-43

- start retry commands 3-33
- startup validation events 3-43
- stop retry commands 3-33
- SUSPEND mode 3-10
- sysplex names 8-6 to 8-9
- system names 8-9 to 8-11
- Valid System names 10-3
- Definition Base field 6-18
- Definition Base name, defining 10-3
- definition databases
  - activating 6-19
  - creating multiple 7-23 to 7-25
  - managing 6-16, 6-18
  - switching 6-20
- definition databases panel 6-19
- DELETE parameter, SETPROPERTY function 9-27
- dependencies
  - calendar 1-6
  - scheduling 1-6
- Dependency panel 8-20
- dependency properties 8-20
- Dependency Property value field 3-56
- Description field 3-6
- DESCRIPTION variable 3-65
- documentation
  - conventions xxiii
  - electronic, online Help xxii
  - online xxii
  - related xxi

## E

- editing default Calendar Base 5-6, 5-7
- electronic documentation xxii
- entering
  - Layer IDs 8-9 to 8-11
  - sysplex names 8-6 to 8-9
  - system names 8-9 to 8-11
- examples
  - ACTIVATE function 9-6
  - BLOCK function 9-9
  - BOUNCE function 9-11
  - Calendar Dependency panel 5-25
  - cloning 7-2
  - GET function 9-15 to 9-17
  - Layering 8-2

---

examples (*continued*)

- LIST function 9-18
- LOCK function 9-20
- MOVE function 9-23
- RESET function 9-26
- resolving object names 8-41
- resolving variables 3-63, 3-64
- SETPROPERTY 9-28
- SHUTSET function 9-29
- SHUTSYS function 9-30
- START function 9-32
- STOP function 9-35
- SUSPEND function 9-37
- SYSINFO function 9-39
- TOM log 7-27
- UNBLOCK function 9-42
- UNLOCK function 9-43

Except column 6-4

exception messages 10-11

EXECs

- CLIST 9-4
- creating 9-2
- scheduling for objects 3-36

## F

F (MODIFY) console command 8-47

failed TOM systems, restarting 7-4

FAILURE-ABEND @STATUS A-2

FAILURE-AOLINK @STATUS A-2

FAILURE-CMDCOUNT @STATUS A-2

FAILURE-CMD-INIT @STATUS A-2

FAILURE-CMD-TERM @STATUS A-2

FAILURE-CONFIRM @STATUS A-2

FAILURE-PRESTART @STATUS A-2

FAILURE-PRESTOP @STATUS A-2

FAILURE-PROPERTY @STATUS A-2

FAILURE-REC-INIT @STATUS A-2

FAILURE-REC-TERM @STATUS A-2

fields

- Abnormal termination validation 3-48
- Affinity 3-54
- Calendar Dependency panel 5-25
- Condition 3-57
- Control 3-11
- Count 3-60
- Definition Base 6-18

Dependency Property value 3-56

Description 3-6

Include logic 5-24

Last Modified 3-6

Lock at next IPL 3-26

Max start count 3-28

Modeled from 3-9

Name 3-6, 3-56

On system 3-57

Post-start EXECs 3-40, 5-22

Post-stop EXECs 3-42, 5-22

Pre-start EXECs 3-27, 3-39, 5-22

Pre-stop EXECs 3-41, 5-22

Product 3-6

Property 3-57

Property Value 3-57

Reinstatement 3-60

Require confirmation 3-60

Reset start count after xxxx Minutes 3-27

Reset start count at termination 3-27

Restart only 3-26

Schedules 5-22

Set associations 3-15

Shutdown mode 3-26

Shutdown validation 3-45

Start command 3-27

Start command type 3-27

Start retry commands 3-35, 5-22

Start time out 3-28

Startup validation 3-45

STC name 3-6

Step name 3-6

Stop command 3-31

Stop command type 3-31

Stop retry commands 3-35, 5-22

Stop time out 3-31

This is a Model only 3-9

Try alternate system upon failure 3-50

Type 3-6

Valid Systems 3-18

Verify Start 3-62

Verify Stop 3-62

FULLAUTO parameter

BOUNCE function 9-10

START function 9-31

STOP function 9-34

---

functions

- ACTIVATE 9-6
- BLOCK 9-8
- BOUNCE 9-10
- GET 9-14
- LIST 9-18
- LOCK 9-20
- MOVE 9-22
- RESET 9-25
- SETPROPERTY 9-27
- SHUTSET 9-29
- SHUTSYS 9-30
- START 9-31
- STOP 9-34
- SUSPEND 9-37
- SYSINFO 9-39
- UNBLOCK 9-41
- UNLOCK 9-43

## G

### GET function

- command 9-14
- description 9-14
- examples 9-15 to 9-17
- parameters 9-14
- return codes 9-17

### GROUP keyword 2-5

grouping multiple TOM systems 2-4

## H

### H (Shutdown) line command

- SHUTSET function 9-29
- SHUTSYS function 9-30

### High affinity level 3-51

### HOST parameter

- BLOCK function 9-8
- BOUNCE function 9-10
- CHSTATUS function 9-12
- LOCK function 9-20
- MOVE function 9-23
- SHUTSYS function 9-30
- START function 9-31
- STOP function 9-34
- UNBLOCK function 9-41

- UNLOCK function 9-43
- HOSTS.#.SYSNAME variable 3-66

## I

- I (IPL) Reinstatement value 3-58
- IC1704I message 1-13
- IDCAMS REPRO utility 8-48
- implementing multiple TOM systems 2-4
- Include logic field 5-24
- inheritance 3-7
- IPLSTART variable 3-65
- IRXCMPTM module 2-3
- ISPF, online Help xxii

## K

K (Block) line command, BLOCK function 9-8

keywords

- AOEXEC TOM 9-5
- GROUP 2-5
- SIM 1-7

## L

L (Lock) line command 9-20

Last Modified field 3-6

### Layer ID

- assigning 8-9 to 8-11
- definition 8-4
- sysplex names 8-6

### Layer Management

- menu 8-25
- panel 8-7

### Layer objects

- adding to a system layer 8-25 to 8-27
- creating 3-12, 8-14
- creating for multiple systems 8-43 to 8-45
- definition 8-3
- naming conventions 8-12
- patterns 8-12
- specifying dependency properties 8-20
- specifying valid systems 8-18
- updating 8-46

layer patterns 8-13

Layer set definition panel 8-22

---

Layer sets  
  adding to a system layer 8-25 to 8-28  
  creating 8-21 to 8-24  
  definition 8-3  
  difference from non-Layer sets 8-21  
  naming conventions 8-13  
  specifying names 8-21 to 8-24  
  updating 8-46

