MainView AutoOPERATOR Solutions Guide

Supporting products
Version 8.1 of MainView AutoOPERATOR
December 2016
Contacting BMC Software

Several methods are available for contacting BMC Software.

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

United States and Canada

<table>
<thead>
<tr>
<th>Address</th>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMC SOFTWARE INC 2103 CITYWEST BLVD HOUSTON TX 77042-2827 USA</td>
<td>1 713 918 8800 or 1 800 841 2031</td>
<td>1 713 918 8000</td>
</tr>
</tbody>
</table>

Outside United States and Canada

<table>
<thead>
<tr>
<th>Telephone</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>+01 713 918 8800</td>
<td>+01 713 918 8000</td>
</tr>
</tbody>
</table>

© Copyright 2006-2016 BMC Software, Inc.

BMC, BMC Software, the BMC logo, the BMC Software logo, and other BMC marks are the exclusive properties of BMC Software, Inc. and are registered or may be registered in the U.S. and in other countries. BladeLogic and other BladeLogic marks are the exclusive properties of BladeLogic, Inc. and are registered or may be registered in the U.S. and in other countries. All other trademarks or registered trademarks are the property of their respective owners.

Explorer and IOS are trademarks or registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

CICS, CICSPlex, DB2, IBM, IBM z, IMS, Language Environment, MVS, NetView, OS/390, RMF, System z, Tivoli, Tivoli Enterprise Console, VTAM, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

UNIX is the registered trademark of The Open Group in the US and other countries.

The information included in this documentation is the proprietary and confidential information of BMC Software, Inc., its affiliates, or licensors. Your use of this information is subject to the terms and conditions of the applicable End User License agreement for the product and to the proprietary and restricted rights notices included in the product 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 UNPUBLISHED—RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES.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, 2103 CITYWEST BLVD, HOUSTON TX 77042-2827, USA. Any contract notices should be sent to this address.
Customer support

Support website
You can obtain technical support from BMC 24 hours a day, 7 days a week at http://www.bmc.com/support. From this website, you can:

■ Read overviews about support services and programs that BMC offers
■ Find the most current information about BMC products
■ Search a database for problems similar to yours and possible solutions
■ Order or download product documentation
■ Download products and maintenance
■ Report a problem or ask a question
■ Subscribe to receive proactive e-mail alerts
■ Find worldwide BMC 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 1 800 537 1813 or send an e-mail message to customer_support@bmc.com. (In the subject line, enter SupID:yourSupportContractID, such as SupID:12345). Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC
Have the following information available so that Customer Support can begin working on your issue 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
  ― Messages from related software
License key and password information

If you have questions about your license key or password, contact Customer Support through one of the following methods:

- Send an e-mail message to customer_support@bmc.com. (In the Subject line, enter SupID:yourSupportContractID, such as SupID:12345.)
- In the United States and Canada, call 1 800 537 1813. Outside the United States and Canada, contact your local support center for assistance.
## Contents

### About this book

- Related publications ................................................................. 18
- Conventions ............................................................................. 19

### Chapter 1 Introduction to MainView AutoOPERATOR solutions

- EXEC distribution ...................................................................... 21
- Rules distribution ...................................................................... 21
- Documentation boxes .............................................................. 22
- Modifying solutions .................................................................. 25

### Chapter 2 Managing critical applications with Continuous State Manager

- Introduction to CSM ................................................................. 27
  - CSM requirements .................................................................. 28
  - Benefits of using CSM .......................................................... 28
  - CSM terms and concepts ....................................................... 29
  - CSM objects .......................................................................... 30
  - CSM object relationships ..................................................... 31
  - CSM object states .................................................................. 32
  - CSM schedules ..................................................................... 34
- CSM processing ........................................................................ 35
  - How CSM starts stops and cancels objects .......................... 36
  - How CSM recognizes when an object starts or stops .......... 36
  - Create start and stop Rules for CSM objects ....................... 37
  - How CSM connects and disconnects objects ....................... 37
  - Examples of how CSM works .............................................. 38
- Examples of CSM processing .................................................... 43
  - How CSM automates the IPL process .................................. 43
  - How CSM automates the shutdown process ....................... 46
  - How CSM manages routine object scheduling ................. 50
  - How CSM handles global calendar overrides ................... 51
- Plan for CSM implementation .................................................. 53
  - Determine object relationships .......................................... 54
  - Plan object schedules .......................................................... 55
  - List start stop cancel and (dis)connect commands ............. 56
  - Choose start and stop text IDs ............................................ 56
  - Choose text IDs for abnormal termination events ............. 57
Before you begin ................................................................................................................. 57
  Shared repository allocation ............................................................................................ 57
  Cross system object management with CSM groups ....................................................... 58
  CSM shared repository serialization ............................................................................. 58
  CSM security ................................................................................................................ 59
  AATWAIT object creation program .............................................................................. 59
  CSM repository back ups .............................................................................................. 59
  CSM panel navigation .................................................................................................. 60
Accessing CSM for the first time ...................................................................................... 61
  CSM panel filtering ........................................................................................................ 64
  Global Overview panel filtering .................................................................................. 64
  Mismatched object states ............................................................................................. 67
Defining CSM objects ......................................................................................................... 68
  Creating a group ........................................................................................................... 69
  Entering object information ......................................................................................... 70
  Command Specifications for an object ......................................................................... 73
  Conditional command specifications for an object ....................................................... 76
  Specifying start and stop events Event Specifications 1 ............................................. 77
  Specifying abnormal termination events Event Specifications 2 ............................... 78
  Specifying User Notification user IDs ........................................................................ 79
  Associating an object with a group ............................................................................. 81
  Moving an object between groups ............................................................................. 82
  Defining the objects parents ....................................................................................... 83
  Defining the objects servers ....................................................................................... 83
  Transient object definitions ......................................................................................... 85
  Grouping object definitions ......................................................................................... 85
  CSM object types comparisons .................................................................................. 85
  Using the repeat line command to create objects ....................................................... 86
  Modifying object definitions ....................................................................................... 87
  Adding parents ............................................................................................................ 87
  Deleting parents .......................................................................................................... 89
  Adding servers ............................................................................................................. 90
  Deleting servers .......................................................................................................... 92
  Modifying routine schedules ....................................................................................... 94
  Determining the schedule of the objects parent ......................................................... 94
  Modifying the objects schedule .................................................................................. 95
  Modifying an objects schedule with CSM schedule tags ......................................... 97
  Resolving scheduling conflicts at modification .......................................................... 99
  Overriding the calendar for special situations .......................................................... 100
  Enabling cross-system object management .............................................................. 102
Performing routine object tasks ...................................................................................... 103
<table>
<thead>
<tr>
<th>Chapter 8</th>
<th>AlarmPoint solution</th>
<th>159</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlarmPoint Standard or Professional Infrastructure Edition version 3.0.x</td>
<td>159</td>
<td></td>
</tr>
<tr>
<td>EXECs</td>
<td>159</td>
<td></td>
</tr>
<tr>
<td>Configuring the EXECs</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>Migrating to version 3.0</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>AlarmPoint version 6.x and earlier</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>EXECs</td>
<td>162</td>
<td></td>
</tr>
<tr>
<td>Configuring the EXECs</td>
<td>162</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 9</th>
<th>JES2 solution</th>
<th>163</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>163</td>
<td></td>
</tr>
<tr>
<td>Invocation</td>
<td>163</td>
<td></td>
</tr>
<tr>
<td>Flow process</td>
<td>164</td>
<td></td>
</tr>
<tr>
<td>EXECs</td>
<td>164</td>
<td></td>
</tr>
<tr>
<td>Rules</td>
<td>165</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 10</th>
<th>Storage subsystems solutions</th>
<th>167</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared DASD control solution</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>Invocation</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>Flow process</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>EXECs</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>Rules</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>TLMS solution</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>Invocation</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>Flow process</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>EXECs</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>Rules</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>Storage reply solution</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>Invocation</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>Flow process</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>EXECs</td>
<td>171</td>
<td></td>
</tr>
<tr>
<td>Rules</td>
<td>171</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 11</th>
<th>Monitoring solutions</th>
<th>173</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dump data sets monitoring solution</td>
<td>173</td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td>173</td>
<td></td>
</tr>
<tr>
<td>Invocation</td>
<td>173</td>
<td></td>
</tr>
</tbody>
</table>
### Chapter 14  MVS performance management solutions

<table>
<thead>
<tr>
<th>Solution</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load balancing solution</td>
<td>187</td>
</tr>
<tr>
<td>DASD reserve analysis solution</td>
<td>189</td>
</tr>
<tr>
<td>Exception monitoring solution</td>
<td>191</td>
</tr>
</tbody>
</table>

### Chapter 15  Error recovery solutions

<table>
<thead>
<tr>
<th>Solution</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage violation solution</td>
<td>195</td>
</tr>
<tr>
<td>CICS abnormal termination solution</td>
<td>197</td>
</tr>
<tr>
<td>VSAM subtask abnormal termination</td>
<td>198</td>
</tr>
<tr>
<td>CICS temporary storage suspensions solution (CICSTSS)</td>
<td>200</td>
</tr>
</tbody>
</table>
### Chapter 17  VTAM NCP solution 221
- TCAS reply solution ................................................................. 221
- Variables ............................................................................... 221
- Invocation ............................................................................. 221
- Flow process ......................................................................... 222
- EXECs ................................................................................... 222
- Rules .................................................................................... 222

### Chapter 18  DB2 solutions customization 223
- Customizing for DB2 solutions ............................................... 223

### Chapter 19  DB2 global operations solutions 227
- DB2 major messages (DB2ALRT solution) ............................... 227
  - Variables ............................................................................... 227
  - Invocation ............................................................................ 227
  - Flow process ........................................................................ 232
  - EXECs .................................................................................. 232
  - Rules ................................................................................... 232
- DB2 environment set-up (DB2INIT solution) ............................ 232
  - Variables ............................................................................... 232
  - Invocation ............................................................................ 233
  - Flow process ........................................................................ 234
  - EXECs .................................................................................. 234
  - Rules ................................................................................... 234
- MainView for DB2 major messages (DMRALRT solution) .......... 234
  - Variables ............................................................................... 235
  - Invocation ............................................................................ 235
  - Flow process ........................................................................ 236
  - EXECs .................................................................................. 236
  - Rules ................................................................................... 236
- MainView for DB2 utilities (DMRUTIL solution) ...................... 237
  - Variables ............................................................................... 237
  - Invocation ............................................................................ 237
  - EXECs .................................................................................. 237
  - Rules ................................................................................... 238
## Chapter 20  DB2 resource contention analysis solutions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 table space filling up solution (DB2TFUL)</td>
<td>239</td>
</tr>
<tr>
<td>Variables</td>
<td>239</td>
</tr>
<tr>
<td>Invocation</td>
<td>241</td>
</tr>
<tr>
<td>Flow process</td>
<td>241</td>
</tr>
<tr>
<td>EXECs</td>
<td>241</td>
</tr>
<tr>
<td>Rules</td>
<td>241</td>
</tr>
<tr>
<td>DB2 thread control solution (DB2THRD)</td>
<td>242</td>
</tr>
<tr>
<td>Variables</td>
<td>242</td>
</tr>
<tr>
<td>Invocation</td>
<td>242</td>
</tr>
<tr>
<td>Flow process</td>
<td>243</td>
</tr>
<tr>
<td>EXECs</td>
<td>243</td>
</tr>
<tr>
<td>Rules</td>
<td>243</td>
</tr>
</tbody>
</table>

## Chapter 21  DB2 performance management solutions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 runaway query control solution (DB2RNWY)</td>
<td>245</td>
</tr>
<tr>
<td>Variables</td>
<td>246</td>
</tr>
<tr>
<td>Invocation</td>
<td>246</td>
</tr>
<tr>
<td>Flow process</td>
<td>246</td>
</tr>
<tr>
<td>EXECs</td>
<td>247</td>
</tr>
<tr>
<td>Rules</td>
<td>247</td>
</tr>
<tr>
<td>DB2 response time solution (DB2RESP)</td>
<td>248</td>
</tr>
<tr>
<td>Variables</td>
<td>248</td>
</tr>
<tr>
<td>Invocation</td>
<td>248</td>
</tr>
<tr>
<td>Flow process</td>
<td>249</td>
</tr>
<tr>
<td>EXECs</td>
<td>250</td>
</tr>
<tr>
<td>Rules</td>
<td>250</td>
</tr>
</tbody>
</table>

## Chapter 22  E-mail solution

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-mail solution</td>
<td>251</td>
</tr>
<tr>
<td>EXECs</td>
<td>251</td>
</tr>
<tr>
<td>QAOSMTMP1</td>
<td>252</td>
</tr>
<tr>
<td>QAOEMAIL</td>
<td>255</td>
</tr>
</tbody>
</table>

## Chapter 23  SNMP solution

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>QAOSNMP</td>
<td>257</td>
</tr>
<tr>
<td>EXECs for traps that contain one OID</td>
<td>258</td>
</tr>
<tr>
<td>QAOSNMP1</td>
<td>258</td>
</tr>
<tr>
<td>EXECs for traps that contain multiple OIDs</td>
<td>262</td>
</tr>
<tr>
<td>QAOSNMP3</td>
<td>262</td>
</tr>
</tbody>
</table>
### Variables for QAOSNMP1 and QAOSNMP3

Chapter 24  
**Paging solution**  

EXECs  
- QAOSNPP1  
- QAOSNPP2  
- Parameters for MODE(SEND)  
- Parameters for MODE(STATUS)  

Flow process  

Variables returned from QAOSNPP2

Chapter 25  
**QAOUSS solution**

Chapter 26  
**Sample solutions**

Distributed sample solutions  

EXEC naming conventions  

Standards for sample solutions  

Sample VTAMNCP solution  
- Variables  
- Invocation  
- Flow process  
- EXECs  
- Rules  

Sample operations monitors solution  
- Variables  
- Invocation  
- Flow process  
- EXECs  
- Rules  

Sample CICSPlex solution  
- Invocation  
- EXECs  
- Rules  

Sample infosystem management solution  
- Invocation  
- EXECs  
- Rules  

Sample MVALARM solution  
- Invocation

Appendix A  
**CSM errors**
Creating a cross reference report ................................................................. 287
CSM environmental information ................................................................. 288
CSM EXECs for diagnostics ...................................................................... 288
CSM EXEC management .......................................................................... 289
CSM EXEC activity ................................................................................. 289
CSM EXEC traces .................................................................................. 290
CSM start and stop Rules ........................................................................ 290
Common questions and answers ............................................................... 292
Starting and stopping errors ..................................................................... 293
Restart Only objects ............................................................................... 294
Problems with special objects .................................................................. 295
CSMACT usage ..................................................................................... 298
BBI terminal sessions issues .................................................................... 300
TSOE or environmental problems ............................................................. 300
CSM installation issues .......................................................................... 301
Other tips and questions ......................................................................... 301
Maximize CSM implementation ................................................................ 303

Appendix B  CSM panels 305
CSM Global Overview panel ................................................................. 305
Defining and modifying object panels ...................................................... 313
Object Detail Control panel .................................................................. 314
Command Specifications panel ............................................................... 316
Conditional Command Specifications panel ........................................... 321
Event Specifications 1 panel ................................................................. 322
Event Specifications 2 panel .................................................................. 325
User Notification panel ......................................................................... 327
Object Groups panel ............................................................................. 328
Object Requisites panel ........................................................................ 329
Client Server Relationships panel .......................................................... 331
Object Scheduling panel ...................................................................... 333
Global Calendar Override panel .............................................................. 336
Object Groups panel 2 .......................................................................... 339
Group Detail Control ........................................................................... 342

Appendix C  Sample EXECs 345
JESDOWN sample EXEC ......................................................................... 345

Appendix D  Sample REXX EXECs 347
REXX compound variable initialization EXECs ...................................... 347
REXX DASD reserve analysis EXEC ....................................................... 349
REXX QUEUE PULL and VPUT VGET usage ............................................................. 352

Appendix E  MVS solution variables .............................................................. 355
Variables ............................................................................................................. 355

Appendix F  CICS solution variables ............................................................. 357
Variables ............................................................................................................. 357

Appendix G  DB2 solution variables and parameters ...................................... 361
Data sets containing variables .......................................................................... 361
Data sets containing threshold parameters ...................................................... 362
Variables and parameters .................................................................................. 362
  All DB2 solutions ........................................................................................... 362
  DB2RNWY threshold parameters .................................................................. 363
  DB2TFUL variables ......................................................................................... 364
  DB2THRD variables ......................................................................................... 365
  DB2RESP monitor thresholds ........................................................................ 365
  DB2RESP variables ......................................................................................... 366
About this book

This book documents solutions for IBM MVS, CICS, DB2, as well as other solutions that are distributed by BMC to run under MainView AutoOPERATOR. This book is intended for installers who install MainView AutoOPERATOR solutions and administrators who maintain MainView AutoOPERATOR solutions.

Throughout this book, references to z/OS support also include support for IBM MVS operating system and the IBM OS/390® operating system.

Like most BMC documentation, this book is available in printed and online formats. To request printed books or to view online books and notices (such as release notes and technical bulletins), see the support website at http://www.bmc.com/support.

**Note**

Online books are formatted as PDF or HTML files. To view, print, or copy PDF books, use the free Adobe Reader from Adobe Systems. If your product installation does not install the reader, you can obtain the reader at http://www.adobe.com.

The software also offers online Help. To access Help, press F1 within any product or click the Help button in graphical user interfaces (GUIs).

You must have MainView AutoOPERATOR installed to run MainView AutoOPERATOR solutions. In addition, the following MainView components are required for Rule Sets as shown in Table 1.

**Table 1: MainView components required for Rule sets**

<table>
<thead>
<tr>
<th>MainView Components</th>
<th>Rule set: TYPE</th>
<th>AAORUL00 MVS</th>
<th>AAORULM1 MVS</th>
<th>AAORULM1 CICS</th>
<th>AAORULD1 —D8 DB2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MainView AutoOPERATOR for z/OS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MainView AutoOPERATOR for CICS</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MainView AutoOPERATOR for IMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>SYSPROG Services</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MainView for IMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>MainView for CICS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Related publications

From the BMC Support Central website, you can use the following methods to access related publications that support your product or solution:

- Link to the BMC Documentation Center to browse documentation sets ([http://www.bmc.com/available/documentation-center.html](http://www.bmc.com/available/documentation-center.html) or, for secured documentation sets, [http://www.bmc.com/available/documentation-center-secure.html](http://www.bmc.com/available/documentation-center-secure.html)).

- View Quick Course videos (short overviews of selected product concepts, tasks, or features), which are available from the following locations:
  - Documentation Center (primary center and secured center)
  - BMC Mainframe YouTube channel ([https://www.youtube.com/user/BMCSoftwareMainframe](https://www.youtube.com/user/BMCSoftwareMainframe))

- View individual product documents (books and notices) within the “A – Z Supported Product List” ([https://webapps.bmc.com/support/faces/az/supportlisting.jsp](https://webapps.bmc.com/support/faces/az/supportlisting.jsp)).

Products with online interfaces also offer online Help via the **F1** key or, for graphical user interfaces (GUIs), via a **Help** button.

---

**Tip**

If you prefer hardcopy documentation, you can order it from your BMC sales representative or from Support Central. Also, from Support Central you can subscribe to receive proactive e-mail alerts when BMC issues notices.
Conventions

This document uses the following special conventions:

- All syntax, operating system terms, and literal examples are presented in this typeface.

- Variable text in path names, system messages, or syntax is displayed in italic text: `testsys/instance/fileName`

- Menu sequences use a symbol to convey the sequence. For example, Actions => Create Test instructs you to choose the Create Test command from the Actions menu.
Introduction to MainView AutoOPERATOR solutions

BMC provides a platform for developing automated solutions to problems within a data center. This platform consists of Rules, EXECs, timer facilities, and SYSPROG Services commands.

Using this platform, BMC has developed solutions to problems that are common across many data centers. These solutions assist you in accomplishing many of the initial automation tasks quickly and efficiently after you install the MainView AutoOPERATOR product.

EXEC distribution

All EXECs for solutions are distributed in the BBPROC data set allocated to the SYSPROC DD statement in the MainView AutoOPERATOR subsystem.

Rules distribution

All Rules for MainView AutoOPERATOR solutions are distributed DISABLED in the BBPARM data set.

The distribution of Rules within this data set differs depending on the environment of the solution.

Table 2: Rules distribution in different environments

<table>
<thead>
<tr>
<th>Solution environment</th>
<th>Distribution of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVS and CICS</td>
<td>one set of Rules for all solutions in one member (AAORULM1 and AAORULC1, respectively)</td>
</tr>
<tr>
<td>DB2</td>
<td>one set of Rules for each solution in a separate member (AAORULD*)</td>
</tr>
</tbody>
</table>
For information about how to enable any of these sets of Rules, any Rules within the individual solutions, or other sets of Rules in the MainView AutoOPERATOR product, see the Rules chapters of the *MainView AutoOPERATOR Basic Automation Guide, Volume 1: Using Rules.*

**Documentation boxes**

The MainView AutoOPERATOR solutions EXECs are distributed with a documentation box.

The following figure shows an example of a documentation box.

**Figure 1: Documentation box fields**

```plaintext
****************************************************************************
/* DOC GROUP(MVS) FUNC(MONITORS) CODE(MN) */
/* DOC DISP(YES) AUTHOR(B&B) */
/* DOC DESC(RESOLVE WTO BUF SHORTAGE) */

/* NAME: */
/* IEE249I */

/* DESCRIPTION: */
/* RESOLVE THE WTO BUFFER SHORTAGE SITUATION */

/* SEE ALSO: */
/* IEA404A */
/* IEA405E */

/* INVOKED BY: */
/* IEA404A */
/* IEA405E */

/* INPUT PARAMETERS: */
/* NONE */
```
/* OUTPUT: */
  /*    NONE */
/* */
/* OUTBOARD CALLS: */
  /*    NONE */
/* */
/* EXTERNAL ROUTINES CALLED: */
  /*    MVS "K" OR "Control" COMMANDS ARE ISSUED. */
  /*    NOTE: FOR THE "K" COMMANDS TO BE PROPERLY AUTHORIZED, THEY MUST BE */
  /*    ---- ISSUED FROM A NON-SUBSYSTEM CONSOLE. EDIT THE MEMBER, */
  /*    BBISSPOO, IN YOUR BBPARM DATA SET AND ADD THE KEY WORD, */
  /*    CMDCON=1; THIS WILL DIRECT ALL COMMANDS WHICH DO NOT REQUIRE */
  /*    A RESPONSE TO CONSOLE ID 1. */
/* */
/* ALERTS ISSUED: */
  /*    NONE */
/* */
/* VARIABLES USED: */
  /*    NAME:       DESCRIPTION:          DEFAULT VALUE: */
  /*    MSG         MESSAGE ID            IEE249I */
  /*    P1          WORD 1                CUA OF CONSOLE OR SYSLOG */
  /*    P2          WORD 2                CONSOLE ID */
  /*    P3          WORD 3                COND= */
  /*    P4          WORD 4                AUTH= */
  /*    P5          WORD 5                NBUF= */
/* */
/* TEST SCRIPT INFO: */
  /*    GENERATE A WTO FROM A TEST EXEC TO TRIGGER A CALL TO EITHER OF */
  /*    THE DRIVER EXECs: IEA404A OR IEA405E */
/* */
/* RETURN CODES: */
  /*    NONE */
/* */
/* CHANGE LOG: */
The information contained at the top of the documentation box is used by the MainView AutoOPERATOR EXEC Management Application to display information about the EXECS in the SYSPROC libraries. GROUP, FUNC, CODE, DISP, AUTHOR, and DESC are displayed on one line by the MainView AutoOPERATOR EXEC Management Application (see to the MainView AutoOPERATOR Basic Automation Guide, Volume I: Using Rules for more information). The rest of the information in the documentation box is for self-documentation and easy reference.

The following table describes the documentation box fields.

### Table 3: Documentation box fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP</td>
<td>group to which this EXEC belongs (for example, MVS, CICS, DB2, or IMS)</td>
</tr>
<tr>
<td>FUNC</td>
<td>functional category within the group</td>
</tr>
<tr>
<td>CODE</td>
<td>two-character function code</td>
</tr>
<tr>
<td>DISP</td>
<td>whether information about this EXEC should be displayed under the EXEC Management Application (refer to the MainView AutoOPERATOR Basic Automation Guide, Volume I: Using Rules)</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>author of the EXEC</td>
</tr>
<tr>
<td>DESC</td>
<td>description of what the EXEC does (25 characters or fewer)</td>
</tr>
<tr>
<td>NAME</td>
<td>name of the EXEC</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>detailed description of what the EXEC does</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>cross-reference listing</td>
</tr>
<tr>
<td>INVOKED BY</td>
<td>list of other EXECS that call this one</td>
</tr>
<tr>
<td>INPUT PARAMETERS</td>
<td>parameter name, description, and default value required for input parameters</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>information or names of variables that are returned</td>
</tr>
<tr>
<td>OUTBOARD CALLS</td>
<td>whether any pager calls have been placed</td>
</tr>
<tr>
<td>EXTERNAL ROUTINES CALLED</td>
<td>other routines or services called, along with any special considerations for those calls</td>
</tr>
<tr>
<td>ALERTS ISSUED</td>
<td>brief text of long EXECS where ALERTs are issued; otherwise, a YES or NO is returned</td>
</tr>
<tr>
<td>VARIABLES USED</td>
<td>variable name, description, and default value of any variables used</td>
</tr>
</tbody>
</table>
### Modifying solutions

Some national characters do not translate correctly when you translate from EBCDIC format to ASCII format and then back to an EBCDIC format.

In particular, the national characters #, !, %, and @ translate differently depending on the country in which the translation takes place. BMC recommends that you avoid using national characters in your naming conventions.

You might want to customize or modify the solutions. If you do modify a solution, BMC recommends that you retain the original EXEC or Rule for reference. To do so, follow these steps:

1. Copy the solution’s components into the UBBPROC data set.
2. Make the modifications.
3. Concatenate the UBBPROC data set to the front of the BBPROC data set in your BBI-SS JCL.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST SCRIPT INFO</td>
<td>necessary environment and steps that are needed to test the EXEC if changes are required</td>
</tr>
<tr>
<td>RETURN CODES</td>
<td>what return codes (if any) are returned</td>
</tr>
<tr>
<td>CHANGE LOG</td>
<td>chronology of changes</td>
</tr>
</tbody>
</table>
Managing critical applications with Continuous State Manager

This chapter describes the Continuous State Manager (also referred to as CSM) application.

Introduction to CSM

The following list important CSM terms:

- objects
- relationships between objects
- states of objects
- schedules

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects and objects</td>
<td>With the ISPF dialogs provided by Continuous State Manager, you define relationships and establish dependencies between them, thereby enabling the application to automatically manage your system in an ordered way.</td>
</tr>
<tr>
<td>relationships</td>
<td></td>
</tr>
<tr>
<td>States</td>
<td>To refer to an object’s accessibility is to describe the state of the object. An object that is available is in an UP state and when it is not available, it is in a DOWN state.</td>
</tr>
</tbody>
</table>
### Term | Definition
---|---
Schedules | By default, Continuous State Manager assumes that any object you define is UP 24 hours a day, 7 days a week. The object’s schedule is what the application uses to determine when to make objects UP (available) or DOWN (unavailable). The schedule consists of all the times an object should be down. For example, an object might need to be brought down every Thursday from 1:00 A.M. to 3:00 A.M. for maintenance. If you do not define a schedule for an object to be DOWN, Continuous State Manager defines a default schedule where the object is UP 24 hours a day, 7 days a week. Ideally, by using CSM, you are enabling important processes such as IPL and orderly shutdowns to be managed and controlled automatically. However, if you do encounter situations where you must change the state of an object manually, CSM also provides a facility that enables you to do this modification.

---

**CSM requirements**

The MainView AutoOPERATOR for z/OS option is required for CSM to operate and a valid password must be installed.

For information about obtaining and applying passwords refer to the *Installation System User Guide*.

**Benefits of using CSM**

This section describes some of the benefits to using CSM.

CSM improves data center performance by

- eliminating potential errors due to manual interventions

Once CSM begins managing your system, manual interventions should become minimal and hence, the system operates more smoothly, with fewer errors.

- reducing IPL time up to 50%

One of the key processes that CSM performs is IPL, which reduces the amount of time spent performing an IPL.

- reducing the time needed for an orderly system shutdown

Likewise, CSM can be used to perform an orderly, controlled shutdown of the system that also saves time.
■ restarting failed objects (if needed)

Once they are properly defined to CSM, all the Started Tasks can be controlled and managed by CSM, including the automatic recovery of failed Started Tasks.

■ starting and stopping groups of objects as if they were a single object

For example, you can use CSM to start or stop all your CICS regions with a single command.

■ notifying you of problems

For example, you can be notified automatically about objects that have changes of states outside of CSM control or about objects that fail to start or stop correctly.