Layering  
  example 8-2  
  getting started 8-6  
  overview 8-2  
  task descriptions 8-5  
  terms 8-3 to 8-4

line commands  
  Activate (A) 4-9, 5-4, 6-5  
  attach sysplex layer (AT) 8-39  
  Block (K) 9-8  
  Bounce (O) 9-10  
  Change status (CH) 6-5, 9-12  
  Lock (L) 9-20  
  Move (M) 9-22  
  Reset (T) 4-9, 6-5, 9-25  
  Shutdown (H) 9-29, 9-30  
  Start (S) 4-9, 6-5, 9-31  
  Stop (P) 4-9, 6-5, 9-34  
  Suspend (U) 4-9, 6-5, 9-37  
  Unblock (UK) 9-41  
  Unlock (UL) 9-43  
  Validate (V) 5-4

LIST function  
  command 9-18  
  description 9-18  
  examples 9-18  
  parameters 9-18  
  return codes 9-19

Lock (L) line command, LOCK function 9-20

Lock at next IPL field 3-26

LOCK function  
  command 9-20  
  description 9-20  
  examples 9-20  
  Lock (L) line command 9-20  
  parameter 9-20  
  return codes 9-21

LOCK TOM console command 9-46, 9-47

LOCKED @STATUS A-2

LOCKED-ERR @STATUS A-3

log data sets, sharing 2-4

LOGGER TOM console command 9-51

Low affinity level 3-53

## M

M (Manual) Reinstatement value 3-58

M (Move) line command 9-22

MAINVIEW AutoOPERATOR, starting first 2-3

MAINVIEW Total Object Manager. *See* TOM

managing  
  definition databases 6-16, 6-18  
  objects on multiple systems 7-3  
  sets 4-9

Max start count field 3-28

MDLTYPE variable 3-66

Med (Medium) affinity level 3-52

Members of Set panel  
  add object (AO) command 8-23  
  add set (AS) command 4-7  
  example 4-6

menus  
  Administration Options 8-7  
  AUTOMATION 2-7  
  Layer Management 8-25  
  TOM Primary Options 7-23

Message panel 3-44

messages  
  ACM901I 10-14  
  ACM902I 10-14  
  exception 10-11  
  IC1704I 1-13  
  TOM log 7-27

migration considerations, CSM (Continuous State Manager) 10-2

migration utility. *See* conversion utility

Modeled from field 3-9

models  
  defining 3-8  
  description 1-5, 3-7  
  object inheritance 3-7

MODIFY (F) console command 8-47

Move (M) line command, MOVE function 9-22

MOVE function  
  command 9-22  
  description 9-22  
  examples 9-23

---

MOVE function (*continued*)

- Move (M) line command 9-22
- parameters 9-22
- return codes 9-23

- MOVE TOM console command 9-46, 9-47
- multiple TOM systems, implementing 2-4

## N

- Name field 3-6, 3-56

- NAME parameter, SETPROPERTY function 9-27

- name patterns, selecting for objects 10-3

- naming conventions

  - Layer objects 8-12

  - Layer sets 8-13

- naming restrictions

  - objects 9-3

  - sets 9-3

- nesting sets 4-1

- New Calendar Base panel 5-7

- New Set name panel 4-4

- nodes

  - &SYSCLONE 8-12

  - &SYSNAME 8-12

  - &SYSPLEX 8-12, 8-13

  - user-specified object name 8-12, 8-13

- normal objects 1-4, 3-2

## O

- O (Bounce) line command 9-10

- OA03333, APAR 2-2

- object driver component 1-10

- OBJECT parameter

  - ACTIVATE function 9-6

  - BLOCK function 9-8

  - BOUNCE function 9-10

  - CHSTATUS function 9-12

  - GET function 9-14

  - LOCK function 9-20

  - MOVE function 9-22

  - RESET function 9-25

  - SETPROPERTY function 9-27

  - START function 9-31

  - STOP function 9-34

  - SUSPEND function 9-37

  - UNBLOCK function 9-41

  - UNLOCK function 9-43

- OBJECT TOM console command 9-51

- objects

  - activating 6-13 to 6-15

  - adding 4-3 to 4-6

  - Calendar Dependency definitions 5-22

  - changing 6-16

  - defining 3-4

  - defining dependencies between 3-55

  - defining in SUSPEND mode 3-10

  - defining schedules 3-20 to 3-22

  - dependencies 1-6

  - exception messages 10-11

  - information, viewing 6-2

  - inheritance 3-7

  - inheriting all properties 3-8

  - inheriting some properties 3-9

  - initial Control setting 10-3

  - managing on multiple systems 7-3

  - models 1-5

  - names 8-13

  - naming restrictions 9-3

  - normal 1-4, 3-2

  - recovering failed 1-12

  - reference variables 3-65

  - reinstating 3-58

  - removing from SUSPEND mode 3-58

  - resetting 6-11

  - restarting 1-14

  - schedule an EXEC for 3-36

  - selecting name patterns 10-3

  - shutdown events 3-43

  - start verification message 3-61

  - starting with Start (S) line command 6-6 to 6-8

  - startup events 3-43

  - statuses 1-6

  - stop verification message 3-61

  - stopping manually 7-17, 7-22

  - stopping with Stop (P) line command 6-9 to 6-10

  - stopping within a set 7-19

  - storing 3-3

  - suspending 6-13

  - transient 1-5, 3-2

  - types 1-4, 3-2

---

objects (*continued*)

- using definitions to start 3-24 to 3-26
- using definitions to stop 3-30
- variables 3-65
- verifying stop 7-16, 7-21
- viewing information 6-2

Objects panel 4-5

Offload utility 8-47

On system field 3-57

online documentation xxii

online Help xxii

OW56486, APAR 2-2

## P

P (Stop) line command

- Started Task Overview panel 6-5

P line command

- Started Task Sets panel 4-9
- STOP function 9-34

panels

- ABEND validation 3-47
- Abnormal Termination Validation 3-46
- Add Sysplex Definition 8-8
- Add Sysplex Layer 8-30
- Add System Definition 8-11
- Add System Layer 8-26
- Attach Sysplex Layer 8-39
- Attach System Layer 8-34
- Base Calendar Entries 5-9
- Calendar Bases 5-6
- Calendar Day 5-9
- Calendar Dependency 5-25
- Calendar Period Entry 5-13
- Calendar Set 5-19
- Calendar Time 5-16
- Confirm Activate 6-15
- Confirm Change Status 6-17
- Confirm Reset 6-11
- Confirm Start 6-6
- Confirm Stop 6-9
- Confirm Suspend 6-14
- CSM v6 to TOM Conversion 10-6 to 10-8
- Definition databases 6-19
- Dependency 8-20
- Layer Management 8-7
- Layer set definition 8-22