■ enabling you to manage objects defined on different BBI subsystems (BBI-SS PASs)

For example, CSM enables you to see objects defined on another BBI-SS PAS target and issue commands to start, stop, cancel, and bounce those objects.

### CSM terms and concepts

You should be familiar with the following CSM terms and concepts.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSM-PLEX</td>
<td>a group of BBI-SS PASs, each of which is running CSM and share a common object repository</td>
</tr>
<tr>
<td>CSM partner</td>
<td>one BBI-SS PAS within a CSM-PLEX</td>
</tr>
<tr>
<td>object group</td>
<td>a set of objects managed by a CSM partner                                                                                                                                  An object can be associated with one, several or as many as 20 object groups. Object groups replace the need for alternate databases that were used in previous releases of MainView AutoOPERATOR because once an object is defined and is associated with an object group, it (and its schedule) can be shared with other BBI-SS PASs on other OS/390 system images that are running CSM. The name of an object group can be up to eight alphanumeric characters in length.</td>
</tr>
<tr>
<td>Object repository</td>
<td>a keyed VSAM data set shared by all members of the CSM-PLEX                                                                                       A record of this data set contains all of the information related to a particular object. The object repository enables you to define CSM objects and use them and their schedules across all the CSM-PLEX.</td>
</tr>
</tbody>
</table>
**CSM objects**

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

- normal
- transient
- grouping

**Normal objects**

A normal object is a Started Task (STC) that, once defined to CSM, has its state completely controlled by CSM.

Examples of these objects are TSO, CICS regions, and JES.

CSM becomes completely responsible for automatically starting, stopping and, in some cases, restarting or recovering normal objects according to the schedule you specify when you define the object.

**Transient objects**

A transient object is a Started Task (STC) which is started by CSM once during the life of CSM but its state is not managed by CSM after it is started.

This STC might be an object that starts, performs a batch process, and terminates normally when the job is completed. CSM will start these objects and from that point on, CSM ceases to monitor or manage the object.

When a transient object is started, its state is COMPLETE, where a normal object is UP. In other words, CSM does not monitor whether a transient object is UP or DOWN and it never attempts to change the object’s state.

**Grouping objects**

A grouping object is a user-defined imaginary object that enables you to group other objects together.

This function is very important within CSM because by using grouping objects, you can start or stop multiple objects with a single command.

For example, you might decide to stop all your CICS regions (CICS1, CICS2, CICS3) together. You can define a grouping object, CICSPROD, to be a parent to all these objects. By stopping object CICSPROD, the children CICS1, CICS2, and CICS3 stop automatically, too.
The grouping object is defined either as a parent to a group of objects or as a child to a group of objects; therefore, you can use grouping objects to start or stop groups of objects.

In this example, if you define a grouping object named CICSPROD2 as a child to CICS1, CICS2, and CICS3 and then start CICSPROD2, the three parent objects will automatically start.

**CSM object relationships**

When objects are defined to CSM, they are defined according to their relationships to other objects in the system.

You can define the following types of object relationships:

- parent-child
- client-server

There is no limit to the number of relationships you can establish between objects.

**Parent-child relationships**

A parent-child relationship is one in which the child object is completely dependent on the parent object for operation.

For example, NET (the VTAM address space) must be fully active before starting TSO, so TSO is a child (or dependent) of parent NET.

In a parent-child relationship, when a parent object is stopped, CSM automatically stops all the child objects. Likewise, when a child object is started, CSM automatically starts the parent objects.

Every object must have at least one parent. Therefore, since all objects cannot be active without MVS, MVS is ultimately the parent of all objects in the system, although in CSM, you need to define only an object’s direct parents. You do not need to define "grandparents" of objects.

To continue with the example, NET (a child of JES2) must be started before TSO starts so you must define NET as a parent of object TSO. Note that you do not need to define JES2 and MVS as direct parents of TSO. However, when CSM stops JES2, both NET and TSO are stopped first. When TSO is started, the parents NET and JES2 are started first.
Client-server relationships

In a client-server relationship, the client object receives part, but not all, of its service from a server object.

In this relationship, the server object can be stopped without stopping the client object.

CSM can manage complex object configurations in your data center. You might find it helpful to define a diagram of the system and the relationships that objects should have before you begin defining objects to CSM.

CSM object states

Two categories of states exist in CSM:

■ object desired states

When you define an object to CSM, you can include a schedule of when the object is supposed to be DOWN. The desired state of an object is the state of the object according to how you scheduled the object.

In other words, if the object is supposed to be UP 24 hours a day, Monday through Friday, its desired state is that the object is UP during the week and DOWN on the weekends. If you do not define a schedule for an object, CSM defines a schedule for the object where the object’s desired state is UP 24 hours a day, 7 days a week.

■ object actual states

An object’s actual state is the current status of the object and it might not always match the desired state.

CSM attempts to change the actual state of each object to match the desired state; for example, if CSM finds that the actual state of an object is DOWN when it is scheduled to be UP (the desired state), CSM tries to start the object.

There are also actual states that exist between UP and DOWN (see “Actual (intermediate) states” on page 33.

Object desired states

Every time an event affects an object, CSM first modifies the desired state.

An event that affects the state of an object can be
• a scheduled change that specifies the object must be brought DOWN or brought back UP

For example, a planned IPL or shutdown of the system.

• a manual command that affects the state of the object

For example, an unplanned IPL or shutdown of the system.

CSM checks the database where the object is defined and attempts to change the actual state of the object to match the desired state. CSM can change not only the object’s actual state but also the object’s children or parents’ states.

If, for any reason, CSM cannot change the actual state to match the desired state, an ALERT is issued that explains the discrepancy. The ALERT

• uses a Help panel to explain the discrepancy

• proposes a solution

• enables you to attempt to correct the situation by entering a response in the ALERT’s response field

**Actual (intermediate) states**

Between the UP and DOWN states, CSM recognizes intermediate states that are between UP and DOWN.

They are useful to both CSM and to you for keeping track of the life of an object.

For example, when an object is brought UP (the desired state), the actual states that the object passes through are

• DOWN

• STARTING

• UP

If any object stays in the STARTING state, probably some problem exists with your object definitions. The problem is usually in the defined commands or the message IDs that are used to indicate that the object is UP.

Some intermediate states also are specific to the type of object. For example, the COMPLETE state is equivalent to UP for only transient or grouping objects. Normal objects do not have a state of COMPLETE.
Another instance of an intermediate state is when you manually change the object’s state. For example, if you bring an object UP while the object is in the middle of scheduled DOWN period, the desired state is UPEARLY (because you want the object UP earlier than planned).

Use the CSM Global Overview panel to see the actual and desired states for every object defined to CSM.

**CSM schedules**

Once you have defined an object, it is scheduled to be UP all the time unless you specify a schedule of DOWN times.

You will find that some objects need to be DOWN and this state is accomplished by creating a schedule as part of the object’s definition.

CSM enables you to define a weekly schedule for all objects. Once the object is defined with a schedule, CSM tries to match the desired state (defined by the schedule) with the object’s actual state. CSM manages the object according to the schedule, automatically bringing it UP or DOWN at the specified times.

When the object has children, they are brought UP and DOWN automatically with the parent object’s schedule. When the object has clients, they are disconnected from it and reconnected to it.

Two types of object schedules can be defined: routine schedules and exception schedules (where a routine schedule is overridden).

**Routine scheduling for objects**

A routine schedule consists of the day of the week and the time of each day that an object is to be brought DOWN.

The following example shows a schedule for when an object should be DOWN:

<table>
<thead>
<tr>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wednesdays from 3:00 P.M. to 7:00 P.M.</td>
</tr>
<tr>
<td>Fridays from 2:00 A.M. to 3:00 A.M.</td>
</tr>
<tr>
<td>Fridays from 6:00 A.M. to 9:00 A.M.</td>
</tr>
<tr>
<td>Sundays from 3:00 P.M. to 6:00 P.M.</td>
</tr>
</tbody>
</table>

Outside of these times, CSM assumes the object should be UP.

When you define objects to CSM, you should plan and compare the schedule that the object needs with the schedule of any children objects. This planning will help you when an object’s parent might have a schedule that conflicts with the child because stopping the parent at the scheduled time will stop the child first.
Therefore, the schedule for a child object must include the DOWN time of its parents. If it does not, a message prompts you during the definition of the object to correct the situation.

You can also schedule DOWN times of CSM objects with

- **DAILY**: specifies that an object should be DOWN at a specified time every day of the week (Monday through Sunday) every week
- **WEEKDAY**: specifies that the object should be DOWN at a specified time on Monday through Friday of every week
- **WEEKEND**: specifies that the object should be DOWN at a specified time on every Saturday and Sunday

### Exceptional scheduling for objects

CSM enables you to define Global Calendar Overrides for special one-time situations where your plans require that objects must be brought UP or DOWN outside the routinely scheduled UP or DOWN time.

For example, you might discover that for a particular day an object that is scheduled to be DOWN at 4:00 P.M. might need to stay UP for an additional hour. You can schedule a Global Calendar Override for the object to stay UP until 5:00 P.M. for that one day.

The Global Calendar Overrides are valid only once. You want to use Global Calendar Overrides to alter an object’s schedule whenever you foresee a change to routine schedules.

---

**Note**

The Global Calendar Override does not check for conflicting schedules between parents and dependents. Therefore, when an override is specified for an object, that object and all its dependents are taken down on the specified date and time. When the object is returned to its routine schedule, the dependents also are brought back on schedule, which eliminates the need to specify the special down times for each object affected by the temporary schedule change.

### CSM processing

In addition to defining objects, their relationships, and schedules, you must define additional information that CSM uses to actually manage the objects.

This information includes
- which commands are to be executed to start, stop, or cancel the object

CSM uses MVS commands (or EXECs) that you specify to start, stop, and cancel objects.

- which messages indicate to CSM that an object is UP or DOWN

CSM uses MVS message IDs (also called text IDs) that are generated by objects when they are started or stopped to keep track of object states.

- which connect and disconnect commands are to be used when CSM manages client-server relationships

CSM uses the connect or disconnect commands that you specify to define client-server relationships between objects.

**How CSM starts stops and cancels objects**

In the object’s definition you must specify the actual MVS command or EXEC that CSM uses when CSM starts, stops, or cancels the object.

Cancel commands are used by CSM when a stop command does not work. CSM executes these commands when an IPL, a shutdown, or an event affects an object.

You can specify either the MVS command that you might use to start, stop, or cancel the object from an MVS console or a user-written EXEC that issues the command.

You might choose to define an EXEC because you need several commands to start the object or you want to perform some additional function such as set a variable when the object starts, stops, or is canceled.

You must enter these start, stop, and cancel commands or EXECs on the same object definition panel where new objects are defined. They are a mandatory part of an object’s definition.

**How CSM recognizes when an object starts or stops**

The OS/390 messages that are generated by objects when they start or stop are very important to CSM processing.

These messages are used by CSM to keep track of the object states.

CSM uses the text IDs to automatically generate Rules that are fired every time a message is received that signals an object to either start or stop. These Rules are
called Start or Stop Rules; in turn, they schedule CSM EXECs that change the actual states of the object. These Rules reside in a CSM Rule Set that is specified in the object group definition and is not user-modifiable. All CSM Rules are defined with a strategy of FIRST.

You must enter these text IDs for objects on the same object definition panel where new objects are defined. They are a mandatory part of an object’s definition.

CSM provides default text IDs for objects that trigger the Start and Stop Rules, but you can choose to specify other text IDs. For example, for a CICS region starting, you might choose the message:

DFH1517 Region Control is being given to CICS.

This message is generated when the CICS region is completely operational.

Create start and stop Rules for CSM objects

Occasionally, you might find that the CSM-generated Start and Stop Rules for an object do not meet all of your needs.

If this situation happens, you can define your own CSM Start and Stop Rules for that object.

Note that, any time you define a Start Rule and Stop Rule, you must not attempt to add them to Rule Sets AAORULBC, AAORULBD, AAORULBE, AAORULBG or to the Rule Set identified in the object group definition. The Rules can be added to any other Rule Set.

How CSM connects and disconnects objects

You can define a client-server relationship between objects for an object that receives part of its service from another object.

For example, you might have a client-server relationship between CICS (client) and DB2 (server).

CSM uses connect and disconnect commands to link these objects together after the objects have come UP. These connect and disconnect commands are specified as part of the server’s definition and are the same commands you might use if you issued them through an OS/390 console.
Examples of how CSM works

The following discussion describes how CSM actually works and the order that CSM performs its processes.

**CSM initialization**

CSM initializes when the BBI-SS PAS initializes.

CSM processing accesses the database where all the object definitions, their start and stop commands, and the text IDs are stored.

The initialization phase checks the actual state of an object and attempts to make the actual state of the object match its desired state (which is defined by the schedule). Based on the schedules for each object, CSM issues the start commands for all the objects that are scheduled to be UP.

For example, if CSM is initialized at 9:00 A.M. Monday, CSM reads the database and finds all the objects that are, according to their schedules, supposed to be UP at 9:00 A.M. Monday.

CSM issues the start commands (or EXECs) for these objects, updates the actual state to STARTING, and waits for the messages that signify the objects have started. When CSM receives these messages, CSM Rules are triggered that mark the object’s actual state as UP and CSM has successfully started the objects in your system.

In summary, when CSM receives an object’s start message, the following actions occur:

1. The CSM Start Rule fires.
2. The CSM Start Rule schedules the CSMUP EXEC to process the event and change the object’s state to UP.

When CSM receives an object’s stop message, the following actions occur:

1. The CSM Stop Rule fires.
2. The CSM Stop Rule schedules the CSMDOWN EXEC to process the event and change the object’s state to DOWN. Sometimes the address space might terminate with an end-of-memory event. When MainView AutoOPERATOR detects an address space end-of-memory event:

   The IMFEOM EXEC is scheduled, which you can set up to call the CSMEOM EXEC. This EXEC processes the event and schedules the CSMDOWN EXEC.
You can also initialize CSM with a different database that has specific automation steps, for example, to address a disaster recovery situation. This is useful because you might have a different set of automation needs during a recovery situation. For more information about initializing CSM with a different database, refer to the *MainView AutoOPERATOR Customization Guide*.

**Routine scheduling**

At initialization, CSM also determines which objects need to be brought UP or DOWN at their scheduled times.

For these objects, CSM schedules time-initiated EXECs that will bring the objects DOWN or UP at their scheduled times.

For example, suppose that at 10:00 A.M. Monday, a CICS object is scheduled to be brought UP. When CSM initializes at 9:00 A.M., a time-initiated EXEC is scheduled to issue the start command defined for the CICS object at 10:00 A.M.

**Recover failed objects**

An object might fail in CSM in the following situations:

- The start command is issued for an object but the start message is not received within the Start Command Time Out period (as defined for each object).

- The object abends unexpectedly.

When CSM does not receive the start message for an object within the time-out period, CSM issues an ALERT. No automatic recovery is attempted but an ALERT is issued, notifying you of the failure.

However, in the second case, if an object abends and comes DOWN when it is not scheduled to be DOWN, CSM automatically attempts object recovery. When an object stops and is DOWN (actual state) and it is scheduled to be UP (desired state), CSM attempts to make the actual state match the desired state.

CSM attempts this action when CSM receives the stop message for the object and fires a Rule. The Rule checks the desired state of the object, and if the desired state (according to the object’s schedule) is UP, CSM automatically issues the start command to try to start the object. If CSM cannot start the object, CSM issues an ALERT.

You can specify how many times CSM will issue the start command to recover an object; refer to “Specify start command limits” on page 40.
**Specify start command limits**

You can define the number of times CSM attempts to start a failed object.

Suppose you specify that the start command limit is three. Then, when CSM detects a failed object, CSM automatically issues the object’s start command, trying to restart the object (first attempt). If CSM cannot start the object, an ALERT is issued that states the object’s state is DOWN and restarting has failed.

If the object does start and fails again at some later time, CSM will issue the start command (second attempt). If restarting the object is successful and the object fails again, CSM will issue the start command (third and final attempt).

If the object fails a fourth time, CSM does not attempt to restart the object and issues an ALERT to notify you of the failure.

When an object has exceeded its Start Command Limit, it can be restarted in one of following ways:

- You can reply to the ACM750A ALERT.
- You can use the RESET line command (and answer Y (for yes) on the last prompt) on the Global Overview panel.
- You can issue the command `%CSMACT object RESETC on the COMMAND line and start the object from the panel.
- You can schedule the EXEC `%CSMXLCMD object RETRY.

**Specify abnormal termination events**

As part of an object’s definition, you can specify an abnormal termination event.

This event occurs when an object unexpectedly terminates (abends) and a message is issued. The abnormal termination event informs CSM that an object that is supposed to be UP is no longer UP.

If an abnormal termination message ID (or text ID) is included as part of the object’s definition, when an abend occurs CSM receives the text ID indicating the object has terminated. CSM automatically defines a MainView AutoOPERATOR ALERT to notify you about the abend. The state of the object is ABENDED.

In addition, you can specify that an EXEC or command is issued when CSM receives the text ID. The command or EXEC can attempt to rectify the situation or even restart the object.

For example, when an IMS database procedure ends unexpectedly, a database recovery job must be run before IMS can be restarted. You can specify that when an
IMS event abends and CSM receives the abnormal termination text ID, an EXEC is scheduled. The EXEC can run the database recovery program.

If an EXEC or command is not specified and an abnormal termination event is specified, when the event occurs CSM defines a MainView AutoOPERATOR ALERT stating that no recovery command or EXEC has been defined.

If a recovery command or EXEC does start the object, the object’s state is UPMANUAL. To restart an object with CSM and maintain the object’s state, issue the command:

```
CSMEXEC object RESET
```

from the recovery EXEC with IMFEXEC SELECT EXEC WAIT(NO) statement or as the recovery command. Restarting the object this way gives the object a state of UP.

**Manually interfering with CSM processing**

Ideally, you should use CSM to handle the routine management of all the objects in your system, which means that all objects are defined in proper relationship to other objects and the schedules are correctly and accurately planned.

An override to the routine schedule of an object can be defined and CSM will handle the one-time deviation from a routine schedule.

However, you might decide to (or have to) start, stop, or cancel an object at the last minute. You can do these actions from CSM in three ways:

- from the CSM panels
- with the CSMEXEC EXEC
- with the CSM command line interface

If you use any of these methods, CSM is still able to track the state of the object.

**Start stop and cancel objects from CSM panels**

You can issue start, stop, and cancel commands for an object from the Global Overview panel when you want to change the state of the object outside of its regular schedule.

You can use the following line commands to manage objects from the panel:

- (S)tart/sto(p)
- (C)ancel
manage objects across BBI-SS PASs

When cross-system object management is enabled, CSM enables you to monitor and manage objects that are managed by CSM on another BBI-SS PAS.

Cross-system management can be performed from several CSM panels.

The Global Overview panel enables you to issue commands against any CSM object running on a CSM enabled subsystem through either the default monitoring mode or the exception mode.

If you want see the objects for only a particular group, you can select that specific group. On the Object Group panel, use the LIST line command to show the objects of a specific group.

While viewing CSM objects on a remote BBI-SS PAS, you can enter the following line commands to remotely manage objects on the other BBI-SS PAS:

- (S)tart/sto(p)
- (C)ancel
- b(O)unce where an object is stopped and immediately restarted
- rese(T) where an object is returned to its defined schedule
- (M)ove
- e(X)ec where an EXEC that is associated with a CSM ALERT is scheduled

Start stop and cancel objects with the CSMACT EXEC

You can use the BMC Software distributed EXEC CSMACT to start and stop objects outside of the schedule.

You might use the CSMACT EXEC when:

- Changing the state of an object from the OS/390 console
Examples of CSM processing

The following sections provide examples of CSM processing.

How CSM automates the IPL process

When CSM initializes after the BBI-SS PAS, the first thing CSM does is read the database where all the object definitions, schedules, and relationships are stored.

CSM issues the start commands for all the objects that are scheduled to be brought UP and schedules time-initiated EXECs for the objects that are scheduled to be brought UP or DOWN at a later time.

This section provides an example of the order in which objects are brought UP when CSM initializes.

The basic principle during CSM initialization is that CSM always attempts to start all the direct children of an object automatically after the parent object is UP. In parent-child relationships, children objects cannot operate without the parent object. When CSM initializes after an IPL, none of the objects’ actual states are UP (except MVS) so CSM tries to start all the objects that are defined in its database.

The numbers next to the objects in this figure denote the order in which CSM attempts to start the objects.

Figure 2: Example of the order CSM brings up objects during IPL
Continuous State Manager (CSM) – MainView AutoOPERATOR

During IPL

This section describes the steps CSM takes during IPL of a simplified system.

In the figure in “How CSM automates the IPL process” on page 43, MVS is the first object UP and the parent (either directly or indirectly) of all the other objects in the system.

As IPL continues, CSM checks to see if JES (the only direct child of MVS) is UP. Finding JES to be DOWN, CSM attempts to bring JES UP and issues the start commands specified for the JES object. When CSM receives the start message
indicating that JES is UP, the Start Rule fires and calls the CSMUP EXEC, which changes the actual state of JES to UP. Starting of the JES object is now complete.

Then, CSM attempts to start the three direct children of JES: TMSINIT, NET, and DB2. CSM uses the same process to start these objects as it did for JES.

When CSM recognizes that NET is UP, CSM attempts to start the two direct children of NET: TSO and CICS. When CICS is UP, CSM connects CICS and DB2 in a client-server relationship.

Note
In the event that a parent object fails to start, the children are not started. In this example, if NET failed to start, CSM would not attempt to start its children objects.

After IPL is complete

After the IPL process is complete, CSM is constantly aware of the state of all the objects so that it continually tries to match the actual state of the objects to the desired state (which is dictated by the object’s schedule).

CSM takes actions when any of the following situations occur:

- An object’s schedule dictates that it is supposed to be brought DOWN at a specific time and then brought back UP at a later time

For example, CICS might have a scheduled down time every other Wednesday from 12:00 P.M. to 1:00 P.M. CSM automatically performs scheduled processes at the specified times (see “How CSM manages routine object scheduling” on page 50.

- An object fails unexpectedly

If an object fails or abends, CSM attempts object recovery by issuing the start command for the failed object (see “Recover failed objects” on page 39.

- An operator requests an action

For example, from the CSM panels or from the CSMACT EXEC, you might start, stop, cancel, or bounce an object.

- An Administration task is performed

For example, if you have access to Administration functions in CSM, you might define or change a schedule for an object. Or you might issue a Global Calendar Override for an object.

Benefits

Having CSM manage the IPL process greatly reduces the chances of operator errors.

You also can save time because CSM automatically starts objects immediately after CSM recognizes that their parent objects are UP. Another benefit is that CSM notifies you automatically with an ALERT if an object fails to start.

How CSM automates the shutdown process

Using CSM for system shutdowns enables you to perform the shutdown in an orderly manner and by issuing only a single command SHUTSYS.

Just as in the IPL of a system where CSM brings parent objects UP before bringing any children objects UP, during shutdown, CSM stops children objects first before stopping parent objects. This chain of events is initiated by simply issuing the SHUTSYS command for the group of objects that you want to stop.

From the COMMAND line on the CSM panels, type SHUTSYS group name. You are prompted for confirmation. By specifying Yes, CSM begins the shutdown of the entire system beginning with the objects furthest away from the MVS parent.

The numbers next to the objects in this figure denote the order in which CSM attempts to stop the objects.

Figure 3: Example of shutdown
Example of shutdown

When the **SHUTSYS** command is issued against a group of objects, CSM finds the child that is both furthest away from MVS and has the most dependencies to operate.

CSM proceeds to stop the children objects of NET (TSO and CICS) and the child object of JES (DB2).

CSM issues the stop commands for these objects and receives the stop messages. The Stop Rules fire and the CSMDOWN EXEC changes the actual state of each of the objects to **DOWN**.
The same process occurs for the next object stopped, which is NET. Then JES is
stopped.

Note that TMSINIT is omitted during this process because TMSINIT is a transient
object and CSM only starts transient objects: CSM does not issue stop commands for
transient objects.

TMSINIT might have already stopped when the shutdown began. If not, when CSM
completes the shutdown, TMSINIT might still be UP and you must manually stop it.

After shutdown, all the objects have a desired state of DOWNFORCE. If any objects
in the system were not defined to CSM, they might still be running and you must
shut them down manually.

The greatest benefit is that you do not have to manually shut down all the objects,
which can be time-consuming and tedious. Also, CSM already knows which objects
to stop first. Using CSM to shut down the system means that time and effort are
saved.

The figure below shows a shutdown, a 15 minute system unavailability (note that the
shutdown itself might take as much as 15 minutes), and an IPL. The numbers next to
the objects denote the order in which CSM brings objects DOWN during shutdown
and UP during IPL.

Figure 4: Example of shutdown and IPL
Example of shutdown and IPL

MONDAY IPL

Shutdown 7:00 am

IPL 7:15

Note: The column of numbers to the left of each group indicates the different order in which Objects are handled for Shutdown versus IPL.
How CSM manages routine object scheduling

CSM schedules time-initiated EXECs to start and stop objects.

**Scheduling to start objects**

For example, suppose IPL occurs at 9:00 A.M. and an object is scheduled to be started at 10:00 A.M. Prior to 10:00 A.M., the desired and actual states are DOWN. At 10:00 A.M., a time-initiated EXEC sets the desired state to UP and CSM detects that the object’s actual state is still DOWN.

CSM attempts to make the two states match by automatically issuing the start command for the object. The object’s start message is received, the Start Rule is fired, and the CSMUP EXEC executes and changes the actual state to UP. Now the actual and desired state match (UP).

**Scheduling to stop objects**

The scheduled stopping of an object happens much the same way as starting an object.

At 10:00 P.M., the time-initiated EXEC changes the desired state to DOWN, which does not match the object’s actual state of UP so CSM issues the stop commands. The object’s stop messages are received, the Stop Rule is fired, and the CSMDOWN EXEC executes and changes the actual state to DOWN. Now, the actual and desired state match (DOWN).

**Example**

The figure below shows a scheduled down time for the object NET and illustrates what happens to the dependents when NET is scheduled for DOWN time every Friday from 1:00 P.M. to 1:30 P.M. NET has a child object of CICS. DB2 is a server to client CICS.

CSM stops the dependents of an object first; in this case, CICS. First, CSM disconnects DB2 from CICS with the disconnect command that was specified when DB2 was defined as a server to CICS. DB2 does not have to come down because it does not rely on CICS to function.

With DB2 disconnected, CSM stops CICS and, when CICS is completely DOWN, CSM stops NET. Note that the shutdown of CICS and NET might each take a few minutes so they might not actually be DOWN until 1:10 P.M.
At 1:30 P.M., CSM starts the parent object NET and, when NET is completely UP, starts child object CICS. After CICS is UP, CSM reconnects CICS to DB2. This process will occur every Friday.

**Figure 5: Example of scheduled down time**

**How CSM handles global calendar overrides**

Occasionally, you might have to schedule an override to an object’s routine schedule.

There are two types of overrides:

- One type is where you add or delete UP or DOWN times from an object’s routine schedule.
- Another type is where you extend the object’s routine scheduled UP or DOWN times.

**Example 1**

This section shows a Global Calendar Override where an additional down time is added to the routine schedule of object CICS.
CICS has a routine schedule to be DOWN on Fridays from 1:00 P.M. to 1:30 P.M. when its parent, NET, is DOWN (because the child object cannot operate without the parent object).

Figure 6: Example of global calendar override: add down time

This example also shows CICS has a Global Calendar Override scheduled that brings CICS DOWN from 6:00 P.M. to midnight in addition to its routine down time. At 6:00 P.M., CSM disconnects CICS from DB2, brings CICS DOWN, and reconnects to DB2 when CICS comes UP at midnight. Notice that the server, DB2, can stay UP through both of CICS’ scheduled down times.

All Global Calendar Overrides are valid only once. After an override is complete, CSM returns to routine schedules.

Example 2

This section shows an instance where a Global Calendar Override is scheduled to bring NET DOWN an hour earlier than the routine schedule specifies.

Figure 7: Example of global calendar override: extend down time
CSM does not explicitly check to see if the CICS object’s schedule conflicts with its parent for an override. In this example, even though the override is for the object NET, the override also applies to CICS because CICS is a child of NET.

When CSM brings DOWN NET and CICS an hour earlier than the routine scheduled time, DB2 is disconnected from CICS an hour earlier. When NET is started, CICS is also started and reconnected to DB2.

### Plan for CSM implementation

Effective planning, before you ever define the first object, is the most important part of getting CSM to manage your system.

The following lists the items you must plan for a smooth CSM implementation:

- Identify objects and their relationships

You must identify all the objects and how they are related or dependent on one another because CSM uses these dependencies to manage the system in a controlled way.

- Map out periods of object inactivity
All the objects need to have their DOWN periods identified and then scheduled. All objects that have parents must include the DOWN time for the parents with their own DOWN time.