Members of Set 4-6

Message 3-44

New Calendar Base 5-7

New Set Name 4-4

Objects 4-5

Pre-start EXEC 3-37

SCHEDULE Calendar Dependencies 5-23

Sets 4-7

Start retry commands 3-33

Started Task Overview 6-3

Started tasks sets 3-14

Startup validation 3-43

Sysplex Layer 8-29

Sysplex Matrix 8-8, 8-32

System Definitions 8-10

System Layer Objects 8-27

System Layer Sets 8-28

System Layers 8-26

TOM Conversion Log 10-10

Valid System 8-18

parameters

- BYPASSCMD CNT, START function 9-32
- DELETE, SETPROPERTY function 9-27
- FULLAUTO, BOUNCE function 9-10
- FULLAUTO, START function 9-31
- FULLAUTO, STOP function 9-34
- HOST, BLOCK function 9-8
- HOST, BOUNCE function 9-10
- HOST, CHSTATUS function 9-12
- HOST, LOCK function 9-20
- HOST, MOVE function 9-23
- HOST, SHUTSYS function 9-30
- HOST, START function 9-31
- HOST, STOP function 9-34
- HOST, UNBLOCK function 9-41
- HOST, UNLOCK function 9-43
- NAME, SETPROPERTY function 9-27
- OBJECT, ACTIVATE function 9-6
- OBJECT, BLOCK function 9-8
- OBJECT, BOUNCE function 9-10
- OBJECT, CHSTATUS function 9-12
- OBJECT, GET function 9-14
- OBJECT, LOCK function 9-20
- OBJECT, MOVE function 9-22
- OBJECT, RESET function 9-25
- OBJECT, SETPROPERTY function 9-27
- OBJECT, START function 9-31
- OBJECT, STOP function 9-34

---

parameters (*continued*)

OBJECT, SUSPEND function 9-37  
OBJECT, UNBLOCK function 9-41  
OBJECT, UNLOCK function 9-43  
RESETOPT, RESET function 9-25  
RESNEXTIPL, START function 9-31  
RESNEXTIPL, STOP function 9-34  
SET, ACTIVATE function 9-6  
SET, BLOCK function 9-8  
SET, BOUNCE function 9-10  
SET, GET function 9-14  
SET, LOCK function 9-20  
SET, MOVE function 9-22  
SET, RESET function 9-25  
SET, SETPROPERTY function 9-27  
SET, SHUTSET function 9-29  
SET, START function 9-31  
SET, STOP function 9-34  
SET, SUSPEND function 9-37  
SET, UNBLOCK function 9-41  
SET, UNLOCK function 9-43  
STEM, GET function 9-14  
STEM, LIST function 9-18  
TOHOST, MOVE function 9-23  
TOMID, ACTIVATE function 9-6  
TOMID, BLOCK function 9-8  
TOMID, BOUNCE function 9-11  
TOMID, CHSTATUS function 9-12  
TOMID, GET function 9-14  
TOMID, LIST function 9-18  
TOMID, LOCK function 9-20  
TOMID, MOVE function 9-23  
TOMID, overview 9-2  
TOMID, RESET function 9-25  
TOMID, SETPROPERTY function 9-28  
TOMID, SHUTSET function 9-29  
TOMID, SHUTSYS function 9-30  
TOMID, START function 9-32  
TOMID, STOP function 9-35  
TOMID, SUSPEND function 9-37  
TOMID, UNBLOCK function 9-41  
TOMID, UNLOCK function 9-43  
TYPE, LIST function 9-18  
VARIABLE, SETPROPERTY function 9-27  
XCFGROUP, ACTIVATE function 9-6  
XCFGROUP, BLOCK function 9-8  
XCFGROUP, BOUNCE function 9-11  
XCFGROUP, CHSTATUS function 9-12

XCFGROUP, GET function 9-14  
XCFGROUP, LIST function 9-18  
XCFGROUP, LOCK function 9-20  
XCFGROUP, MOVE function 9-23  
XCFGROUP, overview 9-2  
XCFGROUP, RESET function 9-25  
XCFGROUP, SETPROPERTY function 9-28  
XCFGROUP, SHUTSET function 9-29  
XCFGROUP, SHUTSYS function 9-30  
XCFGROUP, START function 9-32  
XCFGROUP, STOP function 9-35  
XCFGROUP, SUSPEND function 9-37  
XCFGROUP, SYSINFO function 9-39  
XCFGROUP, UNBLOCK function 9-41  
XCFGROUP, UNLOCK function 9-43

patterns

layer 8-13  
Layer objects 8-12  
naming objects 10-3  
specifying names 8-14 to 8-17

PENDING @STATUS A-3

Post-start EXECs field 3-40, 5-22

Post-stop EXECs field 3-42, 5-22

Pre-start EXECs field 3-27, 3-39, 5-22

Pre-start EXECs panels 3-37

Pre-start EXECs, specifying 3-36 to 3-38

Pre-stop EXECs field 3-41, 5-22

prioritizing eligible systems 3-51

Product field 3-6

product support iii

PRODUCT variable 3-66

propagate, definition 8-4

Property field 3-57

Property Value field 3-57

PTFs

UA00713 2-2

UW94273 2-2

## Q

QUICK shutdown mode 7-14

## R

recovering failed objects 1-12

- reference variables 3-65
- registry data sets
  - backing up 7-5
  - sharing 2-4
- REGISTRY TOM console command 9-50
- Reinstatement field 3-60
- reinstating objects 3-58
- related documentation xxi
- release notes xxii
- removing SUSPEND objects 3-58
- Require confirmation field 3-60
- requirements, TOM 2-2
- Reset (T) line command
  - RESET function 9-25
  - Started Task Overview panel 6-5
  - Started Task Sets panel 4-9
- RESET function
  - command 9-25
  - description 9-25
  - examples 9-26
  - parameter 9-25
  - Reset line command 9-25
  - return codes 9-26
- Reset start count after xxxx Minutes field 3-27
- Reset start count at termination field 3-27
- RESET TOM console command 9-46, 9-48
- RESETOPT parameter, RESET function 9-25
- resetting
  - objects 6-11
  - sets 4-14
- RESNEXTIPL parameter
  - START function 9-31
  - STOP function 9-34
- resolving object names after attach 8-41
- Restart only field 3-26
- RESTART variable 3-65
- restarting
  - failed TOM systems 7-4
  - objects 1-14
- RESTIME variable 3-65
- restrictions
  - object naming 9-3
  - set naming 9-3
- return codes
  - ACTIVATE function 9-7
  - AOEXEC function 9-4
  - BLOCK function 9-9
  - CHSTATUS function 9-13