- List start, stop, cancel, and (dis)connect commands

You should identify all the start, stop, and cancel commands that will be used by CSM to start, stop, and cancel all the objects. For client-server relationships, you should list the connect and disconnect commands that CSM will use to establish those relationships.

- Choose Start and Stop text IDs

The text IDs of objects actually notify CSM of the object’s change in status.

- Choose text IDs for objects where you want to define an abnormal termination event

As part of an object’s definition, you can define that an abnormal termination event occurs and a MainView AutoOPERATOR ALERT is issued if the object terminates unexpectedly (abends). In addition, CSM can automatically issue an EXEC or command, but first you must determine the text ID of the message that is issued when the object abends.

**Note**

The best source of information for developing your object definitions is to work with the operators who regularly IPL and shut down the system. Another good source of information is the systems programming group. Both of these groups should have a good idea about which objects should be started or stopped in the system and how they are related to each other.

Many data processing installations also have documentation that describes the Started Task structure in their systems. Operations training materials might also cover this subject.

---

**Determine object relationships**

The first step in defining a system to CSM is to list all the objects running in that system and determine the relationships the objects have to one another.

Guidelines: Put MVS at the top (MVS collectively refers to all those tasks automatically started by the operating system at IPL time).

Follow MVS with the objects as they are manually started during an IPL. Pay special attention to points in the process where the startup of one object is the prerequisite...
for the startup of one or more other objects. This type of relationship is a direct parent-child relationship.

Parent-Child relationships: A good example of a direct parent-child relationship is the group of objects that usually start after the VTAM network starts. In this case, the VTAM network Started Task is the parent and the group of objects that can be started immediately after the VTAM are the children.

Client-Server relationships: Another type of relationship to note is when an object acquires services from other objects: this type of relationship is a client-server relationship. An example of a client-server relationship is one between a CICS terminal-owning region and a CICS application-owning region.

Both regions can run on their own and have functions that can be performed without the other. The terminal-owning region (client), however, requires the services of the application-owning region (server) to access CICS applications.

Noting the direct parent-child and client-server relationships both simplifies and ensures the accuracy of the definition of these objects in the database.

Plan object schedules

Objects defined to CSM that do not specify a schedule are given a default schedule of UP 24 hours a day, 7 days a week.

Most objects fall into this category.

Some objects, usually online inquiry tasks, require periods of inactivity for system maintenance and batch file processing. CSM can control the schedule for these objects and stop the object (change its actual state to DOWN) and then restart the object after a specified period of time.

You need to determine the DOWN time and the DOWN duration for all the objects and then input these times to CSM as part of the object’s definition.

<table>
<thead>
<tr>
<th>Hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>The best place to gather object scheduling information is from the Operations staff or systems programming group.</td>
</tr>
</tbody>
</table>

If the object you are defining a schedule for is the child of an object other than MVS, you must take the scheduling of the parent into consideration to avoid conflicts. A child object can be UP only when its parent is UP.

Remember, an object that does not have a schedule always has a desired state of UP unless it is altered by a schedule, a Global Calendar Override, or a CSM ACT EXEC.
Any object that has a schedule, at some point in time, will have desired states that are not UP.

**List start stop cancel and (dis)connect commands**

You must include the OS/390 commands that CSM uses to start, stop, and cancel the object as part of an object’s definition.

If the object has a server, you also need to include the commands issued to connect (and disconnect) to (and from) the server.

To obtain this information, you might consider consulting the operations or systems programming groups at your site. You also can check the manual provided for the object by its vendor. Many times, a manual published by a vendor has a command, implementation, installation, or customization chapter in which you can find the commands.

---

**Note**

Not having correct commands defined to manage an object could cause command time-outs and the object might not start or stop properly. These objects will remain in the intermediate states of STARTING or STOPPING.

---

**Choose start and stop text IDs**

The text IDs that are issued when an object has started or stopped are an integral part of an object’s definition.

CSM uses these text IDs to keep track of the objects. Therefore, if the wrong message is defined for the object, CSM will not be able to determine the actual state of the object. This situation could lead to objects being left in an actual state of STARTING or STOPPING and their states will not be recognized by CSM as UP or DOWN.

The correct Start and Stop text IDs for an object can be determined by looking at a sampling of message traffic for an object. Find this message traffic in the SYSLOG or JES message log file of the object’s output listing.

You can specify up to five Start and Stop text IDs when defining an object. If the object you are gathering information for starts before JES2, look in the SYSLOG for its start and stop messages.
Choose text IDs for abnormal termination events

As part of an object’s definition, you can define that a MainView AutoOPERATOR ALERT is issued if the object terminates unexpectedly (abends).

In addition, CSM can automatically issue an EXEC or command, but first you must determine the text ID of the message that is issued when the object abends.

One place you should look for abnormal termination text IDs is the job log of an abended object. Note that an object might issue several messages when it abends. You can specify up to five text IDs for abnormal terminations when defining an object. Abnormal termination events can also specify a particular text string to look for in a WORD variable (used by the Rules Processor) such as ABEND=S0C7 that will enable greater control over which abnormal termination event is handled and which is not. You must, however, specify the number of the WORD variable that they are to access (for example, specify 3 for WORD3, and 7 for WORD7 and so on).

Before you begin

The following sections describe other aspects of CSM that you should be familiar with before you begin using the application.

Shared repository allocation

CSM objects, once they are defined, reside in the shared repository data set.

This shared repository data set must be allocated before CSM can be used by any system in a CSM-PLEX. Once it is allocated, every BBI-SS PAS that is part of the CSM-PLEX must have the data set name of the shared repository that is specified in the BBIVARxx member used by the BBI-SS PAS.

Once the shared repository is allocated, objects can be used across all the BBI-SS PASs in the CSM-PLEX. Refer to "Cross-System Object Management: Using Group" for more information.

For information about the implementation steps that must be taken before you can bring up CSM, refer to the chapter "Implementing Continuous State Manager" in the MainView AutoOPERATOR Customization Guide.
Cross system object management with CSM groups

After the shared repository is designated in each of the BBIVARxx members in the CSM-PLEX, an object must be defined only one time.

The object and its schedule can be used by any system in the CSM-PLEX by associating the object with one or more groups.

A group is a collection of objects designated to run on a particular CSM partner. When a CSM object is defined, you must identify which group the object is a member of. By default, the object is associated with the active group of the local CSM. Cross-system management enables you to associate an object with up to 20 groups.

In addition, by default each CSM system on running its own BBI-SS PAS uses the object group whose name matches the BBI-SS PAS subsystem ID. To override this default, specify the CSMALTDB parameter in the MVS START command for the BBI-SS PAS.

You can also use the line commands on the Object Groups panel to manually enable another group and disable the existing group.

CSM shared repository serialization

CSM shared repository serialization is provided to make sure multiple users cannot update individual records in the repository data set at the same time.

Serialization occurs on a record-by-record basis so that more than one person can make updates at one time, but one user cannot update a record at the same time that another user is updating it.

Object entries are serialized on an object by object (or group by group) basis. If you have security access to edit objects, you issue the EDIT line command and CSM tries to get an exclusive enqueue. If it can get it, you can edit the object. The enqueue is held until you issue the SAVE command or the CANCEL command. No one else can edit that object. If the resource is being held exclusively, you receive a message indicating that another user is presently accessing the repository and the request is changed to browse.

As far as groups are concerned, you issue the EDIT line command. If CSM can get the exclusive enqueue for the group record, you can edit that group. When you are finished editing the group definition and press END, the group definition is immediately written to the repository and the enqueue is released. If another user tries to edit the group while you are doing so, they receive the appropriate message and their request is changed to browse.
CSM security

You can secure who has access to CSM and its various functions.

For more information, refer to the MainView Security Reference Manual.

AATWAIT object creation program

MainView AutoOPERATOR distributes the AATWAIT program in hilevel.BBLINK that you can use to easily create test Started Tasks (STCs).

This is a small, problem state program that consumes minimal resources.

Having the capability to easily and quickly create a set of test STCs can be helpful when you want to test proposed changes prior to making the changes to the Continuous State Manager or MainView Total Object Manager applications. By using the AATWAIT program, you can verify your changes have the desired effect without affecting the production level objects.

When using AATWAIT, you might see the following message issued (repeatedly) and the number of seconds in the message might vary from one system to the other. The number of seconds within the message, and the rate at which you see the message depends upon the system's parameter settings. The following is an example of the informational message:

AA2105I STIMERM FOR xxxxxxxx SECONDS

Refer to the MainView AutoOPERATOR Customization Guide for more information about using the AATWAIT program.

CSM repository back ups

CSM does not create backups of the shared repository automatically.

You can run a simple job stream that allocates a backup repository data set and copy the current one into the backup one. The JCL might look like the following example:

Figure 8: Sample JCL to back up CSM shared repository

```
//BAOKMZR JOB (3911), 'ZIOLKOWSKI',
//             CLASS=K,
//             MSGCLASS=R,
//             NOTIFY=&SYSUID
//BUILD EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
```
The repository is a keyed VSAM file. Any backup or restoration application that handles VSAM files can do the job. The repository can be copied while CSM is running and using the repository to be copied. You might want to consider using MainView AutoOPERATOR to schedule the backup job at a particular time each day.

### CSM panel navigation

The CSM panels that enable you to define objects and their schedules can all be accessed in the following sequence from the Object Detail Control panel:

- Object Detail Control
- Command Specification
- Conditional Commands
- Event Specification 1
- Event Specification 2
- User Notification
- Object Groups
- Object Prerequisites
- Object Scheduling
- Client/Server Relationships
To proceed through the panels one at a time, press the Enter key. The following figure shows an example of the Object Detail Control panel.

**Figure 9: Object Detail Control panel**

```
BMC Software ------- Object Detail Control - Add ------- MainView AutoOPERATOR
COMMAND ===>                                                  TGT --- KMZ1
The following options are displayed in sequence, or may
be selected by entering the three-character code
CMD - Command Specifications        EV1 - Event Specifications 1
EV2 - Event Specifications 2        NOT - User Notification
GRP - Object Groups                 REQ - Object Prerequisites
CLS - Client/Server Relationships   SCH - Object Scheduling
Object Name          ===> AAOCSM01       Member of Group      ===> KMZ1____
Object Type          ===> NORM____       Type of object ( ? for list)
Verify Force Down    ===> N (Y/N)        Restart Only Control ===> N (Y/N)
Application information:
Author               ===> BAOKMZ1
Description          ===> Child of MVS____________________________
Last Modified by BAOKMZ2 on 01/30/2001 at 15:43
Enter END command to process and return, ENTER to continue, or CANcel
```

CSM enables you to display any of the panels by using shortcut commands entered on the COMMAND line. The commands and the panels they display are listed in the following table.

Press END from the Detail Control Panel to process any changes made to the object definition and return to the Global Overview panel; press END on any other panel in the sequence to return to the Detail Control Panel.

**Table 4: Shortcut commands for navigating CSM panels**

<table>
<thead>
<tr>
<th>Command</th>
<th>Displays this panel</th>
</tr>
</thead>
<tbody>
<tr>
<td>DET</td>
<td>Detail Control</td>
</tr>
<tr>
<td>CMD</td>
<td>Command Specification</td>
</tr>
<tr>
<td>EV1</td>
<td>Event Specification 1</td>
</tr>
<tr>
<td>EV2</td>
<td>Event Specification 2</td>
</tr>
<tr>
<td>NOT</td>
<td>User Notification</td>
</tr>
<tr>
<td>GRP</td>
<td>Object Groups</td>
</tr>
<tr>
<td>REQ</td>
<td>Object Prerequisites</td>
</tr>
<tr>
<td>SCH</td>
<td>Object Scheduling</td>
</tr>
<tr>
<td>CLS</td>
<td>Client/Server Relationships</td>
</tr>
</tbody>
</table>

**Accessing CSM for the first time**

After the CSM shared repository is defined as part of implementing and customizing CSM, you can access CSM from an ISPF-based terminal session.
CSM initialization begins after MainView AutoOPERATOR initialization completes. At this time, CSM runs in minimum mode where no objects are monitored.

To access the CSM application

1. Select Option 3, Continuous State Manager, from the Automation Menu.

**Figure 10: MainView AutoOPERATOR Automation menu**

```
BMC Software ---------------- AUTOMATION MENU ---------------- AutoOPERATOR
OPTION ===>                                               DATE -- YYYY/MM/DD
            TIME -- 08:32:51

Basic Automation:
1 Event Activity Statistics
2 Display/Modify Rules and Rule Sets (Full-Screen Mode)
3 Continuous State Manager - Global Overview
4 Object Management
R Rules Management via Rules Registry (Windows Mode)

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

Utilities:
C CSM Conversion
PF1/13 HELP PF3/15: EXIT
```

The Global Overview panel is displayed.

**Figure 11: Global Overview panel**

```
BMC Software -------------- Global Overview -------------- MainView AutoOPERATOR
COMMAND ===>                                                   SCROLL ===> CSR
TGT ---  KMZ1

Primary Commands: EXCeption Group     SORT    SAVE      ADD      PROFile
CALendar        CANcel    UNsort  CONVERT   BUILD
Locate          CMDSHOW    Xref    SAVERULE  SHUTSYS

LC CMDS -------- (S)tart   b(O)unce  rese(T)    depe(N)dents  (A)dd
               (CAN)cel (M)ove    (R)epeat  (B)rowse    s(Y)nc
               (EN)able  (DIS)able

LC Object    Group    System    Actual      Desired      Ind    Type
________    ________    ________  ____________ ____________ _____ ________
___ MVS                                 NORM
___ AAOCSM01 KMZ1     SJSD      UP           UP                 NORM
___ AAOCSM01 KMZ5               INACTIVE     INACTIVE           NORM
___ AAOCSM01 KMZ6               INACTIVE     INACTIVE           NORM
___ AAOCSM02 KMZ1     SJSD      COMPLETE     COMPLETE           TRAN
___ AAOCSM02 KMZ2               INACTIVE     INACTIVE           TRAN
___ AAOCSM02 KMZ5               INACTIVE     INACTIVE           TRAN
___ AAOCSM02 KMZ6               INACTIVE     INACTIVE           TRAN
___ AAOCSM08 KMZ1     SJSD      UP           UP                 NORM
___ AAOCSM08 KMZ2               INACTIVE     INACTIVE           NORM

This example shows what the panel might look like after groups and objects have been defined.
Use this panel to monitor and manage all CSM objects and their various states, regardless of where the objects are executing. This panel shows:

- The object name and the name of the group with which the object is associated
- The MVS SYSNAME where the object and group are executing
- The actual and desired state of each object and what type each object is (for example, a normal object is NORM)
- Whether the object has a pending ALERT and, if so, does that ALERT have a follow-up EXEC associated with it

To view additional information about each object, scroll left by pressing PF10/PF22.

Figure 12: Global Overview panel: scrolled LEFT

The following information is available for each object:

- First thirty bytes of the object description
- User ID of the last person to modify the object
- Time and date of the last modification to the object

If an object is inactive (because the group that it belongs to is not enabled), the Ind column shows no information.

To return to the original panel, press PF11/PF23. The Global Overview panel is redisplayed.
CSM panel filtering

Throughout the panels in CSM, you will see column headings that are underlined, for example, as shown in the example below:

<table>
<thead>
<tr>
<th>LC</th>
<th>Object</th>
<th>Group-Id</th>
<th>System</th>
<th>Actual</th>
<th>Desired</th>
<th>Ind</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>___</td>
<td>_______</td>
<td>________</td>
<td>_______</td>
<td>_______</td>
<td>________</td>
<td>___</td>
<td>_____</td>
</tr>
</tbody>
</table>

The underlined areas under the column headings are input masking fields where you can enter a text string that qualifies (or masks) the data that is displayed to show specific information. For example, if you enter a specific SSID in the System column and press Enter, the panel shows only those objects running on that subsystem.

Use these fields to mask any of the data in any of the CSM panels where the column headings have input areas. You can also mask in more than one field.

In the input areas, you also can use an asterisk (*) as a wildcard character.

Global Overview panel filtering

The Global Overview panel shows information for all of the objects in all of the groups in the local repository.

It also shows information for objects associated with groups in remote repositories communicating with the local BBI-PAS.

If CSM is monitoring a large number of systems are monitored, it might be difficult to view specific systems on the Global Overview panel. Refreshing the information might also be time consuming.

The following figure shows an example of a large number of systems being monitored:

**Figure 13: Global Overview panel**

<table>
<thead>
<tr>
<th>BMC Software</th>
<th>Global Overview</th>
<th>MainView AutoOPERATOR</th>
<th>SCROLL</th>
<th>CSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND</td>
<td>SCROLL</td>
<td>CSR</td>
<td>TGT</td>
<td>KMZ1</td>
</tr>
<tr>
<td>Primary Commands: EXCeption</td>
<td>CALeNDAR</td>
<td>CANCEL</td>
<td>UNsort</td>
<td>CONVERT</td>
</tr>
<tr>
<td>Locate</td>
<td>CMDSHOW</td>
<td>Xref</td>
<td>SAVERULE</td>
<td>SHUTSYS</td>
</tr>
<tr>
<td>LC</td>
<td>CMD</td>
<td>DS</td>
<td>(S)tart</td>
<td>b(O)unce</td>
</tr>
<tr>
<td>(CAN)cel</td>
<td>(M)ove</td>
<td>(R)epeat</td>
<td>(B)rowse</td>
<td>s(Y)nc</td>
</tr>
<tr>
<td>(EN)able</td>
<td>(DIS)able</td>
<td>(EX)ec</td>
<td>(E)dit</td>
<td>c(L)ients</td>
</tr>
<tr>
<td>(CA)mmand</td>
<td>(DIS)able</td>
<td>(EX)ec</td>
<td>(E)dit</td>
<td>c(L)ients</td>
</tr>
<tr>
<td>LC</td>
<td>Object</td>
<td>Group</td>
<td>System</td>
<td>Actual</td>
</tr>
<tr>
<td>___</td>
<td>_______</td>
<td>________</td>
<td>_______</td>
<td>_______</td>
</tr>
<tr>
<td>___</td>
<td>_______</td>
<td>________</td>
<td>_______</td>
<td>_______</td>
</tr>
<tr>
<td>___</td>
<td>_______</td>
<td>________</td>
<td>_______</td>
<td>_______</td>
</tr>
<tr>
<td>___</td>
<td>_______</td>
<td>________</td>
<td>_______</td>
<td>_______</td>
</tr>
<tr>
<td>___</td>
<td>_______</td>
<td>________</td>
<td>_______</td>
<td>_______</td>
</tr>
<tr>
<td>___</td>
<td>_______</td>
<td>________</td>
<td>_______</td>
<td>_______</td>
</tr>
</tbody>
</table>
In this example, suppose that groups KMZ5 and KMZ6 are rarely enabled. Group KMZ1 is always enabled and group KMZ2 is occasionally inactive.

Use the PROFile command to display the Group Display Selection panel and select the CSM object groups to be included or excluded from the Global Overview panel.

Figure 14: Group Display Selection panel

The first time you display the Group Display Selection panel, all of the groups are marked as INCLUDED. By default CSM shows all of the available groups on the Global Overview panel.

In this example, use the Include and Exclude line command to include only groups KMZ1 and KMZ2 on the Global Overview panel.

As a group is included or excluded, its status changes. The names of the groups to include on the Global Overview panel are in the user’s profile. All changes that are made on the Group Display Selection panel are valid only for the TSO user ID that makes the changes and one user’s changes does not affect other users.

Groups targeted to be displayed on the Global Overview panel have a status of INCLUDED. Groups that are not targeted have a status of EXCLUDED. The Repository column indicates to the user whether the group’s definition resides in the local repository (LOCAL), in a remote repository.

You can choose to include all of the available groups by issuing the primary command INCLUDeall.
If you decide to exclude all of the groups except for groups KMZ1 and KMZ2, complete the panel as shown in the following figure and press the Enter key.

**Figure 15: Group Display Selection panel showing groups to exclude**

<table>
<thead>
<tr>
<th>BMC Software</th>
<th>Group Display Selection</th>
<th>MainView AutoOPERATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND</td>
<td>SCROLL</td>
<td>PAGE</td>
</tr>
<tr>
<td>TGT</td>
<td>KMZ1</td>
<td></td>
</tr>
</tbody>
</table>

Primary Commands: INCludeall CMDSHOW
LC CMDS --------- (E)xclude Group    (I)nclude Group
Enter END command to process and return, ENTER to continue, or CANcel

<table>
<thead>
<tr>
<th>Group</th>
<th>Status</th>
<th>Repository</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KMZTST20</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZTST21</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZTST22</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZTST23</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZ1</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td>Main Group</td>
</tr>
<tr>
<td>KMZ2</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td>Converted on: 08/17/2001 06:02</td>
</tr>
<tr>
<td>KMZ5</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td>Converted on: 08/17/2001 08:55</td>
</tr>
<tr>
<td>KMZ6</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td>Converted on: 08/17/2001 08:56</td>
</tr>
<tr>
<td>KMZ69</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td>Converted on: 08/17/2001 09:01</td>
</tr>
<tr>
<td>KMZ7</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZ70</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td>Converted on: 08/17/2001 09:05</td>
</tr>
<tr>
<td>KMZ8</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZ9</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
</tbody>
</table>

The status changes as shown in the following figure.

**Figure 16: Group Display Selection panel showing change in group status**

<table>
<thead>
<tr>
<th>BMC Software</th>
<th>Group Display Selection</th>
<th>MainView AutoOPERATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND</td>
<td>SCROLL</td>
<td>PAGE</td>
</tr>
<tr>
<td>TGT</td>
<td>KMZ1</td>
<td></td>
</tr>
</tbody>
</table>

Primary Commands: INCludeall CMDSHOW
LC CMDS --------- (E)xclude Group    (I)nclude Group
Enter END command to process and return, ENTER to continue, or CANcel

<table>
<thead>
<tr>
<th>Group</th>
<th>Status</th>
<th>Repository</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KMZTST20</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZTST21</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZTST22</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZTST23</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZ1</td>
<td>INCLUDED</td>
<td>LOCAL</td>
<td>Main Group</td>
</tr>
<tr>
<td>KMZ2</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td>Converted on: 08/17/2001 06:02</td>
</tr>
<tr>
<td>KMZ5</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td>Converted on: 08/17/2001 08:55</td>
</tr>
<tr>
<td>KMZ6</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td>Converted on: 08/17/2001 08:56</td>
</tr>
<tr>
<td>KMZ69</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td>Converted on: 08/17/2001 09:01</td>
</tr>
<tr>
<td>KMZ7</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZ70</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td>Converted on: 08/17/2001 09:05</td>
</tr>
<tr>
<td>KMZ8</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
<tr>
<td>KMZ9</td>
<td>EXCLUDED</td>
<td>LOCAL</td>
<td></td>
</tr>
</tbody>
</table>

The changes become active when you enter the END command or press PF3/PF15. You can reject all changes made by issuing the CANCEL command.
When you are ready to apply the changes made to the list of groups to be displayed, enter **END** and return control to a Global Overview panel that looks like the following figure:

**Figure 17: Filtered Global Overview panel**

```
BMC Software ------- Global Overview - * Filtered * ------- MainView AutoOPERATOR
COMMAND ===>                                                   SCROLL ===> CSR
Primary Commands: EXCeption Group SORT SAVE ADD PROFile
                CALendar CANCEL UNSort CONVERT Build
                Locate CMDSHOW Xref SAVERULE SHUTSYS
LC CMDS -------- (S)tart (B)ounce (R)epeat (B)rowse
                (C)ancel (M)ove (D)elete
                (L)C (E)xec    (E)dit (D)lexport
                (C)lients (M)ake (R)epeat
                (A)dd (E)xport
                (N)able (D)isable

LC Object    Group    System   Actual       Desired      Ind   Type
________  ________ ________ ____________ ____________ _____ ________
___ AAOCSM01 KMZ1      SJSD     UP           UP                  STC-NORM
___ AAOCSM02 KMZ1      SJSD     COMPLETE     COMPLETE            STC-TRAN
___ AAOCSM02 KMZ2      INACTIVE INACTIVE            STC-TRAN
___ AAOCSM08 KMZ1      SJSD     UP           UP                  STC-NORM
___ AAOCSM08 KMZ2      INACTIVE INACTIVE            STC-TRAN
```

The indicator * Filtered * appears in the title line of Global Overview panels that have been filtered with the PROFILE command.

Changes can be made to the user’s list of included groups as often as necessary. These changes last from one CSM-TS session to another.

**Mismatched object states**

The Global Overview panel shows the state of all the objects.

- To see only the objects whose actual and desired states do not match, enter the EXCeption primary command on the COMMAND line to display the CSM Exceptions panel.

- To toggle back to the complete display of objects, enter the MONitor primary command on the COMMAND line.

**Figure 18: CSM Exceptions panel**