- GET function 9-17
- LIST function 9-19
- LOCK function 9-21
- MOVE function 9-23
- RESET function 9-26
- SETPROPERTY function 9-28
- SHUTSET function 9-29
- SHUTSYS function 9-30
- START function 9-32
- STOP function 9-35
- SUSPEND function 9-38
- SYSINFO function 9-40
- UNBLOCK function 9-42
- UNLOCK function 9-44
- reviewing TOM log 7-27
- REXX procedures, running 2-3
- REXX/370 Alternate Library 2-3
- running compiled REXX procedures 2-3

## S

- S (Start) line command
  - Started Task Overview panel 6-5
  - Started Task Sets panel 4-9
- SCHEDULE Calendar Dependencies panel 5-23
- SCHEDULED @STATUS A-3
- SCHEDULED-BLOCKED @STATUS A-3
- SCHEDULED-LOCKED @STATUS A-3
- Schedules field 5-22
- schedules, defining for objects 3-20 to 3-22
- scheduling EXECs for objects 3-36
- scrolling problem, APAR correction 2-2
- Set associations field 3-15
- SET parameter
  - ACTIVATE function 9-6
  - BLOCK function 9-8
  - BOUNCE function 9-10
  - GET function 9-14
  - LOCK function 9-20
  - MOVE function 9-22
  - RESET function 9-25
  - SETPROPERTY function 9-27
  - SHUTSET function 9-29
  - START function 9-31
  - STOP function 9-34
  - SUSPEND function 9-37
  - UNBLOCK function 9-41

---

- SET parameter (*continued*)
  - UNLOCK function 9-43
- SETPROPERTY function
  - command 9-27
  - description 9-27
  - examples 9-28
  - parameter 9-27
  - return codes 9-28
- sets
  - activating 4-16, 4-17
  - adding 4-3 to 4-8
  - defining dependencies between 3-55
  - definition 1-5
  - difference from Layer sets 8-21
  - managing 4-9
  - names 8-14
  - naming restrictions 9-3
  - nesting 4-1
  - resetting 4-14
  - starting 4-10
  - stopping 4-12
  - suspending 4-16
  - understanding 4-1
  - viewing associations 3-13 to 3-15
  - when to use 4-2
- Sets panel 4-7
- setting affinity levels 3-53
- sharing data sets 2-4
- Shutdown mode field 3-26
- SHUTDOWN TOM console command 9-51
- Shutdown validation events 3-43
- Shutdown validation field 3-45
- SHUTDOWN variable 3-65
- SHUTSET function 9-29
- SHUTSET TOM console command 9-46, 9-48
- SHUTSYS function 9-30
- SHUTSYS TOM console command 9-46, 9-48
- shutting down sets
  - overview 7-18
  - procedure 7-19
- shutting down TOM systems
  - overview 7-12
  - QUICK shutdown mode 7-14
  - stopping a specific system 7-13
- SIM keyword 1-7
- single image mode
  - considerations 1-8
  - description 1-7
- special characters xxiv
- specifying
  - affinity levels 3-53
  - BBPARM members 2-7
  - dependency properties 8-20
  - Layer IDs 8-9 to 8-11
  - Layer set names 8-21 to 8-24
  - pattern names 8-14 to 8-17
  - Pre-start EXECs 3-36 to 3-38
  - start command limits 1-14
  - sysplex names 8-6 to 8-9
  - system names 8-9 to 8-11
  - valid systems 8-18
- Start (S) line command
  - START function 9-31
  - Started Task Overview panel 6-5
  - Started Task Sets panel 4-9
- Start command field 3-27
- start command limits, specifying 1-14
- Start command type field 3-27
- START function
  - command 9-31
  - description 9-31
  - examples 9-32
  - parameters 9-31
  - return codes 9-32
  - Start line command 9-31
- Start line command, START function 9-31
- start retry commands 3-33
- Start retry commands field 3-35, 5-22
- Start retry commands panels 3-33
- Start time out field 3-28
- START TOM console command 9-46, 9-48
- START.#.CMD variable 3-66
- START.#.TIMEOUT variable 3-66
- START.#.TYPE variable 3-66
- Started Task Definition panel fields
  - Abnormal termination validation 3-48
  - Affinity 3-54
  - Condition 3-57
  - Control 3-11
  - Count 3-60
  - Dependency Property value 3-56
  - Description 3-6
  - Last Modified 3-6
  - Lock at next IPL 3-26
  - Max start count 3-28
  - Modeled from 3-9

---

Started Task Definition panel fields (*continued*)

- Name 3-6, 3-56
- On system 3-57
- Product 3-6
- Property 3-57
- Property Value 3-57
- Reinstatement 3-60
- Require confirmation 3-60
- Reset start count after xxxx Minutes 3-27
- Reset start count at termination 3-27
- Restart only 3-26
- Schedules 3-22
- Set associations 3-15
- Shutdown mode 3-26
- Shutdown validation 3-45
- Start command 3-27
- Start command type 3-27
- Start retry commands 3-35
- Start time out 3-28
- Startup 3-26
- Startup validation 3-45
- STC name 3-6
- Step name 3-6
- Stop command 3-31
- Stop command type 3-31
- Stop retry commands 3-35
- Stop time out 3-31
- This a Model only 3-9
- Try alternate system upon failure 3-50
- Type 3-6
- Valid Systems 3-18
- Verify Start 3-62
- Verify Stop 3-62

Started Task Definition panel, using variables 3-63 to 3-67

Started Task Overview panel

- displaying 6-3
- example 6-3
- line commands 6-5
- understanding 6-2

Started tasks sets panel 3-14

STARTEVT.#.ID variable 3-66

STARTEVT.#.ORIGINSTC variable 3-67

STARTEVT.#.STEP variable 3-67

STARTEVT.#.TEXT variable 3-67

STARTEVT.#.TYPE variable 3-66

starting

- MAINVIEW AutoOPERATOR 2-3
- objects on alternate systems 3-49
- sets 4-10
- TOM address spaces 2-2
- TOM initially 2-6
- updated TOM systems 7-2