```
BMC Software -------------- CSM Exceptions ------------ MainView AutoOPERATOR
COMMAND ===>                                                   SCROLL ===> CSR
Primary Commands: MONitor Group SORT SAVE ADD PROFile
                CALendar CANCEL UNSort CONVERT Build
                Locate CMDSHOW Xref SAVERULE SHUTSYS
LC Object    Group    System   Actual       Desired      Ind   Type
________  ________ ________ ____________ ____________ _____ ________
___ AAOCSM65 KMZ1     SJSD     STARTING     UP           XE    NORM
```

The CSM Exceptions panel shows a list of those objects where...
any object Actual and Desired states do not match

In the following figure, the Actual and Desired states for object AAOCSM65 do not match.

**Figure 19: CSM Exceptions panel**

---

objects have at least one outstanding ALERT

ALERTs are shown in the **Ind** column. The value XE denotes that an ALERT is associated with this object and the ALERT has an EXEC.

When an object’s Actual and Desired states do not match, you can issue the following line commands against these objects to try to change their states:

- (S)tart/sto(p)
- (CAN)cel
- b(o)unce
- (EX)ec
- (M)ove
- rese(T)

You can add, modify, delete (and so forth) or start, stop, cancel, (and so forth) any object shown on either the Global Overview or the Exceptions panels.

---

**Defining CSM objects**

This section describes how to perform the following tasks:

- Define a group which is the first step required when you bring CSM up for the first time
- Define an object with CSM panels
Make routine modifications to objects once they are defined from the Global Overview panel.

**Creating a group**

Every object must belong to a group. When you create a group, you must name the group the same name as a MainView AutoOPERATOR BBI-SS PAS.

Therefore, when you bring CSM up for the first time, the first task you must complete is to define a group before you can begin defining objects. A group name can be one- to eight-characters long but it is recommended that you name the first group with the SSID of the subsystem that you bring CSM up on.

**Note**

The default action of CSM is to bring up the object group that has the same name as the SSID. To change this default behavior, refer to the chapter named "Implementing Continuous State Manager" in the MainView AutoOPERATOR Customization Guide.

**To define a group for a subsystem named KMZ1, follow these steps:**

1. Enter the Group primary command on the COMMAND line on the Global Overview panel. The Object Groups panel is displayed.

   **Figure 20: Global Overview panel: adding a group**

   ```
   BMC Software ------------ Global Overview ------------- MainView AutoOPERATOR
   COMMAND ===>                                                   SCROLL ===> CSR
   TGT ---  KMZ1
   Primary Commands: EXCeption Group     SORT    SAVE      ADD      PROFile
   CALEndar CANcel UNsort CONVERnt BUILD
   Locate CMDSHOW Xref SAVERULE SHUTSYS
   LC CMDS -------- (S)tart b(O)unce rese(T) depe(N)onents (Add)
   do(P) (EX)ec (E)dit c(L)ients (D)elete
   GCel (M)ove (R)epate (B)rowse s(Y)nc
   LC Object   Group    System   Actual       Desired      Ind   Type
   MVS      KMZ1     SJSD     UP           UP                 NORM
   ***********************************************************************
   ** Figure 21: Object Groups panel **
   ```

2. Enter the ADD primary command on the COMMAND line to add a new object group to the repository. The Group Detail Control - Add panel is displayed.

   **Chapter 2  Managing critical applications with Continuous State Manager  69**
If you enter the ADD primary command with the name of the new group, for example: **ADD SYSBGRP**, the Group Name field is filled in when the Group Detail Control - Add panel is displayed.

**Figure 22: Group Detail Control - Add panel**

![Group Detail Control - Add Panel]

- **Group Name**
- **Command Prefix**
- **Ruleset Dataset**
- **Ruleset Id**
- **Rule Prefix**
- **Suffix**
- **Dataset Name**
- **Disposition**
- **Application information**
  - **Author**
  - **Description**

3 Fill in the information for these required fields:

- Group Name
- Command Prefix
- Ruleset/Dataset
- Ruleset Id
- Rule Prefix
- Suffix

Note: The words Minimal and Local are reserved words in CSM. Group names cannot contain these words. If you attempt to define a group with the name MINIMAL or LOCAL, the error message "Reserved Name" is displayed.

Additions, changes, and deletion of object groups take place immediately. You do not need to issue a SAVE command after adding or changing a group.

4 Press **PF3/END** to process the changes on this panel and return to the Object Groups panel.

5 Press **PF3/END** again to return to the Global Overview panel.

**Entering object information**

The following information describes some characteristics of objects:
When an object is added to the repository it is associated with (or a made a member of) the group running on the local subsystem. The object can be associated with as many as 20 groups. The object is associated with a group as a member by using the Member line command. Being a member of a group means that the object starts if it is not already up when the group is made active.

If the object can be moved from one system to another, CSM assumes that it is intended to be running on only one system at a time. In this case, the object must be a member of only one group. It can be eligible to be moved to as many as 19 groups, but can only be a member of one group. When an object has a Movable association with a group, the object will not start when that group is made active until you instruct CSM to move the object to that group’s control, even if the group is running on another MVS image in the CSM-PLEX.

The Movable line command associates the object with the group it is able to move to. If the object is a member of more than one group and the user issues a Movable line command against another group, CSM will issue an error message indicating that the object is presently a member of more than one group.

An object’s association with a group can be dissolved by using the Remove line command.

If a user wants to know what other objects are associated with a particular group, the List line command will display the names of the objects in that group.

This section shows how to define a Normal object (from the Global Overview panel) named AAOCSM01. In this example, the new object is defined as a child of the MVS parent.

**To define a new object named AAOCSM01:**

1. Perform one of the following actions:
   - Enter the ADD primary command on the **COMMAND** line from the Global Overview panel
   - Enter the ADD line command next to the name of the group where the object will belong.

   You can include the name of the object that is shown in the **Object Name** field with ADD primary command when the Object Detail Control panel is displayed.

   The Object Detail Control panel is displayed.
When you use the ADD primary command, MVS is always indicated as the object’s parent in the description field.

2 Fill in the information for these required fields:

- **Object Name**
- **Object Type**
- **Member of Group**
- **Verify Force Down**
- **Restart Only Control**

By default, the **Member of Group** field is filled in with the local subsystem ID when you use the ADD primary command from the Global Overview panel. You can also enter the name of another group for the object to be associated with.

3 Enter a question mark ( ? ) in the **Object Type** field to see a list of Object Types.

4 Select a type by placing an S next to it. Only one object type can be selected at a time.

5 Press **Enter** or **END** to return to the Object Detail Control panel.

The **Object Type** field contains the selected object type.

If you do not select an object type from the panel, the **Object Type** field contains the field value prior to entering a question mark.

When defining an object, after you specify the object type of Transient or Normal, you cannot change that specification. To change the specification, delete the object and define a new one.

6 Press **Enter** to continue to the next object definition panel, Command Specifications.
Command Specifications for an object

Specify an object start, stop, cancel and recovery commands (or EXECs) on the Command Specification panel.

You can also specify post-start and post-stop EXECs that CSM schedules after CSM starts or stops the object.

The panel provides default start, stop and cancel commands and these defaults specify the most common choices for starting, stopping and canceling an object.

The format of the default commands are as follows:

- \#S objectname to start an object
- \#P objectname to stop an object
- \#C objectname to cancel an object

The following shows an example of the Command Specifications panel.

Figure 23: Command Specifications panel

BMC Software ---- Command Specifications for AAOCSM01 - Add ---- AutoOPERATOR
COMMAND ===>                                                  TGT --- KMZ1
Start Command/EXEC      ===> #S AAOCSM01_______________________________
User Post Start EXEC    ===> __________________________________________
Stop Command/EXEC       ===> #P AAOCSM01_______________________________
User Post Stop EXEC     ===> __________________________________________
Cancel Command/EXEC     ===> __________________________________________
Recovery Command/EXEC   ===> __________________________________________
Start Command Time Out  ===> 0002         Start Command Limit ===> 3___
Stop Command Time Out   ===> 0002         Perform Count Reset ===> Y (Y/N)
Conditional Start       ===> N (Y/N)      Conditional Stop    ===> N (Y/N)
Enter END command to process and return to Detail Control. ENTER to continue

CSM simple and compound variables

On the Command Specifications panel, you can specify both simple and compound variables in the Start, Stop, Post Start, Post Stop, Cancel and Recovery command fields.

Some basic variables are supplied:

- CSMDAY: resolves to the current day of the week in three-character format
- CSMMON: resolves to the current month in three-character format
- CSMDATE: resolves to the current date in the format yyymmdd
- **CSMGROUP**: resolves to the name of the group active on the local subsystem

**CSMSYSNM**: resolves to the SYSNAME from the local subsystem’s MVS system control blocks

Variables from the SHARED pool can also be used in the **Start**, **Post Start**, **Stop**, **Post Stop**, **Cancel**, and **Recovery** command fields.

**Note**
The only variables that cannot be used for this facility are the variables prefixed with IMF that are supplied by MainView AutoOPERATOR.

### Example

An example of a start command containing a simple variable name follows:

```
#S AAOKMZSS,START=&STATYPE,VPOOL=&RESET
```

The variable is resolved when the CSM subsystem application processes the command and the variable is read from the SHARED variable pool. If the variable is not present in the SHARED pool, it resolves as the variable name (as in REXX language coding).

Using a compound variable is the same as a simple variable. For example, if an object has a start command defined as

```
#S AAOCSM09,PARM=&AO.CSMDAY
```

the variable AO.CSMDAY is treated in the same manner as a REXX compound symbol: AO is the stem and CSMDAY is the tail.

On Monday, the variable resolves into shared variable AO.MON. It contains the value STUFF. The resulting command is as follows:

```
#S AAOCSM09,PARM=STUFF
```

You can set the variable with a Rule, an EXEC, or an entry in the BBIVAR xx member. The variable must already be defined when CSM executes the defined command.

This allows you to specify Start and Stop commands that can change periodically. It also provides flexibility for object definitions in the shared repository whose Started Task name is the same across multiple partners but use significantly different parameters from system to system.
All commands entered in any of the command fields: **Start**, **Stop**, **Post Start**, **Post Stop**, **Cancel**, and **Recovery** must be prefixed by one of the following control characters:

- Period ( . )
- Pound sign (#)
- Forward slash (/)
- Question mark (?)
- Percent sign (%)

Single quotation marks can be used in commands. The quotation marks must be doubled, so that they are properly processed; for example:

```
#S ABCPDQ.PDQ,LIBRARY='SYS1.PARMLIB'
```

CSM running in the BBI-PAS strips out the extra quotation marks and sends the command to the system for execution. If an object used the this command, CSM translates to the following command:

```
#S ABCPDQ.PDQ,LIBRARY='SYS1.PARMLIB'
```

You can also use values defined in the static system symbol list. These values must be prefixed with two ampersand signs (&&); for example:

```
#S NET,,,LIST=&&SUBAREA
```

The resolved command that is passed to MVS is:

```
S NET,,,LIST=SUBAREA
```

When MVS processes this command, **&SUBAREA** is substituted with a value defined in the static system symbol list.

If the command is not prefixed with a control character, MainView AutoOPERATOR tries to process the command as an EXEC using EXEC parameters.

You can use the Command Specifications panel to also specify the following values:

- Start and Stop Command Time Out values (in minutes)
- Start Command Limit value
Whether to reset the Start Command count where a Y or N specification determines whether the object command count shared variable is reset each midnight to zero

Conditional Start or Conditional Stop setting where a Y setting in either field will cause the Conditional Commands Specification panel to appear

Press Enter to display the Conditional Command Specification panel.

Conditional command specifications for an object

If you entered Y for either the Conditional Start or the Conditional Stop fields on the Command Specifications panel, the Conditional Commands Specification panel is displayed.

The following shows an example of the Stop commands panel.

Use a conditional Start or Stop command to specify that CSM issues a command after any of the following conditions are met:
A period of minutes has elapsed.

The defined Start or the defined Stop command is issued.

Another conditional command has been issued for this object.

**Note**
The conditional command is not issued if the object's Start or Stop event is triggered.

If the Start or Stop command and all the conditional Start or Stop commands are issued, and the object Start or Stop event has not occurred, an ALERT (ACM720A) is issued informing you that the command has timed out.

1. Enter the number of minutes after which a conditional Start or Stop command will be issued for an object.

2. Enter the conditional Start or Stop command. In addition:
   - You must enter each conditional command sequentially on a new line without skipping between commands.
   - Each command can be up 126 bytes long and can contain variables. Any variables specified will be resolved from the local subsystem’s SHARED pool.
   - When a conditional command is issued after the original command, CSM issues an escalated ALERT. An example of the ALERT (ACM789A) is

   ACM789A Conditional command found for <object>.

   Unless a <start|stop> event fires for <object> the command will be issued in <xx> minutes.

3. Press **Enter** to continue to Event Specifications 1 panel.

**Specifying start and stop events Event Specifications 1**

Use the first Event Specification panel to specify text identifiers for Start and Stop events.

You can specify the text ID and type for up to five Start events and five Stop events.

For each event specification, you can specify

- the type of job that issues the event (JOB, STC or TSO)
- the name of the job that issues the event
To enter start events

1. Enter the text IDs and the event types for each of the Start or Stop events.

**Figure 24: Event Specifications 1 panel**

You must enter each text ID and event type sequentially on a new line without skipping between lines. The valid event types are MSG, CMD, ALRT, and JRNL.

2. Enter the type of job that will issue the event (JOB, STC or TSO).

3. Enter the name of the job that will issue the event. Wildcard characters + and * can be used as they would be in a Rule.

4. Press Enter to continue to the next object definition panel, Event Specifications 2.

Specifying abnormal termination events Event Specifications 2

Use the second Event Specification panel to specify (up to 5) identifiers for Abnormal Termination events.

For each Abnormal Termination event, you can specify

- the type of job that issues the event (JOB, STC or TSO)
- the name of the job that issues the event

To enter start events

1. Specify the abend message text in the **Text String** field to search for within the Text ID specified.

**Figure 25: Event specifications 2 panel**

78 MainView AutoOPERATOR Solutions Guide
Every message is parsed into separate values (delimited by a space) and stored in WORD x variables. You can specify a word to match the search string and the type of comparison operation between the search string and the WORD x variable. This allows you to define very specific circumstances for abnormal termination processing to occur.

For example, the following lines show how to specify abnormal termination processing that occurs when the fifth word of the IEF450I message contains the abend code S0C7:

```
Text Id ===> IEF450I________ Job Name ===> ________ Job Type ===> ___
Text String ===> ABEND=S0C7______ Word/Op ===> WORD5 / EQ Event Type ===> ___
```

2 Enter the Text IDs and the event types for each of the Abnormal Termination events in the Text Id and Event Type fields.

You must enter each Text ID and event type sequentially on a new line without skipping between lines.

3 Enter the type of job (JOB, STC or TSO) that issues the message in the Job Type field.

4 Enter the name of the job that issues the message in the Job Name field. You can use the plus sign (+) and asterisk (*) wildcard characters.

5 Press Enter to continue to the next object definition panel, User Notification.

### Specifying User Notification user IDs

Use the User Notification panel to

- send TSO messages
- manage the ALERT queue
- notify up to 10 TSO users of error conditions in object processing
**Note**
You cannot modify the message sent to the TSO users.

---

**To complete the User Notification information**

1. Enter any valid one- to seven-character TSO user IDs for each user to be notified of error conditions in object processing in the **TSO SEND Users** fields.

   **Figure 26: User Notification panel**

<table>
<thead>
<tr>
<th>BMC Software</th>
<th>User Notification for AA0071X</th>
<th>Edit</th>
<th>AutoOPERATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND</td>
<td>TGT --- D71A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSO SEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users</td>
<td>Pager Notification (ELAN) not supported.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Object Alert Queue ====&gt; MAIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alert Publish Mode ====&gt; ADD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enter END command to process and return to Detail Control, ENTER to continue</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Enter the name of the ALERT queue that will receive ALERTs for this object from CSM in the **Object Alert Queue** field.

3. In the **Alert Publish Mode** field, specify whether an ALERT for this object is published and how it is published to connected PATROL EM workstations that have subscribed to receive ALERTs through the General Message Exchange (GME).

   Possible values are as follows:

   - **REPLACE**: an ALERT replacement for the ALERT’s key and queue is sent to all MainView AutoOPERATOR PATROL EM Integration (MAPEMI) workstations that have subscribed to receive ALERTs from this MainView AutoOPERATOR.
     
     If an ALERT with that key and queue on a COMMAND/POST workstation already exists, it is deleted before writing the new ALERT with that key and queue.

   - **ADD**: an ALERT ADD is sent to all workstations that have subscribed to receive ALERTs from this MainView AutoOPERATOR.
     
     If an ALERT with that key and queue on a PATROL EM workstation already exists, it is not deleted before writing the new ALERT with that key and queue. ADD is the default.

   - **NO**: the ALERT is not written to the connected COMMAND/POST workstations even if they have subscribed to receive ALERTs.

4. Press **Enter** to continue to the next object definition panel, **Object Groups**.
Associating an object with a group

Use the Object Groups panel to associate an object with a particular object group.

Every CSM partner has its own object group. Any BBI-SS PAS running CSM can have only one group. When an object is a member of a group, CSM will start that object (if it is not already up) on a BBI-SS PAS when the group is active.

**Note**
The first time you use CSM, you must define a group before you can start defining objects. This section describes only the process of associating the object with one or more already defined groups.

**To associate an object with one or more already defined groups**

1. Enter the M line command next to a group name. The object will be associated with that group.

   **Figure 27: Object Groups panel**

<table>
<thead>
<tr>
<th>Group</th>
<th>Status</th>
<th>Objects</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KMZ1</td>
<td>Member</td>
<td>9</td>
<td>Group List for SYSD Again</td>
</tr>
<tr>
<td>KMZ2</td>
<td></td>
<td>23</td>
<td>Group List for SJSE</td>
</tr>
<tr>
<td>KMZ6</td>
<td>Member</td>
<td>4</td>
<td>Group List for SYSD</td>
</tr>
<tr>
<td>KMZ7</td>
<td></td>
<td>80</td>
<td>test group 7</td>
</tr>
</tbody>
</table>

   The **Status** field shows all of the groups that the object is associated with in the CSM-PLEX. The **Objects** field lists the number of object within each group.

2. Enter the R line command next to the group name to dissolve the association between an object and its group.

To see all the objects within a group, enter the List line command next to the group name. The following shows an example of all the objects in a group named KMZ2.

**Figure 28: Object Groups panel**

<table>
<thead>
<tr>
<th>Object</th>
<th>Command</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAOCSM02</td>
<td>END</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAOCSM08</td>
<td>END</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAOCSM10</td>
<td>END</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AAOCSM17</td>
<td>END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Enter END or ENTER to continue.
3 Press **Enter** to continue to the Object Groups panel.

**Moving an object between groups**

When an object is a member of a group, CSM starts that object (if it is not already UP) on a BBI-SS PAS when the group is active.

Therefore, an object (and its schedule) can be defined once and then if it runs on multiple CSM partners, you can associate the object, using the Member line command with a group on each of the partners. Each object can be a member of up to 20 CSM groups.

You can also define an object that is normally active on only one CSM partner and moveable to another BBI-SS PAS (in cases of emergency backups or if you need more system resources) with the Moveable line command. In this case, the object is defined as a member to one group where you want the object to be normally active and, as moveable to all groups where you would possibly move the object to. Once an object is defined as Moveable to any groups, this object can be a Member of only one group, and can be moveable to 19 groups.

**To make an object moveable to another system**

1. Enter the Moveable line command next to the group name to which you want to be able to move the object.

This example shows object AAOCSM01 is a member of group KMZ1 but moveable to KMZ2 and KMZ6.

**Figure 29: Object Groups panel: creating a moveable object**

<table>
<thead>
<tr>
<th>Primary Commands: Locate CMDSHOW CANCEL</th>
<th>TGT === KMZ1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LC CMDS --------- (R)emove (L)ist (M)ember (O)moveable</td>
<td></td>
</tr>
<tr>
<td>Enter END command to process and return to Detail Control, ENTER to continue</td>
<td></td>
</tr>
<tr>
<td>LC Group Status Objects Description</td>
<td></td>
</tr>
<tr>
<td>___ KMZ1 Member 9 Group List for SYSD Again</td>
<td></td>
</tr>
<tr>
<td>o__ KMZ2 23 Group List for SJSE</td>
<td></td>
</tr>
<tr>
<td>o__ KMZ6 4 Group List for SYSD</td>
<td></td>
</tr>
<tr>
<td>___ KMZ7 80 test group 7</td>
<td></td>
</tr>
</tbody>
</table>

An object can be made moveable to as many as 20 groups. Once an object is denoted as moveable, it can be the member of only 1 group.
When an object is made moveable to a group, the object does not automatically start when that group is active until you instruct CSM to move the object to that group’s control.

2 Press Enter to continue to the next object definition panel, Object Requisites.

Defining the objects parents

For this example, object AAOCSM01 does not have parent objects associated with it. The following figure shows the Object Requisites panels for AAOCSM01 where you can an object’s parents.

**Figure 30: Object Requisites for AAOCSM01 panel**

<table>
<thead>
<tr>
<th>Requisite</th>
<th>Group Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVS_______</td>
<td>KMZ1_______</td>
<td>THE BIG GIANT HEAD</td>
</tr>
</tbody>
</table>

To add a parent

1 Enter the ADD primary command. The Add Object Requisites panel is displayed.

**Figure 31: Add Object Requisites for AAOCSM01 panel**

2 Press Enter to continue to the next object definition panel, Client/Server Relationships.

Defining the objects servers

For this example, object AAOCSM01 does not have server objects associated with it.
The following figure shows the Client/Server Relationships for the AAOCSM01 panel where you can add servers to the object.

**Figure 32: Client/Server Relationships for AAOCSM01 panel**

| BMC Software -- Client/Server Relationships for AAOCSM01 - Add --- AutoOPERATOR |
| COMMAND ===> | SCROLL ===> CSR |
| TGT --- KMZ1 |

**Primary Commands:** ADD END CMDSHOW

LC CMDS --------- (E)dit server, (D)elete server

Enter END command to process and return to Detail Control, ENTER to continue

LC Server Group Description
Connect Command
Disconnect Command

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

**To add a server**

1. Enter the ADD primary command. The Add Client / Server Relationships panel is displayed.

**Figure 33: Add Client/Server Relationships for AAOCSM01 panel**

| BMC Software -- Client/Server Relationships for AAOCSM01 - Add --- AutoOPERATOR |
| COMMAND ===> | TGT --- KMZ1 |
| C | BMC Software Add Client / Server Relationships AutoOPERATOR |
| COMMAND ===> | TGT --- KMZ1 |
| P | COMMAND ===> |
| L | Serveriomanip  Group Name ===>
| E | Connect Command ===>
| L | Disconnect Command ===>

Press ENTER to continue processing
Press END(PF3) / CANcel to cancel request

2. Press **Enter** to continue to the next object definition panel, Object Scheduling.

   In this example, a schedule is not defined for object AAOCSM01.

3. Press **Enter** to return to the CSM Object Detail Control panel.

4. Press **PF3/END** to return to the CSM Global Overview panel.

**Figure 34: CSM Global Overview panel**

| BMC Software ------------------ Global Overview ------------------ MainView AutoOPERATOR |
| COMMAND ===> | SCROLL ===> CSR |
| Primary Commands: EXCeption | Group | SORT | SAVE | ADD | PROFILE |
| CALeendar | CANcel | UNsort | CONVERT | BUILD |
| LC CMDS ------- (S)tart | b(O)unce | rese(T) | depe(N)ents | (A)dd |
| (CAN)cels | (M)oVe | (R)peat | (B)rowse | s(Y)nc |
| L | Object | System | Actual | Desired | Ind | Type |
| (CAN)els | (D)elete | (C)Lients | (D)elete | (C)Lients | (D)elete |

---

84  MainView AutoOPERATOR Solutions Guide
Object AAOCSM01 is added as the child of MVS.

5 Repeat this procedure for all the new Normal objects that you want to define.

For the new objects to be added to the repository, you must issue the ADD primary command on the CSM panel.

After you issue the ADD command, the object is added. The SAVE command stores its definition in the repository. CSM begins controlling the object after the BUILD command is processed.

**Transient object definitions**

To define a Transient object, follow the same instructions as for defining a Normal object except for these differences:

- Specify **TRAN** (Transient) for the **Object Type** field instead of **NORM** (Normal) on the Object Detail Control panel.

- Do not specify a value in the **Stop Command/EXEC** field on the Command Specifications panel.

**Note**

When defining an object to CSM, once you specify the object type of Transient or Normal, you cannot change the specification. To change the specification, you must delete the object and define a new one.

**Grouping object definitions**

To define a Grouping object, follow the same instructions as for defining a Transient object.

The only difference is that you do not specify a start command in the **Start Command/EXEC** field on the Command Specifications panel. The absence of the start command for an object is what CSM uses to differentiate between Grouping objects and Transient objects.

**CSM object types comparisons**

The following summarizes the similarities and differences for the three object types.
Table 5: Comparison between three CSM object types

<table>
<thead>
<tr>
<th>Object type</th>
<th>Can be parents</th>
<th>Can have schedules</th>
<th>Can be clients or servers</th>
<th>Requires a CSM start command</th>
<th>Requires a CSM stop command</th>
<th>Can be manually started</th>
<th>Can be manually stopped</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Objects</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Transient Objects</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grouping Objects</td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
</tr>
</tbody>
</table>

Using the repeat line command to create objects

From the Global Overview panel, use the line command (R)epet to duplicate an existing object’s definition to define a new object with the same information.

You can use this line command when you want to define objects with similar attributes.

The only limitation is that no two objects can have the same name so you must define a new name for the new object.

To use the Repeat line command

1. Enter the R line command in the LC field next to the object that you want to duplicate on the Global Overview panel.

The Repeat Object Definition panel is displayed.

Figure 35: Repeat Object Definition panel
2 Enter a new unique name for the object in the New Object Name field.

**Figure 36: Add new object panel: repeat line command example 2**

3 Press *Enter* to process the changes on this panel. The Object Detail Control panel for the new object is displayed.

4 Press *Enter* to proceed through object definition panels and enter new information for the new object AAOCSM03.

5 After all the information for object AAOCSM03 is completed, the Object Detail Control panel is redisplayed. To complete defining the AAOCSM03 object, press PF3/END.

The Global Overview panel is redisplayed with the new object added.

### Modifying object definitions

It might be necessary to modify an object’s relationships; for example, making changes, additions, or deletions to its list of parents.

The following sections describe how to modify an object’s relationship to another object.

### Adding parents

In this example, parent MVS has three child objects: AAOCSM08, AAOCSM09 and AAOCSM10.

This example shows how to add AAOCSM10 to AAOCSM08’s list of parents.
To add parent AAOCSM10 to child object AAOCSM08

1. Enter the E line command next to the AAOCSM08 object on the Global Overview panel. The Object Detail Control panel is displayed.

Figure 37: Adding a parent

2. Enter the REQ primary command to skip the panels and display the Object Requisites panel.

3. Enter the ADD primary command to display the Add Object Requisites panel:

4. In the Requisite field, type the new object name, and in the Group Name field, type the group name of the parent to be added to CSM object AAOCSM0, as shown in the following figure.

5. Press PF3/END to process. The Object Detail Control panel is redisplayed.

To see a list of all the parents of an object, use the Browse line command on the Global Overview panel for a specific object (Object Detail Control panel is displayed). Then enter the REQ command; the following figure shows the result for object AAOCSM08.

You have added AAOCSM10 as a parent to object AAOCSM08.

7 To save this change, enter the SAVE primary command on the Global Overview panel.

Deleting parents

In this example, parent MVS has three children: AAOCSM08, AAOCSM09 and AAOCSM10.
AAOCSM10 is also a parent of AAOCSM08. This example shows how to delete AAOCSM10 from the list of parents for AAOCSM08.

**To delete object AAOCSM10 from child object AAOCSM08**

1. Enter the E line command next to AAOCSM08 on the Global Overview panel. The following figure is displayed and shows the two parents of AAOCSM08; MVS and AAOCSM10.

   **Figure 38: Deleting a parent: step 1**

2. Enter the D Object line command next to AAOCSM010. The line for AAOCSM10 disappears from the panel. See the following figure.

   **Figure 39: Deleting a parent: step 2**

3. Press PF3/END. The Object Detail Control panel is redisplayed.

4. Press PF/3END again. The Global Overview panel is redisplayed.

5. To save this change, issue the SAVE primary command on the Global Overview panel.

**Adding servers**

Along with modifying the parent-child relationships for an object, it might be necessary to make additions or deletions to an object’s list of servers.

A maximum of 45 servers can be entered for each object.
In this example, AAOCSM10 has no server objects assigned to it. AAOCSM09 functions in this system as a server. This example shows how to add AAOCSM09 to AAOCSM10’s list of servers.

To add server AAOCSM09 to client object AAOCSM10:

1. Enter the E line command next to AAOCSM10 on the Global Overview panel. The Object Detail Control panel for AAOCSM10 is displayed.

   **Figure 40: Adding a server**

   ![Figure 40](image_url)

   **BMC Software -------- Object Detail Control - Edit ------- MainView AutoOPERATOR**
   **COMMAND ===> TGT --- A06A**
   
   The following options are displayed in sequence, or may be selected by entering the three-character code CMD - Command Specifications        EV1 - Event Specifications 1
   EV2 - Event Specifications 2        NOT - User Notification
   GRP - Object Groups                 REQ - Object Prerequisites
   CLS - Client/Server Relationships   SCH - Object Scheduling
   
   Object Name ===> AAOCSM10 Member of Group ===> A06A
   
   Object Type ===> NORM Type of object (? for list)
   
   Verify Force Down ===> N (Y/N) Restart Only Control ===> N (Y/N)

   Application Information:
   
   Author ===> BAOKMZ1
   Description ===> Child of MVS

   Last Modified by BAOKMZ1 on 01/30/2001 at 17:36
   
   Enter END command to process and return, ENTER to continue, or CANcel

2. Enter the primary command CLS to display the Client/Server Relationships panel for AAOCSM10.

3. Enter the ADD primary command. The Add Client / Server Relationships panel is displayed.

4. Enter the Server name, the Group Name, and Connect and Disconnect commands for the new server.

   Prefix the commands with a pound sign (#) if the commands are to be issued through MVS.

   The following example shows the completed panel.

   **Figure 41: Adding a server - step 2**

   ![Figure 41](image_url)

   **BMC Software ---- Client/Server Relationships for AA01 - Add ---- AutoOPERATOR**
   **BMC Software Add Client / Server Relationships AutoOPERATOR**
   **COMMAND ===> TGT --- A06A**
   
   Server ===> AAOCSM09
   Group Name ===> A06A
   
   Connect Command ===> #F AAOCSM09,Connect AAOCSM10
   Disconnect Command ===> #F AAOCSM09,Disconnect AAOCSM10

   Press ENTER to continue processing
   Press END(PF3) / CANcel to cancel request

   * *************************************************************************************************************************************
5 Press PF3/END. The Object Detail Control panel is redisplayed.

6 Press PF3/END again. The Global Overview panel is redisplayed.

7 To view the servers for object AAOCSM10, enter the line command L next to AAOCSM10. The following figure is displayed.

Figure 42: Clients of AAOCSM10 panel

8 Press PF3/END to return to the Global Overview panel.

9 To save this change, issue the SAVE primary command on the Global Overview panel.

Deleting servers

In this example, AAOCSM10 has the server object AAOCSM09 assigned to it.

This example shows how to remove AAOCSM09 from AAOCSM10’s list of servers.

To delete server AAOCSM09 from object AAOCSM10

1 Enter the E line command next to AAOCSM09 on the Global Overview panel. The Object Detail Control panel for AAOCSM09 is displayed.

Figure 43: Global Overview panel: deleting a server

Figure 44: Deleting a server - step 2
2 Enter the primary command `CLS` to display the Client/Server Relationships panel for AAOCSM09.

The following figure is displayed and shows that AAOCSM09 is the only server assigned to AAOCSM10.

**Figure 45: Deleting a server - step 3**

3 Enter the D line command next to AAOCSM10. The line for AAOCSM10 disappears from the panel.

4 Press `PF3/END`. The Object Detail Control panel is redisplayed.

5 To save this change, issue the `SAVE` primary command on the Object Detail Control panel panel.

6 Press `PF3/END` again. The Global Overview panel is displayed.

**Figure 46: Global Overview panel: server deleted**
7 Enter the C line command next to AAOCSM10 to view the servers for object AAOCSM10. The following figure is displayed.

**Figure 47: Clients of AAOCSM10 panel**

BMC Software    -------- Clients of AAOCSM10  ------- MainView AutoOPERATOR
COMMAND ===>                                                 TGT --- A06A
***************************************************************************

8 Press **PF3/END** to return to the Global Overview panel.

9 To save this change, issue the SAVE primary command on the Global Overview panel.

**Modifying routine schedules**

The following procedure describes how to schedule object AAOCSM07 to be unavailable (or DOWN) on:

- Wednesdays from 3:00 P.M. to 7:00 P.M.
- Fridays from 2:00 A.M. to 3:00 A.M.
- Fridays from 6:00 A.M. to 9:00 A.M.
- Sundays from 3:00 P.M. to 6:00 P.M.

Because an object cannot be available when its parent is unavailable, your schedule for the object also must reflect the down times of its parents.

**Determining the schedule of the objects parent**

When you want to schedule the unavailability of an object, begin by determining the unavailability of its parent.

The dependent object must be down when the parent is down.

In this case, AAOCSM07 is a child of AAOCSM10.

**To determine the schedule for AAOCSM10, follow these steps:**

1 Enter the B line command next to the object AAOCSM10 on the Global Overview panel. The Object Detail Control - Browse panel is displayed.

2 Enter the SCH primary command to display the Object Scheduling panel for AAOCSM10.
The following figure shows that AAOCSM10 is scheduled to be DOWN on Mondays from 3:00 P.M. to 5:00 P.M.

**Figure 48: Checking the schedule of a parent: step 2**

3 Make a note of this time and press **PF3/END** to return to the Object Detail Control panel.

4 Press **PF3/END** again to return to the Global Overview panel.

5 See the procedure for "Modifying the object’s schedule" for more information about creating a schedule for AAOCSM07.

### Modifying the objects schedule

After checking the schedule, you know that the parent of AAOCSM07 is DOWN on Mondays from 3:00 P.M. to 5:00 P.M. You also want to schedule AAOCSM07 to be DOWN at the following times:

- Wednesdays from 3:00 P.M. to 7:00 P.M.
- Fridays from 2:00 A.M. to 3:00 A.M.
- Fridays from 6:00 A.M. to 9:00 A.M.
- Sundays from 3:00 P.M. to 6:00 P.M.

To avoid scheduling conflicts with the parent (AAOCSM10), you must also include Monday 3:00 P.M. to 5:00 P.M. in this list. If you do not do this step, you will encounter a scheduling conflict. For information about scheduling conflicts, refer to *CR**"Resolving scheduling conflicts at modification" on page 111.

**To modify AAOCSM07’s schedule, follow these steps:**

1 From the CSM Global Overview panel, enter the E line command next to AAOCSM07. The Object Detail Control panel for AAOCSM07 is displayed.
2 Enter the primary command SCH to display the Object Scheduling panel for AAOCSM07.

Figure 49: Modifying an object schedule

3 Enter the ADD primary command. The Add Object Scheduling panel is displayed.

Figure 50: Add Object Scheduling panel

4 Use 24-hour clock format to enter all times. Repeat until all DOWN times are added.

The following figure shows the Object scheduling panel with all the DOWN times filled in.

Note that the DOWN time for parent object AAOCSM10 is also included. (Monday from 3:00 P.M. to 5:00 P.M.)

Figure 51: Added Object DOWN times
AAOCSM07 will be DOWN:

- Fridays from 2:00 A.M. to 3:00 A.M.
- Fridays from 6:00 A.M. to 9:00 A.M.
- Mondays from 3:00 P.M. to 5:00 P.M. (because AAOCSM07’s parent, AAOCSM10 is down at that time)
- Sundays from 3:00 P.M. to 6:00 P.M.
- Wednesdays from 3:00 P.M. to 7:00 P.M.

5 Press PF3/END. The Object Detail Control panel is redisplayed.

6 Press PF/3END again. The Global Overview panel is redisplayed.

7 To save the changes, enter the SAVE primary command.

Changes take effect when you enter the Build primary command for the group.

Modifying an objects schedule with CSM schedule tags

CSM includes three predefined schedules that you can use for scheduling objects:

- WEEKDAY (WKD): specifies that the object should be DOWN at a specified time on Mondays through Fridays every week
- WEEKEND (WKN): specifies that the object should be DOWN at a specified time on every Saturday and Sunday
- DAILY (DLY): specifies that an object should be DOWN at a specified time every day of the week (Monday through Sunday) every week

These tags also can be used in conjunction with schedules for a specific day and time. CSM will take the object down on the specific day in addition to the time specified by any of these tags.

For example, suppose object AAOCSM09 should be DOWN every day from 2:00 A.M. to 3:00 A.M.

**To modify AAOCSM09’s schedule**

1 Enter the E line command next to AAOCSM09. The Object Detail Control panel for AAOCSM09 is displayed.
2 Enter the primary command SCH to display the Object Scheduling panel for AAOCSM09.

**Figure 52: Object Scheduling panel**

```
BMC Software --- Object Scheduling for AAOCSM09 - Edit --- MainView AutoOPERATOR
COMMAND ===> SCROLL ===> CSR
TGT --- A06A
Primary Commands: ADD CANcel END CMDSHOW
LC CMDS --------- (E)dit Time, (D)elete Time
Enter END to process/return to Detail Control, ENTER to continue, or CANcel
LC Object Group Day Down From Down To
Primary Commands: ADD CANcel END CMDSHOW
TGT --- A06A

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

3 Enter the primary command ADD. The Add Object Scheduling panel is displayed.

**Figure 53: Add Object Scheduling panel**

```
BMC Software --- Object Scheduling for AAOCSM09 - Edit --- MainView AutoOPERATOR
COMMAND ===> _________________________________________ TGT --- A06A
DATE --- 01/01/30
TIME --- 19:27:12

Day Selections: MON, TUE, WED, THU, FRI, SAT, SUN or WKD, WKN, DLY
Group Name        ===> ________
Day               ===> _________
Down From (HH:MM) ===> __ : __
Down To   (HH:MM) ===> __ : __

Press ENTER to continue processing
Press END(PF3) / CANcel to cancel request
```

4 Enter the word DLY to specify that this object is supposed to be DOWN every day (Monday through Sunday), and enter the times using 24-hour clock format (02:00 and 03:00).

**Figure 54: Example of times entered on the Add Object Scheduling panel**

```
BMC Software --- Object Scheduling for AAOCSM09 - Edit --- MainView AutoOPERATOR
COMMAND ===> _________________________________________ TGT --- A06A
DATE --- 01/01/30
TIME --- 19:27:12

Day Selections: MON, TUE, WED, THU, FRI, SAT, SUN or WKD, WKN, DLY
Group Name        ===> A06A____
Day               ===> DLY______
Down From (HH:MM) ===> 02 : 00
Down To   (HH:MM) ===> 03 : 00

Press ENTER to continue processing
Press END(PF3) / CANcel to cancel request
```

5 You can enter additional DOWNtimes by using the ADD primary command. Press PF3/END to redisplay the Object Scheduling panel.
6 Press **PF3/END**. The Object Detail Control panel is redisplayed.

7 Press **PF3/END** again. The Global Overview panel is redisplayed.

8 To save the changes, enter the **SAVE** primary command.

9 Enter the primary command **BU** on the **COMMAND** line.

The changes take effect when the **BUILD** command is issued for the group that this object is associated with. **CSM** issues an error message if it detects a schedule conflict. Note that it is not a conflict to have two identical schedule entries, each for a different group.

---

**Resolving scheduling conflicts at modification**

If an object is scheduled to be **UP** when its parent is **DOWN**, **CSM** issues a short error message in the upper right corner when you try to exit from the scheduling panel.

In this example, you have completed the schedule panel for **AAOCSM07**, as shown in the following figure, where the object will be **DOWN** on Wednesdays, Fridays, and Sundays.

**Figure 55: Schedule conflicts: view the schedule**

<table>
<thead>
<tr>
<th>Group</th>
<th>Day</th>
<th>Down From</th>
<th>Down To</th>
</tr>
</thead>
<tbody>
<tr>
<td>A06A</td>
<td><strong>FRIDAY</strong></td>
<td>02:00</td>
<td>03:00</td>
</tr>
<tr>
<td>A06A</td>
<td><strong>FRIDAY</strong></td>
<td>06:00</td>
<td>09:00</td>
</tr>
<tr>
<td>A06A</td>
<td><strong>SUNDAY</strong></td>
<td>15:00</td>
<td>19:00</td>
</tr>
<tr>
<td>A06A</td>
<td><strong>WEDNESDAY</strong></td>
<td>15:00</td>
<td>19:00</td>
</tr>
</tbody>
</table>

---

The **DOWN** time of Mondays from 3:00 P.M. to 5:00 P.M for **AAOCSM07**’s parent, **AAOCSM10**, is not added to the schedule.

**To view the schedule conflict error message**

1 Press **PF3/END** to save this schedule; you will see a short error message as shown in the upper right corner of the following figure.

**Figure 56: Schedule conflicts: error occurs**
2 Press **PF1** to see the entire error message.

The conflict is described with more information in the upper half of the following figure.

**Figure 57: Schedule conflicts: complete error message**

3 To leave this panel, you must delete all the entries that you just entered for AAOCSM07.

4 Press **PF3/END** to return to the CSM panel.

5 To successfully enter the schedule for AAOCSM07, you must perform the procedure described in the procedure "Determining the schedule of the object’s parent," note all the times the parent AAOCSM10 is DOWN, and then perform the steps in "Modifying the object’s schedule".

As a general practice, to avoid these types of conflicts, BMC recommends that you always schedule the children first and then define schedules for the parents.

### Overriding the calendar for special situations

The Global Calendar Override operates under the control of a specific calendar date.

The override is only for the date specified on this panel; therefore, the affected objects do not have to be restored to their normal schedules after the override takes place.
The following procedure describes how to override the Calendar to schedule object AAOCSM07 to be unavailable (or DOWN) on

03/01/97 from 06:00 P.M. to 12:00 A.M.

To access the Global Calendar Override panel

1. Issue the CAL primary command from the Global Overview panel. The following figure is displayed.

   **Figure 58: Global Calendar Override panel**

2. Enter the following required information:

   ■ Object’s name (Name field)

   ■ Group name

   ■ Calendar date (Calendar field)

   ■ Down times (From and To fields)

You also can specify the reason for the downtime and the contact name of the person who asked for it.

If you do not fill in the required information, CSM prompts you in the upper right corner when you press PF3/END.

The following figure shows an example of a Global Calendar Override panel that is filled in.

**Figure 59: Example of global calendar override for object AAOCSM07**
Enabling cross-system object management

To perform cross-system object management, ensure that BBI-SS PAS to BBI-SS PAS communications is active between BBI-SS PASs where CSM is installed and operating.

The following procedure describes how to enable cross-system object monitoring and management.

To enable cross-system monitoring, you must edit the object

1. Enter E in the line command column next to MVS on the Global Overview panel. The Object Detail Control - Edit panel for the MVS object is displayed.

   Figure 60: Object Detail Control - Edit panel for MVS

   

   The Object Detail Control - Edit panel for the MVS panel includes the field **Enable PLEX ==> Y**, where the default setting is YES.

2. To enable cross-system management, ensure that Y is entered in this field.

3. Press PF3/END.

4. To complete enabling cross-system management, you must repeat this process on the other BBI-SS PAS targets.
Performing routine object tasks

The following section describes routine tasks you can perform on CSM objects from the Global Overview panel.

Starting an object

The purpose of CSM is to help manage the availability of objects in your system, which is achieved by performing automatic starts and stops of objects, based on a defined schedule.

Sometimes, however, it is necessary to start an object outside of its schedule.

To start an object from the Global Overview panel when the object’s Actual and Desired states are DOWN

1 Enter the S line command next to AAOCSM07 on the Display the Global Overview panel. The following figure is displayed.

   Figure 61: Object start confirmation panel

<table>
<thead>
<tr>
<th>BMC Software</th>
<th>Global Overview</th>
<th>MainView</th>
<th>AutoOPERATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Confirm: Start for AAOCSM07</td>
<td>TGT --- A06A</td>
<td></td>
</tr>
<tr>
<td>Please confirm START for AAOCSM07</td>
<td>? N (Y/N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>START all PARENTS of AAOCSM07</td>
<td>? Y (Y/N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>START now, but STOP as scheduled</td>
<td>? Y (Y/N)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   Please press ENTER to continue or END / CANcel to cancel request

2 Reply to the start confirmation panel; the following list describes the replies that you can enter on this panel:

- To start AAOCSM07, enter Y for Please confirm START for AAOCSM07.
  If you accept the default setting of N, AAOCSM07 will not start.

- To start all the parents of object AAOCSM07, accept the default setting of Y for START all PARENTS of AAOCSM07.
  If you enter N, CSM attempts to start only AAOCSM07.
To start the object and accept the predefined stop time for AAOCSM07, accept the default setting of **Y** for **START now, but STOP as scheduled**. CSM starts the object now.

After the object is started, CSM stops the object at the scheduled time. The object remains under the control of CSM and has an actual state of UP and a desired state of UPEARLY.

If you enter **N**, CSM starts the object, ignores the schedule, and does not trigger a shutdown at the scheduled time. The object will show an actual state of UP and a desired state of UPFORCE.

3 Press **Enter**. The following figure is displayed.

**Figure 62: Global Overview panel: object started**

The Global Overview now shows that the actual state of AAOCSM07 is **STARTING** and the desired state is **UPEARLY**. Once the object starts, the actual state is **UP**.

---

**Tip**

On rare occasions, CSM issues ALERT ACM720A indicating that the start command has timed out for an object that has just started. The SYSLOG shows that the object indeed started. The ALERT, however, remains. The object’s state variable contains the value **STARTING**. Use the **Y** line command to synchronize the object’s state variable with the actual state of the object.

If the parent objects of the affected object are not started by CSM, it is usually due to the object’s Start Rule not firing. Make sure the Start Rule has fired. If it has not, you must determine why and correct the problem for the next startup of this object.

---

**Stopping an object**

For many of the same reasons for unexpectedly starting an object, an object also might have to be stopped from the CSM Global Overview.
**To stop an object**

1. Enter the P line command next to AAOCSM09 on the Global Overview panel. The following figure is displayed.

   **Figure 63: Object stop confirmation panel**

   Please confirm STOP for AAOCSM09  ? N (Y/N)
   STOP all DEPENDENTS of AAOCSM09  ? Y (Y/N)
   STOP now, but START as scheduled  ? N (Y/N)

   Press ENTER to continue or END / CANCEL to cancel request

2. Reply to the stop confirmation panel; the following list describes the replies that you can enter on this panel:

   - To stop AAOCSM09, enter **Y** for *Please confirm STOP for AAOCSM09*.
     If you accept the default setting of **N**, the object is not stopped.

   - To stop all dependents of object AAOCSM09, accept the default setting of **Y** for *STOP all DEPENDENTS of AAOCSM09*. CSM stops all dependent objects of AAOCSM09 before stopping AAOCSM09.
     If you enter **N**, CSM stops only AAOCSM09.

   **Note**

   The default setting for *STOP all DEPENDENTS* is **Y**. You can change the default to **N** (for NO) by adding a SHARED variable named CSMRELATIVE and set its value to **N**.

   - To stop the object and ignore the scheduled start time for AAOCSM09, accept the default setting of **N** for *STOP now, but START as scheduled*.
     CSM stops the object now but does not restart at the scheduled restart time. The object is now outside of the control of CSM and has an actual state of DOWN and a desired state of DOWNFORCE.
     If you enter **Y**, CSM stops the object, maintains control, and triggers restart at the scheduled time. The actual state of AAOCSM09 will be DOWN and the desired state will be DOWNEARLY.
3 Press Enter. The following figure is displayed.

**Figure 64: Global Overview panel: object stopped**

The Global Overview now shows that the actual state of AAOCSM09 is STOPPING and the desired state is DOWNFORCE. As soon as the object stops, the actual state will be DOWN.

**Tip**

On rare occasions, CSM issues ALERT ACM725A, indicating that the stop command has timed out for an object that has just stopped. A check of the SYSLOG shows that the object indeed stopped. The ALERT, however, remains. The object’s state variable contains the value STOPPING.

If children objects of the affected object are not stopped by CSM, it is usually due to the object’s Stop Rule not firing. Make sure the Stop Rule has fired. If it has not, you must determine why and correct the problem for the next shutdown of this object.

---

**Bouncing an object**

Occasionally, an object might need to be recycled (stopped then immediately started).

CSM simplifies this task with the Bounce function.

The Bounce line command causes a controlled, unscheduled termination of the object by triggering the object’s stop command. During this stoppage, the object’s desired state never changes from UP which causes CSM to restart the object immediately after the object’s Stop Rule is fired.
To bounce an object (AAOCSM10)

1. Enter the O line command next to AAOCSM10 on the Global Overview panel. The following figure is displayed.

   **Figure 65: Confirm bounce panel**

2. Reply to the bounce confirmation panel; the following list describes the replies that you can enter on this panel:

   - To bounce AAOCSM10, enter `Y`. CSM bounces AAOCSM10.

   - To cancel the bounce, accept the default setting of `N` for **Please confirm BOUNCE for AAOCSM10**. The object bounce request is rejected.

3. Press **Enter**. The following panel is displayed.

   **Figure 66: Global Overview panel: object bounce**

   The actual state of the object is DOWN. When the object restarts after being DOWN, the actual state will appear as UP.
Tip
If an ALERT Detail panel is active while a Bounce is being performed, you might see ALERT ACM755A issued by CSMDOWN on behalf of the object being bounced. The ALERT disappears a second or two later. This behavior is normal and expected. The ALERT is being issued because an object is coming down at an unexpected time. Remember that the Bounce command performs a controlled, unscheduled termination of the object.

Resetting an object

Occasionally, objects need to be reset back to schedule control when the object is started or stopped at times other than scheduled times or when the object was removed from schedule control altogether.

At the same time, it might be useful to reset the object’s command limit count to zero.

The following procedure describes how to reset an object to return it to the control of its schedule.

To reset object AAOCSM10

1. Enter the T line command next to AAOCSM10 on the Global Overview panel. The Confirm: Return object to Schedule confirmation panel is displayed.

Figure 67: Confirm resetting an object panel

2. Reply to the reset confirmation panel; the following list describes the replies that you can enter on this panel:

- To reset AAOCSM10, enter Y for Please confirm RESET for AAOCSM10. AAOCSM10 is reset back to schedule control.

If you accept the default setting of N, the object reset request is rejected.
To prevent resetting the object’s command count to zero, accept the default setting of N for **RESET command count for AAOCSM10**.

If you enter Y, CSM resets the command count to zero.

---

**Tip**

The object command count is incremented each time an object is started during the day, from 00:00 to 23:59.

If the command count reaches or exceeds the defined start command limit for the object, ALERT ACM750A is issued and the object start is interrupted.

The command count is reset to zero in one of three ways:

- The object stops and CSMDOWN executes on the behalf of the object. The time-initiated Rule ACMRC060, supplied by BMC Software, fires at 00:00:01 and schedules EXEC ACMM800.

- The command count is set to zero by way of the reset line command.

- For the **Scope of RESET of AAOCSM10** field, you can enter the following replies:
  
  - S: CSM restores to schedule control and, if requested, resets the command count of only the selected object. This setting is the default.
  
  - P: CSM restores the schedule control and, if requested, resets the command count of the selected object and its defined parents.
  
  - C: CSM restores the schedule control and, if requested, resets the command count of the selected object and that object’s children.

3 Press **Enter**. The Global Overview panel is displayed.

The object is returned to scheduled control. CSM either starts or stops the object as you defined it.

---

**Tip**

All the objects defined to CSM can be restored to schedule control with the use of a single line command. Issue the T line command for the MVS object.

By indicating a scope of C, CSM returns all of the children of MVS to schedule control. Every object in the CSM database is a child of MVS by default.

The command count for every object in the CSM database also can be reset to zero with one command by using the same procedure.
**Canceling an object**

In a critical situation, an object might be in a condition where it must be stopped quickly and a normal shutdown would take too long.

The object can even be in a condition that makes a normal shutdown impossible. In either case, the object must be terminated quickly.

Issuing a cancel command for the object through CSM terminates the object quickly and instructs CSM not to restart the object immediately.

Note that the cancel command does not affect the parents or the children of the selected object. Only the selected object is canceled.

**To cancel an object**

1. Enter the C line command next to AAOCSM10 on the Global Overview panel. The following figure is displayed.

   **Figure 68: Confirm cancel panel**

   

   BMC Software ------------- Global Overview ------------ MainView AutoOPERATOR
   BMC Software                          Confirm: Cancel for AAOCSM10     AutoOPERATOR
   COMMAND ===>                                          TGT --- AO6A     |  A
   |                                                                        |
   |                                                                        |
   |                                                                        |
   |  Please confirm Cancel for AAOCSM10    ? N (Y/N)                       |
   L |                                                                        |
   L |                                                                        |
   C |                                                                        |
   _ |  Press ENTER to continue or END / CANcel to cancel request             |
   _ |________________________________________________________________________|
   ******************************* Bottom of data ********************************

2. Reply to the cancel confirmation panel:

   - To cancel the object, enter Y for **Please confirm CANCEL for AAOCSM10**. The cancel for object AAOCSM10 is completed.
   - If you choose not to continue with canceling AAOCSM10, accept the default of N and the object cancel request is rejected.

3. Press **Enter**. The Global Overview panel is displayed.

   **Figure 69: Global Overview panel: object canceled**
The object is canceled and the actual state shows DOWN and the desired state shows DOWNFORCE. CSM does not attempt to start the object until you issue either a S or a T line command for the object.

## Moving an object with the moveable attribute

Occasionally you might want to move an object from the local BBI-SS PAS to a remote BBI-SS PAS.

Moving objects between systems is accomplished by making an object moveable.

For example, you might want to perform maintenance on one system, but you want to move an application running on that BBI-SS PAS to another BBI-SS PAS so it can complete running its jobs.

---

**Note**

An object can be moved from either the Global Overview panel or CSM Exceptions panel. This example shows moving an object from the Global Overview panel.

---

Moving an object requires that the following conditions be in effect:

- The destination BBI-SS PAS has CSM installed and operating.
- BBI-SS PAS to BBI-SS PAS communications between the two subsystems is active.
- The object is associated with the active group on the destination BBI-SS PAS or has been defined on the destination BBI-SS PAS if the two CSMs do not share a repository.

---

**Note**

While an object can be in any state when it is moved, the actual state of the object on the moved to the destination is UP; however, if you are moving objects between CSMs that do not share a repository, the preferred state is UPFORCE.

---

When you want to move an object, the following considerations apply:
Any object that is a candidate to be moved must be associated with the group running on the BBI-PAS that will control the object (usually on another MVS image).

Objects can be moved from a group with the status of *inactive or unknown* to an active group if both groups are defined in the same repository. The object moved from the inactive or unknown group will not start in that group when it becomes active. It will remain in the group it was moved to until it is moved to another group.

The object must be defined to all BBI-SS PASs with exactly the same information except for the schedule.

On the BBI-SS PAS that an object is moved to, the object can have its own schedule definitions.

The object will not start on a BBI-SS PAS until it is moved there.

When an object is moveable, it can be a member of only one group but can be moveable to up to 19 groups.

If you are moving the object and its dependents, the objects will be stopped and re-started in dependency order.

For the object and its dependents to be moved correctly, the objects need to stop and start in the same order. To ensure that this happens, moveable objects should have the same dependency structure in every group they are associated with.

---

**Note**

You cannot move an object to a BBI-SS PAS that is not defined or does not have CSM running.

Once the object is defined, you can move the object from one BBI-SS PAS to another. This example shows moving an object (MAMKZ111) from KMZ2 to KMZ1.

Before moving an object from a group with a status of inactive or unknown, verify that the object is not active on any other group.

The M line command can be issued against any group with a status of inactive or unknown and CSM will unconditionally move the management of the object from the inactive or unknown group to the group that you specify.

**To move an object**

1. Enter the M line command next to the object MAMKZ111 on the Global Overview panel.
Note: Use caution when moving an object from an inactive or unknown group because an object might be running multiple times in your systems. If you move an object under this condition, it could lead to unpredictable results for that object.

The Confirm Move panel is displayed.

**Figure 70: Confirm Move panel**

<table>
<thead>
<tr>
<th>BMC Software</th>
<th>Confirm: Move for MAMKZ111</th>
<th>AutoOPERATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND ====&gt;</td>
<td>TGT --- KMZ2</td>
<td></td>
</tr>
</tbody>
</table>

Please confirm MOVE for MAMKZ111? N (Y/N)
MOVE dependents of MAMKZ111? N (Y/N)
Group id to MOVE MAMKZ111 to? XXXXXXXX

Press ENTER to continue or END / CANcel to cancel request

2. To reply to the move confirmation panel:
   - Enter Y
   - Enter the SSID of the destination BBI-SS PAS. In this case, the destination BBI-SS PAS ID is KMZ1.

3. Indicate whether you want to move the selected object (the default) or move the selected object and its dependents by setting the value in the MOVE dependents of <object_name> field.

4. Press Enter. The Global Overview panel is redisplayed.

The MAMKZ111 object still appears on the CSM Global Overview panel. In addition, the actual state has been updated to DOWN and the desired state to MOVED on the system that it moved from. The actual state has been updated to UP and the desired state to UP on the system that it moved to.

**Figure 71: Global Overview panel: moving an object**
Scheduling an EXEC associated with a CSM ALERT

The **Ind** column of the Global Overview shows the following information about an object:

- **L** indicates that the object has clients.
- **N** indicates that the object has children (dependents).
- **X** indicates that the object has an outstanding ALERT associated with it.

To view the ALERT and its contents, you must use the ALERT Management Facility.

- **E** indicates that the outstanding ALERT associated with the object has a follow-up EXEC.

You can schedule the follow-up EXEC by entering the X line command next to the object in the **LC** column.

Some CSM objects might have an outstanding ALERT that does not have an EXEC associated with it. In this case, the **Ind** column displays an X. When the ALERT associated with a CSM object is also associated with an EXEC, the **Ind** column displays both an X and an E.

Occasionally, you will want to schedule the EXEC that is associated with a CSM ALERT from the CSM Global Overview. The following steps describe this procedure.
To schedule an EXEC associated with a CSM ALERT

1. Check the **Ind** column to ensure that the object has an X and an E displayed.

   In this example, the only object with a CSM ALERT and EXEC associated with it is JESA.

2. Enter the EX line command in the **LC** column next to AAOCSM09.

3. Press **Enter**. The EXEC is scheduled.

Object states

You can view both actual and desired states of all objects defined to CSM either

- with the Global Overview panel
- with the CSM Exceptions panel

or

- with two EXECs, CSMACT and CSMINFO, that enable you to manually track and change the different states of an object between the normal UP or DOWN states

You can use these EXECs when you need to override an object’s predefined schedule and still keep the object within CSM control.

Object desired states

An object’s desired state is the state of the object according to how you scheduled the object, or its desired state can be the result of a command that you issued against the object.

You can issue commands (such as start, stop, reset) against an object to change the desired state either through the CSM dialogs or with the EXEC CSMACT.

Use this table to figure out how an object arrived at its desired state:

- The first column lists possible desired states.
- The second column provides a brief description of the desired state.
- The third column shows what command was issued against the object.
- The fourth column shows the original state of the object before the command was issued against the object.

For example, from the first row of the table, you can see that if you execute any of following the commands in the table against an object that is scheduled to be UP the resulting desired state of the object would be UP:

- bounce
- reset
- resetc
- up

Table 6: Possible desired states an object can have

<table>
<thead>
<tr>
<th>Resulting desired state</th>
<th>Description</th>
<th>Command that was issued</th>
<th>Original desired object state</th>
</tr>
</thead>
<tbody>
<tr>
<td>UP</td>
<td>An object that CSM has started according to its schedule.</td>
<td>BOUNCE</td>
<td>UP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESET</td>
<td>Any state of an object that is scheduled to be up</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESETC</td>
<td>Any state of an object that is scheduled to be UP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UP</td>
<td>UP</td>
</tr>
<tr>
<td>UPEARLY</td>
<td>Intermediate state of UP: usually the result of bringing the object UP before its scheduled UP time.</td>
<td>BOUNCE</td>
<td>UPEARLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPEARLY*</td>
<td>Any state</td>
</tr>
<tr>
<td>UPFORCE</td>
<td>Intermediate state of UP: result of bringing the object UP outside of the object’s schedule, before its scheduled UP time.</td>
<td>BOUNCE</td>
<td>UPFORCE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UP</td>
<td>Any state</td>
</tr>
<tr>
<td>UPLOGICAL</td>
<td>Result of bringing a grouping object UP according to its scheduled UP time.</td>
<td>BOUNCE</td>
<td>Uplogical</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPLOGICAL</td>
<td>Any state</td>
</tr>
<tr>
<td>UPLOGICALE</td>
<td>Intermediate state of UP for grouping objects: result of bringing the object UP before its scheduled UP time.</td>
<td>BOUNCE</td>
<td>UPLOGICALE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPLOGICALE</td>
<td>Any state</td>
</tr>
</tbody>
</table>
### Resulting desired state

<table>
<thead>
<tr>
<th>Description</th>
<th>Command that was issued</th>
<th>Original desired object state</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOWN</td>
<td>RESET</td>
<td>Any state of an object that is scheduled to be DOWN</td>
</tr>
<tr>
<td></td>
<td>RESETC</td>
<td>Any state of an object that is scheduled to be DOWN</td>
</tr>
<tr>
<td>DOWNEARLY</td>
<td>DOWNEARLY*</td>
<td>Any state</td>
</tr>
<tr>
<td>DOWNFORCE</td>
<td>DOWN</td>
<td>Any state except DOWN</td>
</tr>
<tr>
<td></td>
<td>CANCEL</td>
<td>Any state</td>
</tr>
<tr>
<td>DOWNLOGICAL</td>
<td>DOWNLOGICAL</td>
<td>Any state</td>
</tr>
<tr>
<td>DOWNLOGICALE</td>
<td>DOWNLOGICALE</td>
<td>Any state</td>
</tr>
<tr>
<td>COMPLETE</td>
<td>BOUNCE</td>
<td>COMPLETE</td>
</tr>
<tr>
<td></td>
<td>UPLOGICAL</td>
<td>Any state</td>
</tr>
<tr>
<td></td>
<td>UPLOGICALE</td>
<td>Any state</td>
</tr>
<tr>
<td>MOVED</td>
<td>MOVE</td>
<td>Any State</td>
</tr>
<tr>
<td>MOVEDEP</td>
<td>MOVE</td>
<td>Any State</td>
</tr>
</tbody>
</table>

**Note:** The commands UPEARLY and DOWNEARLY cannot be issued on objects that have been defined as Restart Only. When an object is defined as Restart Only, CSM waits for the scheduling product to start the object.