STARTING @STATUS A-3

starting objects

- object driver component 1-10
- using definitions to start 3-24 to 3-26
- using the Start (S) line command 6-6 to 6-8

Startup validation events 3-43

Startup validation field 3-45

Startup validation panels 3-43

statuses

- object 1-6
- TOM A-1 to A-3

STC name field 3-6

STCNAME variable 3-66

STEM parameter

- GET function 9-14
- LIST function 9-18

stem variables 9-4

Step name field 3-6

STEPNAME variable 3-66

Stop (P) line command

- Started Task Overview panel 6-5
- Started Task Sets panel 4-9
- STOP function 9-34

Stop command field 3-31

Stop command type field 3-31

STOP function

- command 9-34
- description 9-34
- examples 9-35
- parameters 9-34
- return code 9-35
- Stop (P) line command 9-34

stop retry commands 3-33

Stop retry commands field 3-35, 5-22

Stop time out field 3-31

STOP TOM console command 9-46, 9-49

STOPEVT.#.ID variable 3-66

STOPEVT.#.ORIGINSTC variable 3-66

STOPEVT.#.STEP variable 3-66

STOPEVT.#.TEXT variable 3-66

STOPEVT.#.TYPE variable 3-66

STOPPED @STATUS A-3

- stopping
  - sets 4-12
  - TOM address spaces 7-17
- stopping objects
  - errors 7-15
  - errors in sets 7-21
  - manually 7-17, 7-22
  - object driver component 1-10
  - on specific TOM systems 7-13
  - QUICK shutdown mode 7-14
  - using definitions to stop 3-30
  - using the Stop (P) line command 6-9 to 6-10
  - verifying 7-16, 7-21
  - within a set 7-19
- storing objects 3-3
- support, customer iii
- Suspend (U) line command
  - Started Task Overview panel 6-5
  - Started Task Sets panel 4-9
  - SUSPEND function 9-37
- SUSPEND function
  - command 9-37
  - description 9-37
  - examples 9-37
  - parameters 9-37
  - return codes 9-38
  - Suspend line command 9-37
- SUSPEND mode 3-10
- SUSPEND TOM console command 9-46, 9-49
- suspending
  - objects 6-13
  - sets 4-16
- switching definition databases 6-20
- synchronization, TOM systems 7-2
- SYSINFO function
  - command 9-39
  - description 9-39
  - examples 9-39
  - parameters 9-39
  - return codes 9-40
- SYSINFO, AOEXEC TOM command 9-2
- sysplex layers
  - adding system layers 8-29 to 8-31
  - attaching to a sysplex 8-38 to 8-40
  - definition 8-4
- Sysplex Layers panel 8-29
- Sysplex Matrix panel 8-8, 8-32
- sysplex names, entering 8-6 to 8-9

- system affinities
  - description 3-51
  - High level 3-51
  - Low level 3-53
  - Med (Medium) level 3-52
- System Definitions panel 8-10
- System Layer Objects panel 8-27
- System Layer Sets panel 8-28
- system layers
  - adding Layer objects 8-25 to 8-27
  - adding Layer sets 8-25 to 8-28
  - adding to a sysplex layer 8-29 to 8-31
  - attaching to a system (remote sysplex) 8-35 to 8-37
  - attaching to a system (same sysplex) 8-32 to 8-34
  - definition 8-4
- System Layers panel 8-26

## T

- T (Reset) line command
  - Started Task Overview panel 6-5
  - Started Task Sets panel 4-9
- technical support iii
- termination, abnormal 1-13, 3-46 to 3-48
- test mode 10-2
- This is a Model only field 3-9
- TOHOST parameter, MOVE function 9-23
- TOM
  - address spaces, starting 2-2
  - address spaces, stopping 7-17
  - benefits 1-3
  - command failures 1-12
  - Conversion Log panel 10-10
  - log 7-27
  - object driver component 1-10
  - object types 1-4
  - Offload utility 8-47
  - overview 1-1
  - Primary Options Menu 7-23
  - recovering failed objects 1-12
  - Registry, copying 8-47 to 8-48
  - requirements 2-2
  - response to abnormal termination 1-13
  - single image mode 1-7
  - starting initially 2-6

- 
- TOM (*continued*)
    - statuses A-1 to A-3
    - systems, synchronization 7-2
    - TOM console command 9-51
    - TOM-plex 1-7
    - Upload utility 8-47
  - TOM console commands
    - ACTIVATE 9-46, 9-47
    - BLOCK 9-46, 9-47
    - BOUNCE 9-46, 9-47
    - CHSTATUS 9-46, 9-47
    - general format 9-45
    - LOCK 9-46, 9-47
    - LOGGER 9-51
    - MOVE 9-46, 9-47
    - OBJECT 9-51
    - REGISTRY 9-50
    - RESET 9-46, 9-48
    - SHUTDOWN 9-51
    - SHUTSET 9-46, 9-48
    - SHUTSYS 9-46, 9-48
    - START 9-46, 9-48
    - STOP 9-46, 9-49
    - SUSPEND 9-46, 9-49
    - TOM 9-51
    - UNBLOCK 9-46, 9-49
    - UNLOCK 9-47, 9-49
    - XCF 9-50
  - TOMID parameter
    - ACTIVATE function 9-6
    - BLOCK function 9-8
    - BOUNCE function 9-11
    - CHSTATUS function 9-12
    - GET function 9-14
    - LIST function 9-18
    - LOCK function 9-20
    - MOVE function 9-23
    - omitting 9-2
    - RESET function 9-25
    - SETPROPERTY function 9-28
    - SHUTSET function 9-29
    - SHUTSYS function 9-30
    - START function 9-32
    - STOP function 9-35
    - SUSPEND function 9-37
    - UNBLOCK function 9-41
    - UNLOCK function 9-43
  - TOM-plex, definition 1-7
  - transient objects 1-5, 3-2
  - troubleshooting cloning failures 7-4
  - Try alternate system upon failure field 3-50
  - Type field 3-6
  - TYPE parameter, LIST function 9-18
  - TYPE variable 3-65
  - typographical conventions xxiii, xxiv
- ## U
- U (Suspend) line command
    - Started Task Overview panel 6-5
  - U line command
    - Started Task Sets panel 4-9
    - SUSPEND function 9-37
  - UA00713, PTF 2-2
  - UK (Unblock) line command 9-41
  - UL (Unlock) line command 9-43
  - Unblock (UK) line command, UNBLOCK function 9-41
  - UNBLOCK function
    - command 9-41
    - description 9-41
    - examples 9-42
    - parameters 9-41
    - return codes 9-42
    - Unblock (UK) line command 9-41
  - UNBLOCK TOM console command 9-46, 9-49
  - UNKNOWN @STATUS A-3
  - Unlock (UL) line command, UNLOCK function 9-43
  - UNLOCK function
    - command 9-43
    - description 9-43
    - examples 9-43
    - parameters 9-43
    - return codes 9-44
    - Unlock (UL) line command 9-43
  - UNLOCK TOM console command 9-47, 9-49
  - updating
    - Layer objects 8-46
    - Layer sets 8-46
  - Upload utility 8-47
  - user-specified nodes 8-12, 8-13