### Object actual states

The following table shows the possible actual (or current) states that an object can have.
The actual state of an object is the current state of the object. You can use this table to figure out how an object arrived at its actual state:

- The first column lists the name of the actual state.
- The second column describes the state.

Table 7: Possible actual states

<table>
<thead>
<tr>
<th>actual state name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UP</td>
<td>object has started successfully</td>
</tr>
<tr>
<td>STARTING</td>
<td>object is in an intermediate state of UP: the result of starting the object but CSM has not yet received the start message for the object. If an object’s actual state stays in STARTING, the object has probably encountered an error.</td>
</tr>
<tr>
<td>UPMANUAL</td>
<td>object has been brought UP (from a desired state of DOWN) outside of CSM control</td>
</tr>
<tr>
<td>DOWN</td>
<td>object has been stopped successfully</td>
</tr>
<tr>
<td>CANCELED</td>
<td>cancel command was issued but CSM has not yet received the stop message for the object</td>
</tr>
<tr>
<td>ABENDED</td>
<td>object has abended. This state indicates that an abnormal termination event has occurred for an object and the object is supposed to be UP and no longer is UP. The object must be RESE(T) to restore it to CSM control.</td>
</tr>
<tr>
<td>COMPLETE</td>
<td>object has been successfully started by CSM according to its schedule</td>
</tr>
</tbody>
</table>

Manual CSM object state monitoring

CSM provides EXECs that you can use to manually monitor the state of CSM objects:

- CSMINFO
  enables you to see the actual and desired states of objects

- CSMACT
  enables you to start or stop CSM objects without using the CSM dialog panels

CSMINFO EXEC

The CSMINFO EXEC displays the actual and desired states of all CSM objects through a series of ALERTs.
These ALERTs are also published to PATROL Enterprise Manager (PATROL EM). But the alerts are usually not useful to the PATROL EM users. Therefore, by using the PUBLISH parameter, you can specify whether the ALERTs should be published. The complete syntax of the CSMINFO EXEC is

```plaintext
%CSMINFO [PUBLISH()]
```

If the PUBLISH parameter is omitted, the shared variable CSMINFO_PUBLISH_DEFAULT is used. If this shared variable is not set, the system default for PUBLISH is used. The system default is usually PUBLISH(NO), but it can be changed in AAOPRM00.

The shared variable CSMINFO_PUBLISH_DEFAULT is used to establish the default for the CSMINFO PUBLISH, and it can be set in the UBBPARM(BBIVAR).

**Example 1**

To see the actual and desired states of the objects defined to CSM, enter

```plaintext
%CSMINFO PUBLISH(ADD)
```

on the COMMAND line on the ALERT Detail panel or from any MainView AutoOPERATOR application to display the current (actual) and desired states of all objects defined to CSM.

**Example 2**

To see the state of a particular object, enter

```plaintext
%CSMINFO object_name PUBLISH(ADD)
```

For more information on PUBLISH parameters, see the MainView AutoOPERATOR Advanced Automation Guide.

**CSMACT EXEC**

Under ordinary circumstances, CSM should control the objects according to their schedules.

The CSMACT EXEC enables you to change an object’s state without using the CSM dialog panels. For example, you might use the CSMACT EXEC to modify an object if TSO/E and ISPF are not active.

When you use CSMACT, the object is still under CSM control and CSM can still track the state of the object. CSMACT can be scheduled on the COMMAND line of
any MainView AutoOPERATOR application. The complete syntax of the CSMACT EXEC is

\%CSMACT object_name command [DEP | NODEP] [REQ | NOREQ] ssid

where

<table>
<thead>
<tr>
<th>Parameter value</th>
<th>Description</th>
</tr>
</thead>
</table>
| object_name     | is the name of the object  
This parameter is required. |
| command         | specifies the command that is to be operated on the object 
This parameter is required. |
| DEP             | applies the command to the object and all its children |
| NODEP           | applies the command to the object only and not its children |
| REQ             | applies the command against the object’s parents |
| NOREQ           | does not apply the command to the object’s parents |
| ssid            | is the 4-character BBI subsystem identifier of the BBI-SS PAS where you want action to occur |

**Example 1**

To start an object named NET3 and all its parents, enter on the COMMAND line

\%CSMACT NET3 UP REQ

The NET object and all the objects that are defined as its parents, both direct parents and indirect parents, will be brought UP.

**Example 2**

To stop an object (named NET5) and all its children, enter on the COMMAND line

\%CSMACT NET5 DOWN DEP

The NET object and all the objects that are defined as its dependents will be brought DOWN.

**Example 3**

To move an object (named CICSA) to another BBI-SS PAS where CSM is operating, at the command prompt, enter on the COMMAND line
%CSMACT CICSA MOVE SS02

The CICSA object will be brought DOWN on the current subsystem and brought UP on subsystem SS02.

**CSMACT commands**

The following list shows valid commands that you can use with the CSMACT EXEC to affect the state of an object.

They are listed in alphabetical order.

<table>
<thead>
<tr>
<th>Command</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOUNCE</td>
<td>stops and immediately restarts an object if the original desired state for the object is UP. When used on an object that is UP, UPEARLY, UPFORCE, UPLOGICAL, or UPLOGICALE, the resultant desired state is UP. When issued on an object that is COMPLETE, the resultant desired state is COMPLETE.</td>
</tr>
<tr>
<td>CANCEL</td>
<td>issues the object’s cancel command When issued on an object in any state, the resultant desired state is DOWNFORCE.</td>
</tr>
<tr>
<td>DOWN</td>
<td>stops an object When issued on an object that is DOWN, the resultant desired state is DOWN. When issued on an object that is in any state except DOWN, the resultant desired state is DOWNFORCE. This command requires the DEP or NODEP parameter.</td>
</tr>
<tr>
<td>DOWNNEARLY</td>
<td>stops an object and continues under CSM control When issued on any object, the resultant desired state is DOWNNEARLY. This command requires the DEP or NODEP parameter.</td>
</tr>
<tr>
<td>DOWNLOGICAL</td>
<td>stops a grouping object and removes it from CSM control When issued on an object that is in any state, the resultant desired state is DOWNLOGICAL. This command requires the DEP or NODEP parameter.</td>
</tr>
<tr>
<td>DOWNLOGICALE</td>
<td>stops a grouping object and continues under CSM control When issued on an object that is in any state, the resultant desired state is DOWNLOGICALE. This command requires the DEP or NODEP parameter.</td>
</tr>
<tr>
<td>Command</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| DISABLE      | disables CSM's control over the object  
CSM does not start or stop this object. CSM ignores the object if it is part of the  
dependency chain of another object that CSM is starting or stopping.  
The object will remain in this condition until the object is enabled. |
| ENABLE       | enables CSM's control over the object  
CSM manages starting and stopping of this object. |
| MOVE         | moves an object to another subsystem  
This command requires the four-character subsystem ID of the moved-to subsystem.  
When an object that has an actual state UP is moved, the resultant actual state on  
the BBI-SS PAS that it is moved to is UPFORCE, and is DOWN on the BBI-SS PAS  
that it is moved from. |
| RESET        | sets an object's desired state based on its schedule and returns an object to CSM control  
When issued on an object that is UP, the resultant desired state is UP.  
When issued on an object that is DOWN, the resultant desired state is DOWN.  
You can use the DEP or REQ parameters but they are not required. |
| RESETC       | resets an object and the command counter to zero  
When issued on an object that is UP, the resultant desired state is UP.  
When issued on an object that is DOWN, the resultant desired state is DOWN.  
You can use the DEP or REQ parameters but they are not required. |
| UP           | starts an object  
When issued on an object that is UP, the resultant desired state is UP.  
When issued on an object that is in any state, the resultant desired state is  
UPFORCE.  
This command requires the REQ or NOREQ parameter. |
| UPEARLY      | starts an object and continues under CSM control  
When issued on an object that is in any state, the resultant desired state is  
UPEARLY.  
This command requires the REQ or NOREQ parameter. |
| UPLOGICAL    | starts a grouping object  
When issued on an object that is in any state, the resultant desired state is  
UPLOGICAL or COMPLETE.  
This command requires the REQ or NOREQ parameter. |
| UPLOGICALE   | starts a grouping object  
When issued on an object that is in any state, the resultant desired state is  
UPLOGICALE or COMPLETE.  
This command requires the REQ or NOREQ parameter. |
Command line interface to CSM

Use the CSM command line interface to manually monitor the state of CSM objects.

When an object group is defined in the local repository, you are asked to specify a command prefix. This unique value is used by CSM to determine which subsystem manages the command that is entered.

When you browse an object group definition, you will see an entry for the group’s command prefix. For example, the following figure shows the settings for group KMZ1.

Figure 72: Command prefix settings for group KMZ1

<table>
<thead>
<tr>
<th>BMC Software</th>
<th>Group Detail Control - Edit</th>
<th>MainView AutoOPERATOR</th>
<th>COMMAND</th>
<th>TGT</th>
<th>KMZ1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Name</td>
<td>KMZ1</td>
<td></td>
<td>Command Prefix</td>
<td>CSMKM</td>
<td></td>
</tr>
<tr>
<td>Rule Information:</td>
<td>BAOKMZ2.AAO63.KMZ1.UBBPARM</td>
<td>Ruleset Id: G2 Rule Prefix: ACM1 Suffix: 0001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object Statistics:</td>
<td>Audit: Y (Y/N)</td>
<td>Dataset Name: BAOKMZ.KMZ1.OFFLOAD</td>
<td>Disposition: MOD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Enter END command to process and return or CANCEL to leave

The command prefix specified for this group is CSMKM. Therefore, any commands to group KMZ1 through the interface must be prefixed with CSMKM.

In this example, where a CSM-enabled MainView AutoOPERATOR subsystem is using a repository that contains group KMZ1, you can issue commands from an MVS console against the CSM objects managed by group KMZ1.

Commands to the interface have the following format:

```
user_specified_prefix command objectname parm1 parm2
```

where the following commands can be issued:

Table 8: Command line interface commands and parameters

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOP</td>
<td>stops an object</td>
<td>no entry</td>
<td>no entry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EARLY</td>
<td>DEP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOGICAL</td>
<td>NODEP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOGICALE</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Purpose</td>
<td>Parameter 1</td>
<td>Parameter 2</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>START</td>
<td>starts an object</td>
<td>no entry</td>
<td>no entry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EARLY\nLOGICAL\nLOGICALE</td>
<td>REQNOREQ</td>
</tr>
<tr>
<td>CANCEL</td>
<td>cancels execution of an object</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>BOUNCE</td>
<td>stops and then immediately starts an object</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>MOVE</td>
<td>transfers monitoring and management from CSM enabled subsystem to another\nExecution of the object moves from one MVS image to another, if necessary.</td>
<td>Either the ID of the group to transfer control to or the value LOCAL, for the local group</td>
<td>none</td>
</tr>
<tr>
<td>RESET</td>
<td>resets object state to that specified in the object’s schedule</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>STATUS</td>
<td>reports the desired and actual state of one or more objects\nIf an object name is not specified then all objects in a particular group are reported</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>ENABLE</td>
<td>enables processing of a group for a local subsystem</td>
<td>Group ID to enable</td>
<td>SSID to enable group on</td>
</tr>
<tr>
<td>DISABLE</td>
<td>disables processing of a group for a local subsystem</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>ENABLE</td>
<td>enables an object to be controlled by CSM</td>
<td>objectname</td>
<td>none</td>
</tr>
<tr>
<td>DISABLE</td>
<td>disables an object to be controlled by CSM</td>
<td>objectname</td>
<td>none</td>
</tr>
</tbody>
</table>

In the case of the START and STOP commands, the values in the Parameter 1 column are based on the input parameters to the CSMACT EXEC. When a user starts an object with the interface command

```
user_specified_prefix START AAOCSM01 EARLY
```
CSMACT is scheduled on the appropriate subsystem with the following parameters:

**AAOCSM01 UPEARLY NOREQ**

Some examples follow.

- Object group KMZ1 manages AAOCSM01. The group’s command prefix is CSMKM. To stop AAOCSM01 and its children:

  **Example:** `CSMKM STOP AAOCSM01 DEP`

- To move object AAOCSM08 from group KMZ1 to group KMZ2. The group prefix for KMZ1 is CSMKM. The group prefix for KMZ2 is CSMK2.

  **Example:** `CSMKM MOVE AAOCSM08 KMZ2`

- To bounce object AAOCSM09 and the group KMZ2 that monitors it:

  **Example:** `CSMK2 BOUNCE AAOCSM09`

- To check the status of all of the objects monitored by group KMZ1:

  **Example:** `CSMKM STATUS`

- To disable object group KMZ1 which is running on subsystem KMZ1:

  **Example:** `CSMKM DISABLE`

- Object group KMZ1 manages AAOCSM22. The group’s command prefix is CSMKM. To start only AAOCSM22 but not its parents:

  **Example:** `CSMKM STOP AAOCSM22 NOREQ`

- To move object AAOCSM08 from group KMZ1 to group KMZ2. The group prefix for KMZ1 is CSMKM. The group prefix for KMZ2 is CSMK2.

  **Example:** `CSMKM MOVE AAOCSM08 KMZ2`

- To move object AAOCSM08 and its dependents from group KMZ1 to group KMZ2. The group prefix for KMZ1 is CSMKM. The group prefix for KMZ2 is CSMK2.

  **Example:** `CSMKM MOVE AAOCSM08 KMZ2 DEP`
**Note**

In addition to an MVS console, the command line interface commands can be entered on a third party software that simulates an MVS console (for example, the IBM product SDSF).

---

**Security considerations**

To issue the commands, you must have security access. If you do not have security access to issue these commands from the terminal session, you will not be able to issue these commands from the command line interface either. If you do not have access, you will receive the WTO:

```
ACM993E User <userid> is not authorized to issue <resource> command
```
File Transfer Protocol (FTP) events solution

This section describes how to use the FTP solution.

Implementing the exit program FTPOSTPR

MainView AutoOPERATOR ships an IBM® FTP exit, FTPOSTPR, that generates a message when the FTP Server finishes a file transfer to or from the host.

By implementing this exit, you can create a MSG-initiated Rule that is triggered by the message and automate an action when the file transfer completes.

Note: The MainView AutoOPERATOR for z/OS option is required for this solution to operate and a valid password must be installed. See the Installation System User Guide for information about obtaining and applying passwords.

To implement the exit program

1. Copy the FTPOSTPX load module from hilivel.BBLINK to a catalogued, APF-authorized data set.

2. Rename FTPOSTPX to FTPOSTPR.

3. Ensure that the newly defined data set is accessible to the FTP Server as defined in the IBM publication z/OS Communication Server: IP Configuration Reference.

In general, the data set must be specified in any one of the following ways where the data set:

- appears in the STEPLIB statement for the address space

- is listed within the linklist concatenation
is loaded to the Link Pack Area (LPA) module

When these steps are completed, the FTP Server can access the FTPOSTPR exit immediately and begins to use it. You do not need to restart the FTP Server to activate the exit.

--- Note ---

The MainView AutoOPERATOR FTPOSTPR exit must precede all other FTPOSTPR modules that might be listed in the loadlib search order (STEPLIB, linklist, and LPA). Only one FTPOSTPR can be active at a time.

---

Messages generated by the FTPOSTPR exit

The following shows an example of the messages generated when the FTP Server uses the FTPOSTPR exit.

**Figure 73: Messages generated by FTPOSTPR**

```
STC01496 00000290 AA0400I FTP Cmd STOR completed 511 511 00000290 AA0401I Dataset XXXMVS.ATLOG.TXT 511 00000290 AA0402I Completion Code 00 511 00000290 AA0403I Userid XXXMVS1 511 00000290 AA0404I Dir Type MVS Transfer Type SEQ 511 00000290 AA0405I Directory XXXMVS. 511 00000290 AA0406I Confidence Level INACTIVE 511 00000290 AA0407I Transfer completed successfully.
```

To ensure that the FTP Server message AA0402I reports an error completion code when data records are truncated, specify **TRUNCATE FALSE** in the FTP.DATA parameters for the FTP Server. If **TRUNCATE TRUE** (default) is specified, truncated records will not produce non-zero completion codes.

To ensure that the FTP Server message AA0406I includes the confidence level, specify **CHKCONFIDENCE TRUE** in the FTP.DATA parameters for the FTP Server. The default is **CHKCONFIDENCE FALSE**.

---

FTPOSTPR messages automation

The messages produced by the FTPOSTPR exit is a single multi-line WTO that you can automate responses to by creating a MainView AutoOPERATOR MSG-initiated Rule.

To create a Rule that fires based on the entire WTO as a single message, on the Selection Criteria -- MSG panel in the Rules Processor application, specify **ALL** on the **MLWTO Minor** field and **AA0400I** on the **Text ID** field. See the MainView

See the IBM document z/OS Communication Server: IP Configuration Reference for additional information about using FTP exits.
FTPOSTPR messages automation
View data solution

This section describes how to use MainView AutoOPERATOR to view windows-mode data in an EXEC.

MainView AutoOPERATOR provides the EXEC QAOVBAPI that you can use to access windows-mode view data in an EXEC. This EXEC fully supports all types of views (detail, tabular, summary, and hybrid) by allowing access to the regular view data and access the dictionary symbols used on hybrid views.

---

Note

You must have a MainView Explorer (MVE) host server address space started to use the QAOVBAPI EXEC.

The MVE task can be running on the local z/OS host or any other z/OS host in the IP network. The MVE task must be connected to a CAS that has access to the data the EXEC will retrieve. QAOVBAPI uses TCP/IP to communicate with the MVE to acquire the data and can access any MVE on any system that has TCP/IP connectivity to the system it is being executed on. See the MainView User Guide for more information about MVE.

---

Scheduling the QAOVBAPI EXEC

This section describes how to schedule the QAOVBAPI EXEC.

The data acquired by the EXEC is placed in REXX compound variables in the LOCAL variable pool where they can be retrieved by the calling EXEC using the IMFEXEC VGET command.

To use the QAOVBAPI EXEC

1. Write an EXEC.

2. Use the IMFEXEC SELECT command and use it schedule the QAOVBPAI EXEC.
Be sure to also specify WAIT(YES) on the IMFEXEC SELECT statement. The WAIT(YES) parameter is required to allow the data to be acquired before control is returned to the calling EXEC.

See the sample below for an example of how to make the call to QAOVBAPI and retrieve the variables created that contain the view data.

---

**Note**

The QAOVBAPI EXEC is completely independent from the IMFEXEC MV commands. No data or parameters are shared between the two methods of acquiring view data. See the *MainView AutoOPERATOR Advanced Automation Guide* for information about the IMFEXEC MV API.

---

**REXX EXEC scheduling QAOVBAPI example**

The following figure shows an example of a REXX EXEC that schedules the QAOVBAPI EXEC with the IMFEXEC SELECT WAIT(YES) command and also inspects the data that is returned in REXX stem variables.

**Figure 74: Sample EXEC calling QAOVBAPI EXEC**

```rexx
/* REXX */
parms = "PT(3986) US(DAVID) PW(123456) PR(MVAO) CT(DL72) VW(EXECMGR)"
    "DS(DIMXNORM DIMXHIGH DINORRUN)"
"IMFEXEC SELECT EXEC(QAOVBAPI" parms") WAIT(YES)"
/* retrieve all the variables from the local pool */
"IMFEXEC VLST MV* LOCAL"
DO ii = 1 TO IMFNOL
    "IMFEXEC VGET LINE"ii" LOCAL"
    varname = VALUE('LINE'ii)
    "IMFEXEC VGET "varname" LOCAL"
END
IF MV.SYM.DINORRUN > 10 then SAY "Caution: Too many EXECs running!"
DO ii = 1 TO MV.ROW.0
    IF MV.AA70EXEC.ii = "EXECABC" then SAY "Hey! Why is this running?"
END
```

The QAOVBAPI EXEC retrieves all the tabular data from the specified view and also the dictionary symbols DIMXNORM, DIMXHIGH, and DINORRUN from the hybrid portion of the view.

The example also specifies the IMFEXEC VGET command to retrieve all the variables from the LOCAL pool and interrogates some of them.

**QAOVBAPI EXEC parameters**

The following table lists the 6 required parameters:
The QAOVBAPI EXEC requires that you use these parameters and specify their values to define the connection to where the EXEC acquires the data.

Use the optional parameters to qualify, filter, or summarize the data. To acquire dictionary symbols from a hybrid view, specify one or more dictionary symbols with the DS() parameter. All other data fields on a view are automatically acquired.

For additional information about the terms used in the following table, see the MainView User Guide manual.

### Table 9: QAOVBAPI EXEC parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT()</td>
<td></td>
<td>port number</td>
</tr>
<tr>
<td>US()</td>
<td></td>
<td>user ID</td>
</tr>
<tr>
<td>PW()</td>
<td></td>
<td>password</td>
</tr>
<tr>
<td>PR()</td>
<td></td>
<td>product</td>
</tr>
<tr>
<td>CT()</td>
<td></td>
<td>context</td>
</tr>
<tr>
<td>VW()</td>
<td></td>
<td>view name</td>
</tr>
<tr>
<td>DA()</td>
<td>no</td>
<td>Specify the starting date of the data. To specify a specific date, use the format ddmmmyyyy; for example: 21FEB2011. You can also specify relative symbols such as TODAY and YESTERDAY to specify the date. See the TIME() command in the MainView User Guide for more information. Use this value with the TI(), DU() and UN() parameters to define a TIME reference. The default is the current date when the TI(), DU(), or UN() parameters are specified; or no TIME reference is used.</td>
</tr>
<tr>
<td>DS()</td>
<td>no</td>
<td>Specify a list of one or more dictionary symbols that the EXEC will retrieve. Dictionary symbols are used primarily on hybrid views but can also appear elsewhere. Use the online help for the view to determine the dictionary symbol name that the EXEC should return. Separate multiple dictionary symbols by one or more blanks. For example: DS(DIMXNORM) DS(DIMXNORM DIMXHIGH DINORRUN) When this parameter is not specified, the EXEC does not retrieve dictionary symbol data.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>DU()</td>
<td>no</td>
<td>Specify a duration (in minutes) using a number between 1 through 3 that is used with the UN() parameter. Use this value with the DA(), TI(), and UN() parameters to define a TIME reference for the data that is to be acquired. The default is 001 if the DA(), TI(), or UN() parameters are specified; or no TIME reference is used.</td>
</tr>
<tr>
<td>HELP</td>
<td>no</td>
<td>When this parameter is specified with no other parameters, it causes the EXEC to write help messages to the BBI-SS Journal. These messages show all of the available parameters and other information that assists in using the EXEC.</td>
</tr>
<tr>
<td>HO()</td>
<td>no</td>
<td>Specify the host name (or IP address) of the system that has the MainView Explorer host server task running on it. The default is the local host which is the BBI-SS where the EXEC is scheduled.</td>
</tr>
<tr>
<td>HQ()</td>
<td>no</td>
<td>Specify a high-level data set qualifier. Use this parameter only if the view you are accessing needs to write to an output data set. The default is the value specified with the US() parameter.</td>
</tr>
<tr>
<td>PR()</td>
<td>yes</td>
<td>Specify the product name that owns the view to be accessed. Use the MainView PLEX Manager view, PLEXPROD, to display a list of available products and use the product abbreviation, shown in the Product column of the PLEXPROD view in this parameter. For example, MVAO is the product name for MainView AutoOPERATOR, MVDB2 is the product code for MainView for DB2, and so on.</td>
</tr>
<tr>
<td>PT()</td>
<td>yes</td>
<td>Specify the port number of the running MainView Explorer host server task to contact to acquire the data.</td>
</tr>
<tr>
<td>PW()</td>
<td>yes</td>
<td>Specify the password to allow logon to access the data.</td>
</tr>
<tr>
<td>QW</td>
<td>no</td>
<td>Specify an optional QWHERE clause to apply to the view. The default is that this parameter is not used.</td>
</tr>
<tr>
<td>RW()</td>
<td>no</td>
<td>Specify the number of rows of data for the EXEC to return. The default is all of the rows available on the view are returned. <strong>Note:</strong> When accessing a detail view (not a tabular or hybrid view), the RW() parameter is ignored. By definition, a detail view has only one row of data.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>SR()</td>
<td>no</td>
<td>Specify the starting row number of data that you want returned by the EXEC. The default is 1. Using this parameter begins the data retrieval at the specified row number and causes the first set of variables produced to use this row number as the starting value. For example, if you specify SR(5), first row of data is row 5 and the first MV.DATA.row.col variable is MV.DATA.5.1. <strong>Note:</strong> This parameter is ignored by the NVBAPI when a view that uses MainView Logger data is referenced. This is because there is no beginning row number in logging data that constantly has new records added. When accessing a detail view (not a tabular or hybrid view), the SR() parameter is ignored. A detail view has only one row of data.</td>
</tr>
<tr>
<td>ST()</td>
<td>no</td>
<td>Specify the STEM root value to use for the variables that are created in the LOCAL variable pool. The value must conform to REXX STEM variable naming standards. The default is that all REXX compound variables created by the EXEC will start with MV.</td>
</tr>
<tr>
<td>SV()</td>
<td>no</td>
<td>Specify the server name. When this parameter is not specified, the default is an asterisk (*)</td>
</tr>
<tr>
<td>SY()</td>
<td>no</td>
<td>Specify the system name. When this parameter is not specified, the default is an asterisk (*)</td>
</tr>
<tr>
<td>TI()</td>
<td>no</td>
<td>Specify the starting time value for the data. The format is hh:mm:ss. Additional specifications include the use of an asterisk (*) and equal sign (=). See the TIME() command in the MainView User Guide for more information. Use this value with the DA(), DU() and UN() parameters to define a TIME reference for the data that is to be acquired. The default is the current time when the DA(), DU(), or UN() parameters are specified; or no TIME reference is used.</td>
</tr>
<tr>
<td>TR()</td>
<td>no</td>
<td>Specify the number of seconds after a timeout before retrying the request. The default is 20 seconds.</td>
</tr>
</tbody>
</table>
Variables returned by the QAOVBAPI EXEC

The output from the QAOVBAPI is a set of variables created in the LOCAL variable pool.

The variables can be accessed with an EXEC that uses the IMFEXEC VGET statement.

The compound variable root is shown as MV. This is the default value. You can change the root value with the ST() parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
</table>
| UN()      | no       | Specify the units associated with the duration of the summarization. Valid values can be:  
- I (intervals)  
- M (minutes)  
- H (hours)  
- D (days)  
Use this value with the DA(), TI(), and DU() parameters to define a TIME reference for the data that is to be acquired. The default is I (intervals) when the DA(), TI(), or DU() parameters are specified; or no TIME reference is used. |
| US()      | yes      | Specify the user ID to use to login to the target system to define the credentials for access to the data. |
| VP()      | no       | Specify the view parameters that would normally be specified on the view command line. Multiple values are separated with a blank space; for example:  

```
VP(DX3100* AAO*)  
VP(DX3100*)
```

The default is none. |
| VW()      | yes      | Specify the view name. |
| WC()      | no       | Specify the number of seconds to wait for a connection to be made. The default is 10 seconds. |
| WH()      | no       | Specify an optional WHERE clause to apply to the view. The default is that this parameter is not used. |
Column numbers always start on the left edge of the view with 1 and proceed incrementally to the right. Columns that are excluded or hidden on the view are not included in the column information.

Table 10: Variables returned by the QAOVBAPI EXEC

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Variable value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV.LOCALDATE</td>
<td>contains the local date associated with the data formatted as: yyyy-mm-dd</td>
</tr>
<tr>
<td>MV.LOCALTIME</td>
<td>contains the local time associated with the data formatted as: hh:mm:ss:tt</td>
</tr>
<tr>
<td>MV.GMTDATE</td>
<td>contains the GMT date associated with the data formatted as: yyyy-mm-dd</td>
</tr>
<tr>
<td>MV.GMTTIME</td>
<td>contains the GMT time associated with the data formatted as: hh:mm:ss:tt</td>
</tr>
<tr>
<td>MV.COL.0</td>
<td>for tabular views, contains the total number of columns in the view For detail views, this is the number of data fields on the view.</td>
</tr>
<tr>
<td>MV.ROW.0</td>
<td>for tabular views, contains the number of rows of data that have been returned in variables from the view For detail views, this number is always 1.</td>
</tr>
<tr>
<td>MV.STARTROW.0</td>
<td>contains the starting row number of the data (relative to 1) For example, if SR(15) was specified as input parameter, then MV.STARTROW.0 = 15 will be returned.</td>
</tr>
<tr>
<td>MV.COL.ID.nn</td>
<td>for each column number (nn), contains the internal column ID</td>
</tr>
<tr>
<td>MV.COL.NAME.nn</td>
<td>for each column number (nn), contains the field name of the data</td>
</tr>
<tr>
<td>MV.COL.WIDTH.nn</td>
<td>for each column number (nn), contains the width of the column as it is shown on the view</td>
</tr>
<tr>
<td>MV.COL.HDR.nn</td>
<td>for each column number (nn), contains the column heading literal</td>
</tr>
<tr>
<td>MV.DATA.row.col</td>
<td>for each row and column, contains the data value in that cell</td>
</tr>
<tr>
<td>MV.name.row</td>
<td>for each row and each column's field name, contains the value in that cell For example, if column 3 of the view uses a field name of AR01PCNT and the value in row four of the view is 5163, the variable values are as follows: MV.COL.NAME.3 = AR01PCNT MV.ARD01PCNT.4 = 5163</td>
</tr>
<tr>
<td>MV.SYM.symbol</td>
<td>for each symbol name specified in the DS() parameter, a variable with a name that uses the symbol's name will be created and it will contain the value of the that symbol For example, if DS(DIMXNORM) is specified and the value of that dictionary symbol is 15, the variable value is as follows: MV.SYM.DIMXNORM = 15</td>
</tr>
</tbody>
</table>
Return Codes

The following table describes the return codes.

**Table 11: Return codes for the QAOVBAPI EXEC**

<table>
<thead>
<tr>
<th>Return code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>request was successful</td>
</tr>
<tr>
<td>8</td>
<td>syntax error in parameters or invalid value in parameters.</td>
</tr>
<tr>
<td>16</td>
<td>a fatal error occurred; processing was terminated</td>
</tr>
<tr>
<td></td>
<td>See related messages for more information.</td>
</tr>
</tbody>
</table>
Enterprise Incident Management (EIM) solution

The Enterprise Incident Management (EIM) solution allows MainView AutoOPERATOR automation to generate BMC Remedy trouble tickets.

Overview

The EIM solution accesses BMC services between MainView AutoOPERATOR and Remedy, allowing you to define MainView AutoOPERATOR ALERTs that can create Remedy tickets. The EIM solution consists of a set of REXX EXECs and MainView AutoOPERATOR Rules that can generate a Remedy ticket when automation detects an ALERT in the ITICKET queue.

The EIM solution can also manage the Remedy ticket when the status of the ALERT changes in the ITICKET queue:

- When the ALERT escalates, the EIM solution updates the Remedy ticket.
- When the ALERT is deleted, the EIM solution sets the Remedy ticket's status to Resolved. (You can customize EIM to set the status to Closed, if you prefer.)

The EIM solution uses the REXX interface of the Atrium Integrated Services for z/OS (AISZ), a component that ships with MainView AutoOPERATOR, to communicate with Remedy via Web Services (WS).

Note
The EIM solution requires version 3.5.00 (or later) of the BMC Service Resolution (BSR) product.

To implement this solution, complete the following procedures:

- “Customizing the EIM solution” on page 140
- “Starting EIM” on page 146
Customizing the EIM solution

Complete the following procedures to customize the EIM solution based on your site’s needs. You will configure UNIX System Services (USS) to work with the solution, edit parameters in two BBPARM members, and (optionally) set up EIM Rules.

**To configure UNIX System Services to work with the EIM solution**

This procedure loads UNIX System Services (USS) with Apache Axis2 configuration information.

1. Make a copy of the JCL found in `HLQ.BBSAMP(ARLODHFS)` in your private libraries, where `HLQ` is your site’s high-level qualifier.

2. Edit the JOB statement to comply with your site standards.

3. Edit the SET statement with a valid file name prefix.

4. *(optional)* In the BBPARM member AISCFG00 configuration parameters, adjust the USS directory structure, if needed.

   **Tip**
   
   The next step provides details about making changes in AISCFG00.

5. Submit the job.

**To edit parameters in BBPARM member AISCFG00**

1. Make a copy of AISCFG00.

2. Add the member to the UBBPARM data set concatenation of your MainView AutoOPERATOR PAS JCL.

3. Review and edit the parameters listed in the following table to define the AR Remedy system and the EIM solution requirements in `HLQ.UBBPARM(AISCFG00)`:

   **Note**
   
   The AISCFG00 member includes additional parameters that are not shown in this table. *Do not change the values of those additional parameters.* Edit only the parameters shown in the table.

   When entering values, remember that they are case sensitive.
Table 12: Editable parameters in BBPARM member AISCFG00

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
</table>
| common:timezone=value | *(optional)* Specifies the time zone to use. The default is the time zone in which the IBM z/OS system is running. You can override this value.  
  **Example:** common:timezone=GMT-6  
  **Note:** If you specify a time zone, you must manually edit the value when going to, or from, Daylight Saving Time (DST). |
| ARService:*=hostname | Specifies the fully qualified domain name or internet protocol (IP) address of the Remedy Web Services (WS) server  
  **Example:** ARService:*=host=myRemedy.myCompany.com |
| ARService:*=port=portName | Specifies the port number of the WS server  
  **Example:** ARService:*=port=8080  
  **Note:** The Remedy port number is usually defined at 8080. |
| ARService:*=server=serverName | Specifies the Remedy WS server name. This is usually the name of the server.  
  **Example:** ARService:*=server=myRemedy |
| ARService:*=user=userName | Specifies the Remedy user ID that EIM uses to establish the connection to Remedy |
| ARService:*=password=password | Specifies the Remedy user ID password that EIM uses to validate the connection to Remedy |
| ARService:*=authentication=name | Specifies the authenticator. You can use a NULL value, or you can specify ARSystem. |
| ARService:*=AXIS2_HOME=home DirectoryAddress | Specifies the home directory for the AXIS2 XML specified in the ARLODHFS job in HLQ.BBSAMP.  
  **Example:** ARService:*=AXIS2_HOME=HFS:/var/bmcuim/v540/aisz/axis2c/client |

**To edit parameters in BBPARM member EIMCFG00**

1. Make a copy of EIMCFG00.
2. Add the member to the UBBPARM data set concatenation of your MainView AutoOPERATOR PAS JCL.
3. Review and edit the parameters shown in the following table to customize the parameter values in HLQ.UBBPARM(EIMCFG00).
The EIM solution uses the information provided in EIMCFG00 to generate variables in the MainView AutoOPERATOR SHARED variable pool. By default, the EIM solution creates all of its SHARED variables with a prefix of TK. (If you use the TK prefix for SHARED variables, you can change the prefix in the next step.)

### Table 13: BBPARM member EIMCFG00 parameters

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parmlib=parmFileName</td>
<td>Specifies the DD name and member name of the AISZ configuration member</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> Parmlib='DDN:BBIPARM(AISCFG00)'</td>
</tr>
<tr>
<td>User=userName</td>
<td>Specifies the Remedy user ID that EIM uses to establish the connection to Remedy</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This value overrides the ARService:<em>:user=userName value that is specified in the AISZ AISCFG</em> member.</td>
</tr>
<tr>
<td>Password=password</td>
<td>Specifies the Remedy user ID password that EIM uses to validate the connection to Remedy</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This value overrides the ARService:<em>:password= password value that is specified in the AISZ AISCFG</em> member.</td>
</tr>
<tr>
<td>Company=companyName</td>
<td>Specifies your company name that is defined in Remedy and that EIM will use to assign tickets</td>
</tr>
<tr>
<td></td>
<td>This parameter is required.</td>
</tr>
<tr>
<td>Notes=value</td>
<td>Specifies what to put in the Notes field of the ticket</td>
</tr>
<tr>
<td></td>
<td>This parameter is optional.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> Notes='&amp;AMFTEXT'</td>
</tr>
<tr>
<td></td>
<td>Notes='Ticket generated by z/OS'</td>
</tr>
<tr>
<td>Categorization_Tier1=value</td>
<td>Specifies the operational category tiers that EIM assigns to the ticket. Users can use these tiers to preassign the ticket to Help desk users.</td>
</tr>
<tr>
<td></td>
<td>The tiers specified must be defined in Remedy.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> Categorization_Tier1='Event'</td>
</tr>
<tr>
<td></td>
<td>Categorization_Tier2='z/OS'</td>
</tr>
<tr>
<td></td>
<td>Categorization_Tier3=''</td>
</tr>
<tr>
<td>Command</td>
<td>Function</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HPD=name</td>
<td>Specifies the configuration management database (CMDB) configuration item (CI) name that is impacted by the tickets</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If the HPD_CI field is set to NULL and the BMC Discovery for z/OS product is connected to MainView AutoOPERATOR via the shared VSAM file, the EIM solution assigns the CI name associated with the events address space if discovered.</td>
</tr>
<tr>
<td>Product_Category_Tier1=tierNumber</td>
<td>Specifies the CMDB CI name that is impacted by this ticket</td>
</tr>
<tr>
<td>Product_Category_Tier2=tierNumber</td>
<td><strong>Note:</strong> Remedy automatically completes these fields, using CI information from the CMDB, provided that:</td>
</tr>
<tr>
<td>Product_Category_Tier3=tierNumber</td>
<td>■ The HPD_CI field contains the CI name obtained from BMC Discovery for z/OS.</td>
</tr>
<tr>
<td>Product_Model_Version=modelNumber</td>
<td>■ The discovered CI exists in the CMDB.</td>
</tr>
<tr>
<td>Product_Name=name</td>
<td></td>
</tr>
<tr>
<td>Manufacturer=name</td>
<td></td>
</tr>
<tr>
<td>UrgencyCritical=value</td>
<td>Specifies the urgency values for the ALERT</td>
</tr>
<tr>
<td>UrgencyMajor=value</td>
<td>You can change the AMFPRIOR to Remedy Urgency translation table to fit your requirements.</td>
</tr>
<tr>
<td>UrgencyMinor=value</td>
<td>Valid urgency values are:</td>
</tr>
<tr>
<td>UrgencyWarning=value</td>
<td>■ 1 (Critical)</td>
</tr>
<tr>
<td>UrgencyInformational=value</td>
<td>■ 2 (High)</td>
</tr>
<tr>
<td>UrgencyClearing=value</td>
<td>■ 3 (Medium)</td>
</tr>
<tr>
<td></td>
<td>■ 4 (Low)</td>
</tr>
<tr>
<td></td>
<td>The following list contains the default definitions:</td>
</tr>
<tr>
<td></td>
<td>■ UrgencyCritical='1-Critical'</td>
</tr>
<tr>
<td></td>
<td>■ UrgencyMajor='2-High'</td>
</tr>
<tr>
<td></td>
<td>■ UrgencyMinor='3-Medium'</td>
</tr>
<tr>
<td></td>
<td>■ UrgencyWarning='3-Medium'</td>
</tr>
<tr>
<td></td>
<td>■ UrgencyInformational='4-Low'</td>
</tr>
<tr>
<td></td>
<td>■ UrgencyClearing='4-Low'</td>
</tr>
</tbody>
</table>
**Command**

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImpactCritical=value</td>
<td>Specifies the impact values for the ALERT</td>
</tr>
<tr>
<td>ImpactMajor=value</td>
<td>You can change the AMFPRIOR to Remedy Urgency translation table to fit your requirements.</td>
</tr>
<tr>
<td>ImpactMinor=value</td>
<td>Valid urgency values are:</td>
</tr>
<tr>
<td>ImpactWarning=value</td>
<td>■ 1 (Extensive/Widespread)</td>
</tr>
<tr>
<td>ImpactInformational=value</td>
<td>■ 2 (Significant/Large)</td>
</tr>
<tr>
<td>ImpactClearing=value</td>
<td>■ 3 (Moderate/Limited)</td>
</tr>
</tbody>
</table>

The following list contains the default definitions:

- ImpactCritical='1-Extensive/Widespread'
- ImpactMajor='2-Significant/Large'
- ImpactMinor='3-Moderate/Limited'
- ImpactWarning='3-Moderate/Limited'
- ImpactInformational='4-Minor/Localized'
- ImpactClearing='4-Minor/Localized'

---

**To set up EIM Rules (optional)**

The EIM solution Rule Set AAORULTK includes several Rules that control EIM processing. The default Rules settings create a Remedy ticket when an ALERT is queued to the ITICKET queue, update the ticket when the ALERT is escalated, and resolve the ticket when the ALERT is deleted.

**WARNING**

The AAORULTK Rule Set is defined and distributed with a Rule Set strategy of FIRST. You must not change this specification.

1. Review the EIM solution Rules (shown in the following table), and enable or disable Rules based on how you want the solution to handle Remedy tickets.

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TKCREATE</td>
<td>Creates a ticket when the ALERT is added. The TKCREATE Rule is defined with a match limit criteria that results in this Rule being suspended when the rate is greater than 10 messages per 60 seconds.</td>
</tr>
<tr>
<td>TKFAIL</td>
<td>Writes a journal message if a high number of create message requests occur. The TKFAIL Rule fires when the TKCREATE Rule is suspended.</td>
</tr>
<tr>
<td>TKESCAL</td>
<td>Updates the ticket when the ALERT is escalated. If you do not want EIM to update the Remedy tickets when an ALERT is escalated, disable the TKESCAL Rule.</td>
</tr>
<tr>
<td>Rule name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| TKRESOLV  | Resolves the ticket when the ALERT is deleted (default setting)  
If you prefer that EIM close the ticket instead of resolving it, disable TKRESOLV  
and enable TKCLOSE.  
If you do not want EIM to modify the status to Resolved, disable TKRESOLV.  
**Note:** You can have either TKRESOLVE or TKCLOSE enabled, but not both. If one is enabled, the other should be disabled. |
| TKCLOSE   | Closes the ticket when the ALERT is deleted  
By default, the TKCLOSE is disabled to allow EIM to set the status of Remedy  
tickets to Resolved. To have EIM close the ticket, disable TKRESOLV and enable  
TKCLOSE.  
If you do not want EIM to change the status to Closed, disable TKCLOSE.  
**Note:** You can have either TKRESOLVE or TKCLOSE enabled, but not both. If one is enabled, the other should be disabled. |
| TKLOG     | Logs Atrium Integrated Services for z/OS (AISZ) errors to the MainView  
AutoOPERATOR Journal |

2. If you use the TK prefix for SHARED variables for your existing automation, edit each of the following Rules to use a different prefix:

**Note**
The EIM solution uses SHARED variables and, by default, the SHARED variables use prefix TK.

- TKCREATE
- TKESCAL
- TKRESOLV
- TKCLOSE

On the A1 (Action Specification(s) I) panel for each Rule, change TK to a different prefix with the PREF( ) parameter. The following shows an example of changing the TK prefix to ITKT:

```
* EXEC Name/Parms ==> PRI(S):QAOINCDT PREF("ITKT") . . . . . .
```

3. If you want to change the name that EIM uses for the ALERT queue name (ITICKET), change it by editing each of the following Rules in Rule Set AAORULTK:

- TKCREATE
- TKFAIL
- TKESCAL
- TKRESOLV
- TKCLOSE
On the S1 (Selection Criteria) panel for each Rule, change ITICKET to the new ALERT queue name. The following shows an example of changing the ALERT queue name from ITICKET to REMEDY:

<table>
<thead>
<tr>
<th>Queue</th>
<th>===&gt; REMEDY</th>
<th>Alert Queue</th>
</tr>
</thead>
</table>

**Where to go from here**

When you finish setting up Rules, you are ready to start the EIM solution as instructed in “Starting EIM” on page 146.

**Starting EIM**

After customizing the EIM solution, use the following procedures to start the solution. You will modify the MainView AutoOPERATOR start procedure JCL, activate the EIM Rule Set, and restart the PAS.

**To modify the start procedure**

1. Add the following statement to the STEPLIB concatenation in the MainView AutoOPERATOR started procedure JCL:

   ```
   //         DD  DSN=HLQ.XXLINK,DISP=SHR
   ```

   You can view an example of the JCL in HLQ.BBSAMP(EIMJCL)

2. Add the DDname for the AISZPRT, as follows:

   ```
   //AISZPRT DD SYSOUT=*,DCB=BLKSIZE=1330
   ```

   If you start the MainView AutoOPERATOR PAS started procedure JCL with SUB=MSTR specified, change the AISZPRT DD name to:

   ```
   //AISZPRT DD SYSOUT=DUMMY,DCB=BLKSIZE=1330
   ```

**To activate the EIM Rule Set and restart the PAS**

1. Update BBPARM member AAOPRM.xx to set the Rule strategy and specify the EIM solution Rule Set:

   a. Enter the suffix **TK** for the AAORULTK Rule Set in the RULESET= parameter list.

      For example, enter `RULESET=[00, 01, TK].`

   b. Enter **RULESCAN=I** to specify a Rule Set automation strategy of Individual.
2 Restart the MainView AutoOPERATOR PAS with START=COLD.

The EIM Rules are now active and will monitor the ITICKET ALERT queue.
zAware LPAR data solution

The MainView AutoOPERATOR zAware LPAR data solution provides summary data obtained from the IBM System zAdvanced Workload Analysis Reporter (IBM zAware) appliance that analyzes data from the OPERLOG.

You should be familiar with the zAware appliance and the data that is displayed on the zAware user interface (UI) before using this solution. Many input and output variables in this solution correspond to the data displayed on the zAware UI.

The solution includes the following EXECs, which allow you to review and automate the LPAR data that the zAware UI displays:

- QAOZAWR1 is a sample EXEC that shows which input variables you need to set and the variables that contain data returned by zAware. QAOZAWR1 is provided in the HLQ.BBSAMP data set (where HLQ represents your high-level qualifier).

- QAOZAWR2 performs communication with zAware and is provided in the HLQ.BBPROC data set.

Customizing the QAOZAWR1 EXEC

To use this solution, you must ensure that these EXECs are customized, and reside in the SYSPROC concatenation for the MainView AutoOPERATOR product address space (PAS).

To customize the QAOZAWR1 EXEC:

1. Implement an Application Transparent Transport Layer Security (AT-TLS) policy that allows the MainView AutoOPERATOR PAS where the solution is implemented to communicate with zAware through SSL encryption.

For more information about the AT-TLS policy and zAware, see the following IBM publications:

- System z Advanced Workload Analysis Reporter (IBM zAware) Guide
- Extending z/OS System Management Functions with IBM zAware
2 Make a copy of the BBSAMP member QAOZAWR1 EXEC.

3 Edit the EXEC as follows:

   a Search for @@INPUT@@ and edit the input variables in this section with valid values for your site.

   b Search for @@OUTPUT@@ and edit this section of the EXEC to perform the notification and automation actions that you want for your site.

4 Ensure the your customized copy of the QAOZAWR1 EXEC is in the SYSPROC concatenation for the MainView AutoOPERATOR BBI-SS PAS.

   For example, many sites use a HLQ.UBBPROC data set for their user-written EXECs.

5 Create a TIME-event Rule that schedules the QAOZAWR1 EXEC at the time interval that you want.

   **Note**
   Avoid setting the initial start time on the 10-minute interval of the hour. BMC recommends that you specify the Selection Criteria for the TIME-event Rule to start firing at 1 minute after midnight, as the following example shows:
   
   Start Specification:
   Start Time ===> 00:01:00
   Frequency:
   Interval ===> 00:10:00
   Stop Specification:
   Stop time ===> 23:51:00

   For example, you could specify that the Rule schedules the QAOZAWR1 EXEC every 10 minutes—the same interval as the zAware interval.

### LPAR data solution variables values

The following table lists the variables that the QAOZAWR1 EXEC passes to QAOZAWR2:

You can manually specify the variable values listed in the table below directly in the EXEC, or you can predefine the value in a PROFILE variable via the windows-mode EZAOSOL menu. Using a PROFILE variable is preferable as the variable value can be easily changed in the future, without having to edit the EXEC source code.
more information about the EZAOSOL menu, see the *MainView AutoOPERATOR Basic Automation Guide: Volume 2.*

Table 14: QAOZAWR1 input variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_TCP_STACK</td>
<td>is the TCP/IP address space name of the stack to be used. This value is optional unless you want to use a non-default TCP/IP stack.</td>
</tr>
<tr>
<td>I_ZAWARE_HOST</td>
<td>is the IP address or host name of your zAware host</td>
</tr>
<tr>
<td>I_ZAWARE_PORT</td>
<td>is the listening port number used by your zAware. Normally this is 443 but note that it could different in your environment.</td>
</tr>
<tr>
<td>I_ZAWARE_USERID</td>
<td>is the zAware user ID</td>
</tr>
<tr>
<td>I_ZAWARE_PSWD</td>
<td>is the zAware password for the user ID.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The password text is stored in clear text and is visible whether you specify it in the EXEC or as a PROFILE variable.</td>
</tr>
<tr>
<td>I_SYSPLEX_LPAR.0</td>
<td>Number of the zAware SYSPLEX-LPAR designation</td>
</tr>
<tr>
<td>I_SYSPLEX_LPAR.n</td>
<td>Number of SYSPLEX-LPAR stem variables that you have created within single quotation marks. The zAware SYSPLEX-LPAR designation is a two-word value (the SYSPLEX name followed by the LPAR name). The names are separated by a blank space, and the entire string is enclosed in single quotation marks. For each zAware SYSPLEX-LPAR that is specified in I_SYSPLEX_LPAR.0, you must define a variable if you want to obtain data from that SYSPLEX-LPAR. For example, assume that you want to obtain the data from the following two zAware SYSPLEX-LPARs: ■ LPAR SYSA runs in SYSPLEX BBPLEX01 ■ LPAR SYSB runs in SYSPLEX BBPLEX02 You must enter the variables as shown in the following example: <code>I_SYSPLEX_LPAR.0 = 2</code> <code>I_SYSPLEX_LPAR.1 = 'BBPLEX01 SYSA'</code> <code>I_SYSPLEX_LPAR.2 = 'BBPLEX02 SYSB'</code></td>
</tr>
</tbody>
</table>
The following table lists the MainView AutoOPERATOR LOCAL variables and values that are created as output after QAOZAWR1 EXEC executes, and are available to QAOZAWR2.

### Table 15: QAOZAWR1 input variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L_CURRENT_INTERVAL</td>
<td>Current interval number</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.SYSID</td>
<td>SYSPLEX-LPAR designation from zAware output</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.MSGID.0</td>
<td>Number of intervals and (always 144)</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.MSGID.n</td>
<td>Number of unique message IDs</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.ASCORE.n</td>
<td>Anomaly Score for the interval</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.INTERVAL</td>
<td>Current zAware interval number</td>
</tr>
</tbody>
</table>
zAware Interval data solution

The MainView AutoOPERATOR zAware Interval data solution provides detailed data obtained from the IBM zAware appliance that analyzes data from the OPERLOG.

You should be familiar with the zAware appliance and the data that is displayed on the zAware user interface (UI) before using this solution. Many input and output variables in this solution correspond to the data displayed on the zAware UI.

The solution includes the following EXECs, which allow you to review and automate the interval data that the zAware UI displays:

- **QAOZINT1** is a sample EXEC that shows which input variables you need to set and the variables that contain data returned by zAware. QAOZAWR1 is provided in the `HLQ.BBSAMP` data set (where `HLQ` represents your high-level qualifier).

- **QAOZINT2** performs communication with zAware and is provided in the `HLQ.BBPROC` data set.

To use this solution, you must ensure that these EXECs are customized, and reside in the SYSPROC concatenation for the MainView AutoOPERATOR product address space (PAS).

**Customizing the QAOZINT1 EXEC**

To use this solution, you must ensure that these EXECs are customized, and reside in the SYSPROC concatenation for the MainView AutoOPERATOR product address space (PAS).

**To customize the QAOZINT1 EXEC:**

1. Implement an Application Transparent Transport Layer Security (AT-TLS) policy that allows the MainView AutoOPERATOR PAS where the solution is implemented to communicate with zAware through SSL encryption.
For more information about the AT-TLS policy and zAware, see the following IBM publications:

- *System z Advanced Workload Analysis Reporter (IBM zAware) Guide*
- *Extending z/OS System Management Functions with IBM zAware*

2 Make a copy of the BBSAMP member QAOZINT1 EXEC.

3 Edit the EXEC as follows:

   a Search for `@@INPUT@@` and edit the input variables in this section with valid values for your site.

   b Search for `@@OUTPUT@@` and edit this section of the EXEC to perform the notification and automation actions that you want for your site.

4 Ensure your customized copy of the QAOZINT1 EXEC is in the SYSPROC concatenation for the MainView AutoOPERATOR BBI-SS PAS.

   For example, many sites use a `HLQ.UBBPROC` data set for their user-written EXECs.

5 Create a TIME-event Rule that schedules the QAOZINT1 EXEC at the time interval that you want.

   **Note**
   Avoid setting the initial start time on the 10-minute interval of the hour. BMC recommends that you specify the Selection Criteria for the TIME-event Rule to start firing at 1 minute after midnight, as the following example shows:

   **Start Specification:**
   ```
   Start Time ===> 00:01:00
   Frequency:
   Interval ===> 00:10:00
   Stop Specification:
   Stop time ===> 23:51:00
   ```

   For example, you could specify that the Rule schedules the QAOZINT1 EXEC every 10 minutes—the same interval as the zAware interval.
Interval data solution variables values

The following table lists the variables that the QAOZINT1 EXEC passes to QAOZINT2:

You can manually specify the variable values listed in the table below directly in the EXEC or, you can predefine the value in a PROFILE variable via the windows-mode EZAOSOL menu. Using a PROFILE variable is preferable so if the value changes in the future, the EXEC will continue to work whereas you must remember to manually find and edit any values you specify directly in the EXEC. For more information about the EZAOSOL menu, see the MainView AutoOPERATOR Basic Automation Guide: Volume 2.

Table 16: QAOZINT1 input variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_TCP_STACK</td>
<td>Is the TCP/IP address space name of the stack to be used</td>
</tr>
<tr>
<td></td>
<td>This value is optional unless you want to use a non-default TCP/IP stack.</td>
</tr>
<tr>
<td>I_ZAWARE_HOST</td>
<td>is the IP address or host name of your zAware host</td>
</tr>
<tr>
<td>I_ZAWARE_PORT</td>
<td>is the listening port number used by your zAware</td>
</tr>
<tr>
<td></td>
<td>Normally this is 443 but note that it could differ in your environment.</td>
</tr>
<tr>
<td>I_ZAWARE_USERID</td>
<td>is the zAware user ID</td>
</tr>
<tr>
<td>I_ZAWARE_PSWD</td>
<td>is zAware password for the user ID</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The password text is visible whether you specify it in the EXEC or as a PROFILE variable.</td>
</tr>
<tr>
<td>I_SYSPLEX_LPAR.0</td>
<td>Number of the zAware SYSPLEX-LPAR designation</td>
</tr>
</tbody>
</table>
### **I_SYSPLEX_LPAR.\(n\)**

Number of SYSPLEX-LPAR stem variables that you have created within single quotation marks.

The zAware SYSPLEX-LPAR designation is a two-word value (the SYSPLEX name followed by the LPAR name). The names are separated by a blank space, and the entire string is enclosed in single quotation marks.

For each zAware SYSPLEX-LPAR that is specified in \(I\_SYSPLEX\_LPAR.0\), you must define a variable if you want to obtain data from that SYSPLEX-LPAR. For example, assume that you want to obtain the data from the following two zAware SYSPLEX-LPARGS:

- LPAR SYSA runs in SYSPLEX BBPLEX01
- LPAR SYSB runs in SYSPLEX BBPLEX02

You must enter the variables as shown in the following example:

\[
\begin{align*}
I\_SYSPLEX\_LPAR.0 &= 2 \\
I\_SYSPLEX\_LPAR.1 &= 'BBPLEX01 SYSA' \\
I\_SYSPLEX\_LPAR.2 &= 'BBPLEX02 SYSB'
\end{align*}
\]

### **I_SYSPLEX_LPAR.TIME.\(n\)**

*(optional)* Exact interval that the solution will retrieve data for the specified LPARs in the format: \(yyyyymmddhhmm\)

This value is specified as the TOD time in the zAware LPAR and not as the system time where the MainView AutoOPERATOR PAS is running.