---

## utilities

- conversion 10-2
- IDCAM REPRO 8-48
- TOM Offload 8-47
- TOM Upload 8-47

UW94273, PTF 2-2

## V

V (Validate) line command 5-4

Valid Systems

- field 3-18
- list 3-16
- names 10-3
- panel 8-18

valid systems, defining 3-16

VARIABLE parameter, SETPROPERTY  
function 9-27

variable reference names 3-65

variables

- ABENDEVT.#.ID 3-67
- ABENDEVT.#.ORIGINSTC 3-67
- ABENDEVT.#.STEP 3-67
- ABENDEVT.#.TEXT 3-67
- ABENDEVT.#.TYPE 3-67
- AFFINITY 3-65
- CONTROL 3-65
- DESCRIPTION 3-65
- examples of resolving 3-63
- HOSTS.#.SYSNAME 3-66
- IPLSTART 3-65
- MLDTYPE 3-66
- PRODUCT 3-66
- reference names 3-65
- resolving from different sources 3-64
- RESTART 3-65
- RESTIME 3-65
- SHUTDOWN 3-65
- START.#.CMD 3-66
- START.#.TIMEOUT 3-66
- START.#.TYPE 3-66
- STARTEVT.#.ID 3-66
- STARTEVT.#.ORIGINSTC 3-67
- STARTEVT.#.STEP 3-67
- STARTEVT.#.TEXT 3-67
- STARTEVT.#.TYPE 3-66
- STCNAME 3-66

- STEPNAME 3-66
- STOPEVT.#.ID 3-66
- STOPEVT.#.ORIGINSTC 3-66
- STOPEVT.#.STEP 3-66
- STOPEVT.#.TEXT 3-66
- STOPEVT.#.TYPE 3-66
- TYPE 3-65

where to use 3-63

verification messages, object start/stop 3-61

Verify Start field 3-62

Verify Stop field 3-62

verifying object start message 3-61

verifying object stop

message 3-61

procedure 7-16, 7-21

viewing

object information 6-2

set associations 3-13 to 3-15

## W

WAIT-START @STATUS A-3

WTOR (write-to-operator-with-response)  
message 3-61, 7-16

## X

XCF TOM console command 9-50

XCFGROUP parameter

- ACTIVATE function 9-6
- BLOCK function 9-8
- BOUNCE function 9-11
- CHSTATUS function 9-12
- GET function 9-14
- LIST function 9-18
- LOCK function 9-20
- MOVE function 9-23
- omitting 9-2
- RESET function 9-25
- SETPROPERTY function 9-28
- SHUTSET function 9-29
- SHUTSYS function 9-30
- START function 9-32
- STOP function 9-35
- SUSPEND function 9-37
- SYSINFO function 9-39

---

XCFGROUP parameter (*continued*)

UNBLOCK function 9-41

UNLOCK function 9-43



# END USER LICENSE AGREEMENT NOTICE

**BY OPENING THE PACKAGE, INSTALLING, PRESSING "AGREE" OR "YES" OR USING THE PRODUCT, THE ENTITY OR INDIVIDUAL ENTERING INTO THIS AGREEMENT AGREES TO BE BOUND BY THE FOLLOWING TERMS. IF YOU DO NOT AGREE WITH ANY OF THESE TERMS, DO NOT INSTALL OR USE THE PRODUCT, PROMPTLY RETURN THE PRODUCT TO BMC OR YOUR BMC RESELLER, AND IF YOU ACQUIRED THE LICENSE WITHIN 30 DAYS OF THE DATE OF YOUR ORDER CONTACT BMC OR YOUR BMC RESELLER FOR A REFUND OF LICENSE FEES PAID. IF YOU REJECT THIS AGREEMENT, YOU WILL NOT ACQUIRE ANY LICENSE TO USE THE PRODUCT.**

This Agreement ("**Agreement**") is between the entity or individual entering into this Agreement ("You") and BMC Software Distribution, Inc., a Delaware corporation located at 2101 CityWest Blvd., Houston, Texas, 77042, USA or its affiliated local licensing entity ("BMC"). "You" includes you and your Affiliates. "Affiliate" is defined as an entity which controls, is controlled by or shares common control with a party. IF MORE THAN ONE LICENSE AGREEMENT COULD APPLY TO THE PRODUCT, THE FOLLOWING ORDER OF LICENSE AGREEMENT PRECEDENCE APPLIES: (1) WEB BASED LICENSE AGREEMENT WITH BMC, (2) WRITTEN LICENSE AGREEMENT WITH BMC, (3) SHRINK-WRAP LICENSE AGREEMENT WITH BMC PROVIDED WITH THE PRODUCT, AND (4) THIS ELECTRONIC LICENSE AGREEMENT WITH BMC. In addition to the restrictions imposed under this Agreement, any other usage restrictions contained in the Product installation instructions or release notes shall apply to Your use of the Product.

**PRODUCT AND CAPACITY.** "**Software**" means the object code version of the computer programs provided, via delivery or electronic transmission, to You. Software includes computer files, enhancements, maintenance modifications, upgrades, updates, bug fixes, and error corrections.

"**Documentation**" means all written or graphical material provided by BMC in any medium, including any technical specifications, relating to the functionality or operation of the Software.

"**Product**" means the Software and Documentation.

"**License Capacity**" means the licensed capacity for the Software with the pricing and other license defining terms, including capacity restrictions, such as tier limit, total allowed users, gigabyte limit, quantity of Software, and/or other capacity limitations regarding the Software. For licenses based on the power of a computer, You agree to use BMC's current computer classification scheme, which is available at <http://www.bmc.com> or can be provided to You upon request.

**ACCEPTANCE.** The Product is deemed accepted by You, on the date that You received the Product from BMC.

**LICENSE.** Subject to the terms of this Agreement, as well as Your payment of applicable fees, BMC grants You a non-exclusive, non-transferable, perpetual (unless a term license is provided on an order) license for each copy of the Software, up to the License Capacity, to do the following:

- (a) install the Software on Your owned or leased hardware located at a facility owned or controlled by You in the country where You acquired the license;
- (b) operate the Software solely for processing Your own data in Your business operations; and
- (c) make one copy of the Software for backup and archival purposes only (collectively a "**License**").

If the Software is designed by BMC to permit you to modify such Software, then you agree to only use such modifications or new software programs for Your internal purposes or otherwise consistent with the License. BMC grants You a license to use the Documentation solely for Your internal use in Your operations.

**LICENSE UPGRADES.** You may expand the scope of the License Capacity only pursuant to a separate agreement with BMC for such expanded usage and Your payment of applicable fees. There is no additional warranty period or free support period for license upgrades.

**RESTRICTIONS:** You agree to **NOT**:

- (a) disassemble, reverse engineer, decompile or otherwise attempt to derive any Software from executable code;
- (b) distribute or provide the Software to any third party (including without limitation, use in a service bureau, outsourcing environment, or processing the data of third parties, or for rental, lease, or sublicense); or
- (c) provide a third party with the results of any functional evaluation or benchmarking or performance tests, without BMC's prior written approval, unless prohibited by local law.

**TRIAL LICENSE.** If, as part of the ordering process, the Product is provided on a trial basis, then these terms apply: (i) this license consists solely of a non-exclusive, non-transferable evaluation license to operate the Software for the period of time specified from BMC or, if not specified, a 30 day time period ("**Trial Period**") only for evaluating whether You desire to acquire a capacity-based license to the Product for a fee; and (ii) Your use of the Product is on an AS IS basis without any warranty, and **BMC, ITS AFFILIATES AND RESELLERS, AND LICENSORS DISCLAIM ANY AND ALL WARRANTIES (INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT) AND HAVE NO LIABILITY WHATSOEVER RESULTING FROM THE USE OF THIS PRODUCT UNDER THIS TRIAL LICENSE ("Trial License").** BMC may terminate for its convenience a Trial License upon notice to You. When the Trial Period ends, Your right to use this Product automatically expires. If You want to continue Your use of the Product beyond the Trial Period, contact BMC to acquire a capacity-based license to the Product for a fee.

**TERMINATION.** This Agreement shall immediately terminate if You breach any of its terms. Upon termination, for any reason, You must uninstall the Software, and either certify the destruction of the Product or return it to BMC.

**OWNERSHIP OF THE PRODUCT.** BMC or its Affiliates or licensors retain all right, title and interest to and in the BMC Product and all intellectual property, informational, industrial property and proprietary rights therein. BMC neither grants nor otherwise transfers any rights of ownership in the BMC Product to You. Products are protected by applicable copyright, trade secret, and industrial and intellectual property laws. BMC reserves any rights not expressly granted to You herein.

**CONFIDENTIAL AND PROPRIETARY INFORMATION.** The Products are and contain valuable confidential information of BMC (“**Confidential Information**”). Confidential Information means non-public technical and non-technical information relating to the Products and Support, including, without limitation, trade secret and proprietary information, and the structure and organization of the Software. You may not disclose the Confidential Information to third parties. You agree to use all reasonable efforts to prevent the unauthorized use, copying, publication or dissemination of the Product.

**WARRANTY.** Except for a Trial License, BMC warrants that the Software will perform in substantial accordance with the Documentation for a period of one year from the date of the order. This warranty shall not apply to any problems caused by software or hardware not supplied by BMC or to any misuse of the Software.

**EXCLUSIVE REMEDY.** BMC’s entire liability, and Your exclusive remedy, for any defect in the Software during the warranty period or breach of the warranty above shall be limited to the following: BMC shall use reasonable efforts to remedy defects covered by the warranty or replace the defective Software within a reasonable period of time, or if BMC cannot remedy or replace such defective copy of the Software, then BMC shall refund the amount paid by You for the License for that Software. BMC’s obligations in this section are conditioned upon Your providing BMC prompt access to the affected Software and full cooperation in resolving the claim.

**DISCLAIMER. EXCEPT FOR THE EXPRESS WARRANTIES ABOVE, THE PRODUCT IS PROVIDED “AS IS.” BMC, ITS AFFILIATES AND LICENSORS SPECIFICALLY DISCLAIM ALL OTHER WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. BMC DOES NOT WARRANT THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE, OR THAT ALL DEFECTS CAN BE CORRECTED.**

**DISCLAIMER OF DAMAGES. IN NO EVENT IS BMC, ITS AFFILIATES OR LICENSORS LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES RELATING TO OR ARISING OUT OF THIS AGREEMENT, SUPPORT, AND/OR THE PRODUCT (INCLUDING, WITHOUT LIMITATION, LOST PROFITS, LOST COMPUTER USAGE TIME, AND DAMAGE OR LOSS OF USE OF DATA), EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND IRRESPECTIVE OF ANY NEGLIGENCE OF BMC OR WHETHER SUCH DAMAGES RESULT FROM A CLAIM ARISING UNDER TORT OR CONTRACT LAW.**

**LIMITS ON LIABILITY. BMC’S AGGREGATE LIABILITY FOR DAMAGES IS LIMITED TO THE AMOUNT PAID BY YOU FOR THE LICENSE TO THE PRODUCT.**

**SUPPORT.** If Your order includes support for the Software, then BMC agrees to provide support (24 hours a day/7 days a week) (“**Support**”). You will be automatically re-enrolled in Support on an annual basis unless BMC receives notice of termination from You as provided below. There is a free support period during the one year warranty period.

(a) **Support Terms.** BMC agrees to make commercially reasonable efforts to provide the following Support: (i) For malfunctions of supported versions of the Software, BMC provides bug fixes, patches or workarounds in order to cause that copy of the Software to operate in substantial conformity with its then-current operating specifications; and (ii) BMC provides new releases or versions, so long as such new releases or versions are furnished by BMC to all other enrolled Support customers without additional charge. BMC may refuse to provide Support for any versions or releases of the Software other than the most recent version or release of such Software made available by BMC. Either party may terminate Your enrollment in Support upon providing notice to the other at least 30 days prior to the next applicable Support anniversary date. If You re-enroll in Support, BMC may charge You a reinstatement fee of 1.5 times what You would have paid if You were enrolled in Support during that time period.

(b) **Fees.** The annual fee for Support is 20% of the Software’s list price less the applicable discount or a flat capacity based annual fee. BMC may change its prices for the Software and/or Support upon at least 30 days notice prior to Your support anniversary date.

**VERIFICATION.** If requested by BMC, You agree to deliver to BMC periodic written reports, whether generated manually or electronically, detailing Your use of the Software in accordance with this Agreement, including, without limitation, the License Capacity. BMC may, at its expense, perform an audit, at your facilities, of Your use of the Software to confirm Your compliance with the Agreement. If an audit reveals that You have underpaid fees, You agree to pay such underpaid fees. If the underpaid fees exceed 5% of the fees paid, then You agree to also pay BMC’s reasonable costs of conducting the audit.

**EXPORT CONTROLS.** You agree not to import, export, re-export, or transfer, directly or indirectly, any part of the Product or any underlying information or technology except in full compliance with all United States, foreign and other applicable laws and regulations.

**GOVERNING LAW.** This Agreement is governed by the substantive laws in force, without regard to conflict of laws principles: (a) in the State of New York, if you acquired the License in the United States, Puerto Rico, or any country in Central or South America; (b) in the Province of Ontario, if you acquired the License in Canada (subsections (a) and (b) collectively referred to as the “**Americas Region**”); (c) in Singapore, if you acquired the License in Japan, South Korea, Peoples Republic of China, Special Administrative Region of Hong Kong, Republic of China, Philippines, Indonesia, Malaysia, Singapore, India, Australia, New Zealand, or Thailand (collectively, “**Asia Pacific Region**”); or (d) in the Netherlands, if you acquired the License in any other country not described above. The United Nations Convention on Contracts for the International Sale of Goods is specifically disclaimed in its entirety.

**ARBITRATION. ANY DISPUTE BETWEEN YOU AND BMC ARISING OUT OF THIS AGREEMENT OR THE BREACH OR ALLEGED BREACH, SHALL BE DETERMINED BY BINDING ARBITRATION CONDUCTED IN ENGLISH. IF THE DISPUTE IS INITIATED IN THE AMERICAS REGION, THE ARBITRATION SHALL BE HELD IN NEW YORK, U.S.A., UNDER THE CURRENT COMMERCIAL OR INTERNATIONAL, AS APPLICABLE, RULES OF THE AMERICAN ARBITRATION ASSOCIATION. IF THE DISPUTE IS INITIATED IN A COUNTRY IN THE ASIA PACIFIC REGION, THE ARBITRATION SHALL BE HELD IN SINGAPORE, SINGAPORE UNDER THE CURRENT UNCITRAL ARBITRATION RULES. IF THE DISPUTE IS INITIATED IN A COUNTRY OUTSIDE OF THE AMERICAS REGION OR ASIA PACIFIC REGION, THE ARBITRATION SHALL BE HELD IN AMSTERDAM, NETHERLANDS UNDER THE CURRENT UNCITRAL ARBITRATION RULES. THE COSTS OF THE ARBITRATION SHALL BE BORNE EQUALLY PENDING THE ARBITRATOR’S AWARD. THE AWARD RENDERED SHALL BE FINAL AND BINDING UPON THE PARTIES AND SHALL NOT BE SUBJECT TO APPEAL TO ANY COURT, AND MAY BE ENFORCED IN ANY COURT OF COMPETENT JURISDICTION. NOTHING IN THIS AGREEMENT SHALL BE DEEMED AS PREVENTING EITHER PARTY FROM SEEKING INJUNCTIVE RELIEF FROM ANY COURT HAVING JURISDICTION OVER THE PARTIES AND THE SUBJECT MATTER OF**

**THE DISPUTE AS NECESSARY TO PROTECT EITHER PARTY'S CONFIDENTIAL INFORMATION, OWNERSHIP, OR ANY OTHER PROPRIETARY RIGHTS. ALL ARBITRATION PROCEEDINGS SHALL BE CONDUCTED IN CONFIDENCE, AND THE PARTY PREVAILING IN ARBITRATION SHALL BE ENTITLED TO RECOVER ITS REASONABLE ATTORNEYS' FEES AND NECESSARY COSTS INCURRED RELATED THERETO FROM THE OTHER PARTY.**

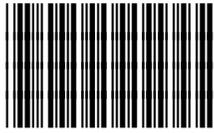
**U.S. GOVERNMENT RESTRICTED RIGHTS.** The Software under this Agreement is "commercial computer software" as that term is described in 48 C.F.R. 252.227-7014(a)(1). If acquired by or on behalf of a civilian agency, the U.S. Government acquires this commercial computer software and/or commercial computer software documentation subject to the terms of this Agreement as specified in 48 C.F.R. 12.212 (Computer Software) and 12.211 (Technical Data) of the Federal Acquisition Regulations ("**FAR**") and its successors. If acquired by or on behalf of any agency within the Department of Defense ("**DOD**"), the U.S. Government acquires this commercial computer software and/or commercial computer software documentation subject to the terms of this Agreement as specified in 48 C.F.R. 227.7202 of the DOD FAR Supplement and its successors.

**MISCELLANEOUS TERMS.** You agree to pay BMC all amounts owed no later than 30 days from the date of the applicable invoice, unless otherwise provided on the order for the License to the Products. You will pay, or reimburse BMC, for taxes of any kind, including sales, use, duty, tariffs, customs, withholding, property, value-added (VAT), and other similar federal, state or local taxes (other than taxes based on BMC's net income) imposed in connection with the Product and/or the Support. This Agreement constitutes the entire agreement between You and BMC and supersedes any prior or contemporaneous negotiations or agreements, whether oral, written or displayed electronically, concerning the Product and related subject matter. No modification or waiver of any provision hereof will be effective unless made in a writing signed by both BMC and You. You may not assign or transfer this Agreement or a License to a third party without BMC's prior written consent. Should any provision of this Agreement be invalid or unenforceable, the remainder of the provisions will remain in effect. The parties have agreed that this Agreement and the documents related thereto be drawn up in the English language. Les parties exigent que la présente convention ainsi que les documents qui s'y rattachent soient rédigés en anglais.

SW Click EULA 071102



# Notes



\*42291\*