**Note:** If the time is 14:25:00, the last available interval is 14:10:00.

### **I_DEBUG**

*(optional)* 0 to turn off debugging messages, or 1 to turn on the messages.

---

The following tables list the MainView AutoOPERATOR LOCAL variables and values that are created as output after QAOZINT1 EXEC executes, and are available to QAOZINT2.

### Table 17: QAOZINT1 LOCAL output variables for summary information for the interval

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L.SYSPLEX.LPAR.SYSID</td>
<td>SYSPLEX-LPAR designation from zAware output</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.START_TIME</td>
<td>Interval start date and time</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.ASCORE</td>
<td>Anomaly score for the interval</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.MESSAGES</td>
<td>Number of message IDs</td>
</tr>
</tbody>
</table>
Table 18: QAOZINT1 LOCAL output variables for detailed information for each message ID

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L.SYSPLEX.LPAR.MSGID.n</td>
<td>Message ID</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.NUMINS.n</td>
<td>Number of instances of the message ID</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.ASCORE.n</td>
<td>Anomaly score associated with each message that is reported during the interval</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.CLID.n</td>
<td>Number of message IDs</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.CLSTAT.n</td>
<td>Cluster status</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.CRWDS.n</td>
<td>Critical words from the message</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.BERN.n</td>
<td>Bernouli value</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.POISS.n</td>
<td>Poisson value</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.TXSUMM.n</td>
<td>Summary of the common message text</td>
</tr>
<tr>
<td>L.SYSPLEX.LPAR.TXSAMP.n</td>
<td>Text of the first message occurrence</td>
</tr>
</tbody>
</table>
AlarmPoint solution

MainView AutoOPERATOR provides a pair of sample EXECs that you can use to issue notification requests to AlarmPoint, which is a product from AlarmPoint Systems.

If you own AlarmPoint, you can use this sample solution to add notification capabilities to your MainView AutoOPERATOR automation. Refer to the AlarmPoint publication, *AlarmPoint User Manual* for more information about the notification methods that are available with the AlarmPoint product.

*Note*
If you are using an earlier version of the AlarmPoint product, BMC currently continues to provide support for the AlarmPoint solution that supports the earlier version. Refer to “Migrating to version 3.0” for more information.

AlarmPoint Standard or Professional Infrastructure Edition version 3.0.x

This section describes the solution for AlarmPoint Standard or Professional Infrastructure Edition, version 3.0. x.

EXECs

MainView AutoOPERATOR provides the QAOALMP4 EXEC in the BBSAMP data set that you can copy and modify to send notifications from MainView AutoOPERATOR EXECs to AlarmPoint.

The solution for AlarmPoint Standard or Professional Infrastructure Edition, version 3.0.x and later uses the following EXECs

- BBSAMP member QAOALMP4
BBPROC member QAOALMP5

---

**Note**

This solution requires REXX Sockets which are part of the IBM TCP/IP product.

---

## Configuring the EXECs

BBSAMP member QAOALMP4 is a sample EXEC that demonstrates what you need to do to invoke EXEC QAOALMP5.

QAOALMP5 is the EXEC that actually formats and sends the notification request to AlarmPoint. In most cases, you should be able to use this EXEC without modification.

**To send a notification:**

1. Make a copy of BBSAMP member QAOALMP4 and store it in UBBPROC.
2. Follow the comments in the EXEC to modify your copy of the EXEC.

You can also incorporate the logic in QAOALMP4 in your own REXX EXECs.

## Migrating to version 3.0

If you are using the EXECs QAOALMP1 and QAOALMP2 and you have upgraded your installation of AlarmPoint Standard or Professional Infrastructure Edition, version 3.0.x, you must perform these migration steps to use the new EXECs, QAOALMP4 and QAOALMP5.

The QAOALMP4 EXEC is a simplified version of the QAOALMP1 EXEC and eliminates the type variable.

In addition, although the QAOALMP2 EXEC communicates directly with the AlarmPoint Server, this option is no longer possible with version 3.0.x. The replacement EXEC, QAOALMP5, communicates with the AlarmPoint Java Agent.

Note: The earlier versions of the AlarmPoint sample solution that were delivered prior to this release of MainView AutoOPERATOR allowed you to specify that notifications were directed to either an AlarmPoint ID or a user-specified telephone number. Because notifying a telephone number prevents you from taking advantage of the advanced capabilities in AlarmPoint version 3.0.x, the updated solution supports notifying only an AlarmPoint ID.
To migrate to update the AlarmPoint solution, perform the following steps.

   
   Ensure that the Java Agent is always running by installing it as a Microsoft Windows service because the MainView AutoOPERATOR EXECs communicate with this agent.

2. Review your automation EXECs and identify where you have customized the QAOALMP1 EXEC (or copied its logic into one of your own EXECs).

3. Search for all EXECs that invoke the QAOALMP2 EXEC.

4. Ensure that you have MainView AutoOPERATOR version 6.5.xx installed and you have applied PTF BPO8451.

5. Review the comments in BBSAMP member QAOALMP4 to understand the revised invocation requirements.

6. Convert all of the existing EXECs to match the QAOALMP4 EXEC.

7. Be sure to remove the type variable and modify all occurrences of QAOALMP2 to QAOALMP5.

8. Update the QAOALMP4 EXEC to specify the IP address or host name of the AlarmPoint version 3.0.x Java Agent.

9. Test the EXECs with AlarmPoint version 3.0.x and verify that the notifications are correctly generated.

Note

When you are troubleshooting any notification problems, ensure that you view both the Java Agent logs and the AlarmPoint Server logs for informational and error messages.

AlarmPoint version 6.x and earlier

The EXECs documented in this sections are not compatible if you have installed AlarmPoint version 3.0.x and later.

If you have installed AlarmPoint version 3.0.x or later, you must use the EXECs documented in "AlarmPoint Standard or Professional Infrastructure Edition, version 3.0.x".
EXECs

MainView AutoOPERATOR provides a sample EXEC, QAOALMP1, that you can copy and modify to send notifications from MainView AutoOPERATOR EXECs to AlarmPoint.

The following lists the EXECs for the AlarmPoint solution:

- BBSAMP member QAOALMP1
- BBPROC member QAOALMP2

This solution requires REXX Sockets which are part of the IBM TCP/IP product.

Configuring the EXECs

BBSAMP member QAOALMP1 is a sample EXEC that demonstrates what you need to do to invoke EXEC QAOALMP2.

QAOALMP2 is the EXEC that actually formats and sends the notification request to AlarmPoint. In most cases, you should be able to use this EXEC without modification.

To send a notification:

1. Make a copy of BBSAMP member QAOALMP1 and store it in UBBPROC.

2. Follow the comments in the EXEC to modify your copy of the EXEC.

   You can also incorporate the logic in QAOALMP1 in your own REXX EXECs.
JES2 solution

The Job Entry Subsystem (JES) is critical to the processing flow of your MVS environment. You might have jobs that execute under the control of JES that are critical to your data center.

BMC provides the JES2 monitoring solution, which alerts data center personnel of any JES2-related problems.

The Job Entry Subsystem is a critical MVS subsystem. If JES2 is not functioning correctly, the throughput on the MVS system is adversely affected. The JES2 monitoring solution ensures that JES2 system problems are recognized immediately.

Variables

For this solution to function properly in your environment, you must establish values for the following variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSPROG</td>
<td>TSO user ID of the primary system programmer who is to receive warning messages</td>
</tr>
<tr>
<td>SYSJES</td>
<td>version, release, and modification level of JES in the form of SP n.n.n</td>
</tr>
<tr>
<td>SYSMVS</td>
<td>version, release, and modification level of MVS in the form of SP n.n.n</td>
</tr>
</tbody>
</table>

See “MVS solution variables” on page 355 for information about initializing variables and default settings.

Invocation

This solution is invoked by Rules for the following JES2 messages:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HASP050</td>
<td>JES2 Resource Shortage</td>
</tr>
</tbody>
</table>
Flow process

When any of the following messages are received, an ALERT is issued with an alarm.

If a value is specified for the SYSPROG variable, a message is sent to the system programmer's TSO user ID.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HASP093</td>
<td>nn% Spool Utilization (JES2 V2)</td>
</tr>
<tr>
<td>$HASP095</td>
<td>JES2 Catastrophic Error Or Abend</td>
</tr>
<tr>
<td>$HASP646</td>
<td>nn% Spool Utilization (JES2 V3)</td>
</tr>
</tbody>
</table>

For $HASP355, the command to drain the initiators is issued.

EXECs

The EXECs for the JES2 monitoring solutions are listed in the following table.

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HASP050</td>
<td>handles $HASP050 message</td>
</tr>
<tr>
<td>$HASP093</td>
<td>handles $HASP093 message</td>
</tr>
<tr>
<td>$HASP095</td>
<td>handles $HASP095 message</td>
</tr>
<tr>
<td>$HASP355</td>
<td>handles $HASP355 message</td>
</tr>
<tr>
<td>$HASP646</td>
<td>handles $HASP646 message</td>
</tr>
</tbody>
</table>
Rules

The Rules for the JES2 monitoring solution are listed in the following table.

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HASP050</td>
<td>text id $HASP050</td>
</tr>
<tr>
<td>$HASP093</td>
<td>text id $HASP093</td>
</tr>
<tr>
<td>$HASP095</td>
<td>text id $HASP095</td>
</tr>
<tr>
<td>$HASP355</td>
<td>text id $HASP355</td>
</tr>
<tr>
<td>$HASP646</td>
<td>text id $HASP646</td>
</tr>
</tbody>
</table>

The Rules are distributed DISABLED in RULESET AAORULM1. You must ENABLE these Rules to implement this solution.
Storage subsystems solutions

This section describes automation solutions for storage activity.

Shared DASD control solution

You must issue commands on each system in the shared DASD configuration to change the status of a volume.

This can also require switching consoles.

The DASD and tape subsystems can be just as important to your data center's performance as the processor. The commands and replies used to communicate with the storage subsystems are not always easy to use. Shared DASD environments present an even greater challenge.

The shared DASD control solution allows the status of a shared DASD volume to be changed using one command. The status change is automatically propagated to the other systems in the configuration.

Variables

For this solution to function properly in your environment, you must establish values for the variables in the following table:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSN</td>
<td>number of MVS systems in the shared DASD configuration</td>
</tr>
<tr>
<td>SYS1-SYN</td>
<td>names of the primary MainView AutoOPERATOR subsystems on each MVS system in the shared DASD configuration</td>
</tr>
</tbody>
</table>

See “MVS solution variables” on page 355 for information about initializing the variables and default settings.
Invocation

This solution is operator-invoked.

To change the status of a shared DASD volume, on the COMMAND line, enter %XSYSVARY P1 P2 P3

where

<table>
<thead>
<tr>
<th>Positional parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>is the UCB address</td>
</tr>
<tr>
<td>P2</td>
<td>is the desired status (for example, online or offline)</td>
</tr>
<tr>
<td>P3</td>
<td>should be SHR for 3480s</td>
</tr>
</tbody>
</table>

Flow process

When the operator requests that the status of a device be changed, an EXEC is scheduled to execute on each system in the shared DASD complex.

The scheduled EXEC issues either the VARY DEV,ONLINE or VARY DEV,OFFLINE command depending on the operator specification.

EXECs

The EXECs for the shared DASD control solution are listed in the following table.

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSYSVARY</td>
<td>Schedule EXECs on all systems.</td>
</tr>
<tr>
<td>MST001C</td>
<td>Issue VARY commands.</td>
</tr>
</tbody>
</table>

Rules

There are no rules for the shared DASD control solution.
**TLMS solution**

Some data centers dedicate a console to the TLMS INQR task so the operator need
not search constantly for the outstanding reply number using the DR,L command.

This solution lets TLMS commands be entered from the MCS console without
knowing the outstanding reply number.

**Variables**

This solution does not require any values to be set for variables.

**Invocation**

This solution is operator-invoked.

To issue a TLMS command, on the **COMMAND** line, enter `%TLMS P1 P2`

where

<table>
<thead>
<tr>
<th>Positional parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>P1</em></td>
<td>is the TLMS command; valid values are DV, DVA,</td>
</tr>
<tr>
<td></td>
<td>DVH, DVL, DVM, and DVR</td>
</tr>
<tr>
<td><em>P2</em></td>
<td>is the volume serial number</td>
</tr>
</tbody>
</table>

**Flow process**

The RESOLVE REPLIES command is used to determine the outstanding reply
number for the CAT2291D Message ID.

When the outstanding reply number is found, the TLMS command is issued using
the command and volume specified by the operator.

**EXECs**

The EXEC for the TLMS solution is listed in the following table.
## EXEC name

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLMS</td>
<td>finds outstanding reply, and issues the command</td>
</tr>
</tbody>
</table>

## Rules

There are no Rules to be set for the TLMS solution.

## Storage reply solution

The DASD and tape storage subsystems issue WTORs that require operator intervention.

If these are not replied to in a timely manner, allocation queues can back up and cause degradation of system throughput. This solution ensures that the storage WTORs are replied to correctly and immediately.

## Variables

This solution does not require any values to be set for variables.

## Invocation

This solution is invoked by Rules for the messages listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC701D</td>
<td>M ddd, Volume To Be Labeled ser</td>
</tr>
<tr>
<td>IEF238D</td>
<td>Reply Device Name, Wait or Cancel</td>
</tr>
<tr>
<td>IEF433D</td>
<td>Wait Requested - Reply Hold or Nohold</td>
</tr>
</tbody>
</table>

## Flow process

The following table shows the replies generated automatically for each WTOR.
Table 19: WTOR generated replies

<table>
<thead>
<tr>
<th>WTOR ID</th>
<th>Description</th>
<th>Reply</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC701D</td>
<td>M ddd, Volume To Be Labeled ser</td>
<td>M</td>
</tr>
<tr>
<td>IEF238D</td>
<td>Reply Device Name, Wait or Cancel</td>
<td>Wait</td>
</tr>
<tr>
<td>IEF433D</td>
<td>Wait Requested - Reply Hold or Nohold</td>
<td>Nohold</td>
</tr>
</tbody>
</table>

EXECs

The EXECs for the storage reply solution are listed in the following table.

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC701D</td>
<td>manages IEC701D message</td>
</tr>
<tr>
<td>IEF238D</td>
<td>manages IEF238D message</td>
</tr>
<tr>
<td>IEF433D</td>
<td>manages IEF433D message</td>
</tr>
</tbody>
</table>

Rules

The Rules for the storage reply solution are listed in the following table.

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC701D</td>
<td>text ID IEC701D</td>
</tr>
<tr>
<td>IEF238D</td>
<td>text ID IEF238D</td>
</tr>
<tr>
<td>IEF433D</td>
<td>text ID IEF433D</td>
</tr>
</tbody>
</table>

The Rules are distributed DISABLED in RULESET AAORULM1. You must enable these Rules to implement this solution.
Monitoring solutions

This section describes monitoring solutions that are designed to assist data center personnel in monitoring events occurring in the system.

Dump data sets monitoring solution

This solution helps the system automatically manage its system dump data sets.

It attempts to keep a minimum number of dump data sets free at all times.

Variables

For this solution to function properly in your environment, you must establish values for the variables in the following table:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUMPCLER</td>
<td>name of task to clear dump data sets</td>
</tr>
<tr>
<td>DUMPMF</td>
<td>number of dump data sets to keep clear</td>
</tr>
<tr>
<td>DUMPNDS</td>
<td>total number of dump data sets</td>
</tr>
</tbody>
</table>

See “MVS solution variables” on page 355 for information about initializing the variables and default settings.

Invocation

This solution is invoked by Rules for the messages listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEA911E</td>
<td>Complete/Partial Dump on SYS1.DUMPnn</td>
</tr>
</tbody>
</table>
### Message ID

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEA994A</td>
<td>All Dump Data Sets Are Full And No SVC Dumps Can Be Taken</td>
</tr>
<tr>
<td>IEA994E</td>
<td>All Allocated SYS1.DUMP Data Sets Are Full</td>
</tr>
</tbody>
</table>

This solution can also be operator-invoked. To clear the oldest dump data set, on the **COMMAND** line, enter

`%@DUMPCLRO`

### Flow process

The following processing checks for a value specification for the variable DUMPCLER.

If a value is not specified, processing ends.

*When the IEA911E Complete/Partial Dump on SYS1.DUMPnn message is received, the current number of free dump data sets is compared to the minimum number that should be kept free. If the number of free data sets is less than the minimum, a task to clear the oldest dump data set is started.*

When either of the following messages is received, a task to clear the oldest dump data set is started:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEA994A</td>
<td>All Dump Data Sets Are Full And No SVC Dumps Can Be Taken</td>
</tr>
<tr>
<td>IEA994E</td>
<td>All Allocated SYS1.DUMP Data Sets Are Full</td>
</tr>
</tbody>
</table>

### EXECs

The EXECs for the dump data sets monitoring solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@DMPCLRO</td>
<td>starts task to clear oldest dump data set</td>
</tr>
<tr>
<td>IEA911E</td>
<td>compares minimum free with current free data sets</td>
</tr>
<tr>
<td>IEA994A</td>
<td>invokes @DMPCLRO</td>
</tr>
<tr>
<td>IEA994E</td>
<td>invokes @DMPCLRO</td>
</tr>
</tbody>
</table>
Rules

The Rules for the dump data sets monitoring solution are listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEA911E</td>
<td>text ID IEA911E</td>
</tr>
<tr>
<td>IEA994A</td>
<td>text ID IEA994A</td>
</tr>
<tr>
<td>IEA994E</td>
<td>text ID IEA994E</td>
</tr>
</tbody>
</table>

The Rules are distributed DISABLED in RULESET AAORULM1. You must ENABLE these Rules to implement this solution.

WTO buffers monitoring solution

This solution lets the system automatically resolve WTO buffer shortages.

Variables

For this solution to function properly in your environment, you must establish values for the following variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSPROG</td>
<td>TSO user ID of primary system programmer to receive warning messages</td>
</tr>
<tr>
<td>SYSMVS</td>
<td>Version, release, and modification level of the MVS system in use</td>
</tr>
</tbody>
</table>

See “MVS solution variables” on page 355 for information about initializing the variables and default settings.

In addition to setting variables, the parameter CMDCON must be specified in member BBISSP00 of the BBPARM data set. This specification is required because the K Q command used to clear buffers cannot be issued from a subsystem console. Specifying CMDCON causes all commands issued without response to be issued with the CMDCON console ID.
**Invocation**

This solution is invoked by Rules for the messages listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEA404A</td>
<td>Severe WTO Buffer Shortage - 100% Full</td>
</tr>
<tr>
<td>IEA405E</td>
<td>Severe WTO Buffer Shortage - 80% Full</td>
</tr>
</tbody>
</table>

**Flow process**

When either the IEA404A or the IEA405E buffer shortage message is received, an ALERT is generated.

If a value was specified for SYSPROG, a warning message is sent to the primary system programmer.

The D C command is issued to determine the number of buffers in use by each console. The buffers are deleted by using one of the following commands:

- K Q,L=console ID: clearing console buffers
- WRITELOG: clearing syslog buffers

After the shortage is relieved, the ALERT is deleted. If a value is specified for SYSPROG, a message is sent to notify the primary system programmer that the shortage has been relieved.

**EXECs**

The EXECs for the WTO buffers monitoring solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEA404A</td>
<td>Generate ALERTs; invoke IEE249I/IEE889I</td>
</tr>
<tr>
<td>IEA405E</td>
<td>Generate ALERTs; invoke IEE249I/IEE889I</td>
</tr>
<tr>
<td>IEE249I</td>
<td>Clear buffers (before MVS SP4)</td>
</tr>
<tr>
<td>IEE889I</td>
<td>Clear buffers (SP4 and later)</td>
</tr>
</tbody>
</table>
Rules

The Rules for the WTO buffers monitoring solution are listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEA404A</td>
<td>text ID IEA404A</td>
</tr>
<tr>
<td>IEA405E</td>
<td>text ID IEA405E</td>
</tr>
</tbody>
</table>

The Rules are distributed DISABLED in RULESET AAORULM1. You must ENABLE these Rules to implement this solution.

SMF data set monitoring solution

This solution manages dumping and switching a system to automatically determine which other system in the shared DASD configuration is preventing access to a shared device.

Variables

For this solution to function properly in your environment, you must establish values for the following variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMFALT</td>
<td>Suffix of alternate SMF parameters</td>
</tr>
<tr>
<td>SMFCLEAR</td>
<td>Task name to dump SMF data sets</td>
</tr>
</tbody>
</table>

See “MVS solution variables” on page 355 for information about initializing the variables and default settings.

Invocation

This solution is invoked by Rules for the SMF messages listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEE361I</td>
<td>SMF Data Lost - No Data Sets Available</td>
</tr>
<tr>
<td>IEE362A</td>
<td>SMF Enter Dump For SYS1.MANx On ser</td>
</tr>
<tr>
<td>Message ID</td>
<td>Message text</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>IEE364I</td>
<td>SMF (Logical/Physical) Error On SYS1.MANx</td>
</tr>
<tr>
<td>IEE366I</td>
<td>No SMF Data Sets Available - Data Being Buffered</td>
</tr>
<tr>
<td>IEE391I</td>
<td>SMF Enter Dump for Data Set on VOLSER ser, DSN=dsname</td>
</tr>
<tr>
<td>IEE392I</td>
<td>SMF Enter Dump for Data Set on VOLSER ser, DSN=dsname</td>
</tr>
<tr>
<td>IEE393I</td>
<td>SMF(LOGICAL</td>
</tr>
</tbody>
</table>

### Flow process

The processing flow for the SMF data set monitoring solution is described as follows:

When the IEE362A, IEE391A or IEE392I SMF Enter Dump For SYS1.MANx ON SER message is received, the task to dump the data set (SMFCLEAR) is started. If a value was not specified for the SMFCLEAR variable, no processing occurs.

When the IEE949I or IEE974I message (output from D SMF command) is received, the task to dump the data set (SMFCLEAR) is started if it is not already active. If a value was not specified for the SMFCLEAR variable, no processing occurs.

When the IEE361I, IEE364I, IEE393I or IEE366I messages are received, the primary system programmer is notified if a value was specified for the SYSPROG variable.

If a value was specified for variable SMFALT, the operator receives an ALERT requesting that the alternate SMF parameters be switched. If the operator confirms the switch, the T SMF command is issued. The primary system programmer is notified of the switch if a value was specified for the SYSPROG variable.

### EXECs

The EXECs for the SMF data set monitoring solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEE361I</td>
<td>manages IEE361I message</td>
</tr>
<tr>
<td>IEE362A</td>
<td>manages IEE362A message</td>
</tr>
<tr>
<td>IEE364I</td>
<td>manages IEE364I message</td>
</tr>
<tr>
<td>IEE366I</td>
<td>manages IEE366I message</td>
</tr>
<tr>
<td>IEE391I</td>
<td>manages IEE391I message</td>
</tr>
<tr>
<td>IEE392I</td>
<td>manages IEE392I message</td>
</tr>
</tbody>
</table>
## Rules

The Rules for the SMF data set monitoring solution are listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEE361I</td>
<td>text ID IEE361I</td>
</tr>
<tr>
<td>IEE362A</td>
<td>text ID IEE362A</td>
</tr>
<tr>
<td>IEE364I</td>
<td>text ID IEE364I</td>
</tr>
<tr>
<td>IEE366I</td>
<td>text ID IEE365I</td>
</tr>
<tr>
<td>IEE391I</td>
<td>text ID IEE391I</td>
</tr>
<tr>
<td>IEE392I</td>
<td>text ID IEE392I</td>
</tr>
<tr>
<td>IEE393I</td>
<td>text ID IEE393I</td>
</tr>
<tr>
<td>IEE949I</td>
<td>text ID IEE949I</td>
</tr>
<tr>
<td>IEE974I</td>
<td>text ID IEE974I</td>
</tr>
</tbody>
</table>

The Rules are distributed DISABLED in RULESET AAORULM1. You must enable these Rules to implement this solution.

## LOGREC data set monitoring solution

This solution lets the system automatically manage the SYS1.LOGREC data set.

## Variables

For this solution to function properly in your environment, you must establish values for the following variables:
Variable name | Description
---|---
LOGREC | Task name to clear SYS1.LOGREC data set
SYSPROG | TSO user ID of primary system programmer to receive warning messages

See “MVS solution variables” on page 355 for information about initializing the variables and default settings.

**Invocation**

This solution is invoked by a Rule for the messages listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFB040I</td>
<td>SYS1.LOGREC Area Is Full</td>
</tr>
<tr>
<td>IFB060E</td>
<td>SYS1.LOGREC Near Full</td>
</tr>
<tr>
<td>IFB070I</td>
<td>Logrec Cannot Be Accessed. Record Is Lost</td>
</tr>
</tbody>
</table>

**Flow process**

When either the IFB040I or the IFB060E LOGREC FULL message is received and a value was specified for variable LOGREC, the following actions are taken:

- The task to clear SYS1.LOGREC is started.
- An ALERT is issued.
- If a value was specified for variable SYSPROG, a message is sent to the system programmer.

When the IFB070I Logrec Cannot Be Accessed message is received, the following actions are taken:

- An ALERT is issued.
- If a value was specified for variable SYSPROG, a notification is sent to the system programmer.
EXECS

The EXECs for the LOGREC data set monitoring solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFB040I</td>
<td>handles IFB040I message</td>
</tr>
<tr>
<td>IFB060E</td>
<td>handles IFB060E message</td>
</tr>
<tr>
<td>IFB070I</td>
<td>handles IFB070I message</td>
</tr>
</tbody>
</table>

Rules

The Rules for the LOGREC data set monitoring solution are listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFB040I</td>
<td>text ID IFB040I</td>
</tr>
<tr>
<td>IFB060E</td>
<td>text ID IFB060E</td>
</tr>
<tr>
<td>IFB070I</td>
<td>text ID IFB070I</td>
</tr>
</tbody>
</table>

The Rules are distributed DISABLED in RULESET AAORULM1. You must enable these Rules to implement this solution.

RMF monitoring solution

This solution issues a MainView AutoOPERATOR ALERT when RMF ends.

Variables

This solution does not require any values to be set for variables.

Invocation

This solution is invoked by a Rule for the following message:
### Message ID

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERB102I</td>
<td>sid: Terminated</td>
</tr>
</tbody>
</table>

### Flow process

When the ERB102I message is received, an ALERT is issued.

### EXECs

The EXEC for the RMF monitoring solution is listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERB102I</td>
<td>handles ERB102I message</td>
</tr>
</tbody>
</table>

### Rules

The Rule for the RMF monitoring solution is distributed DISABLED in RULESET AAORULM1. You must ENABLE this Rule to implement this solution.

You must enable this Rule to implement this solution.

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERB102I</td>
<td>text ID ERB102I</td>
</tr>
</tbody>
</table>
CONTROL-M solution

This solution provides a command interface to CONTROL-M.

The operator uses an MVS MODIFY command to communicate with the CONTROL-M job scheduling software. This solution provides a command interface to CONTROL-M that alleviates the need for the MODIFY command.

Variables

This solution does not require you to set values for any variables.

Invocation

This solution is operator-invoked.

To invoke, enter

%SCHEDCOM P1

on the COMMAND line to issue a Control-M command, where

<table>
<thead>
<tr>
<th>Positional Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Enter a Control-M command; valid values CTMX004 and NEWDEST.</td>
</tr>
</tbody>
</table>

Flow processing

The following command is issued by using the input parameter as the Control-M command:

F ControlM, cmd
EXECs

The EXEC for the Control-M solution is listed in the following table.

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEDCOM</td>
<td>issues the MODIFY command</td>
</tr>
</tbody>
</table>

Rules

The Control-M solution has no Rules.
Job scheduling solution

This solution consists of time-initiated EXECs that are scheduled at BBI-SS PAS startup time.

These EXECs must be modified to contain the commands to start the time dependent task. You can use this solution to assist data center personnel in scheduling and managing time-dependent tasks.

Variables

There are no variables for this solution.

Invocation

AAORULM1 contains disabled Rule PM00101.

When this Rule Set and Rule ID are enabled at PAS initialization, this solution is invoked.

The appropriate EXECs must be modified with the necessary commands to start the various tasks.

Flow process

AAOPRM xx is updated to enable Rule Set AAORULM1 and PM00101 is enabled at a cold start.

The PAS is restarted with either the Rule Set and Rule already enabled from a previous invocation, or a cold start is done with AAORULM1 listed in AAOPRM xx.
The following EXECs can be used for the Job Scheduling solution:

<table>
<thead>
<tr>
<th>EXECs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM00101</td>
<td>scheduled from the PM0010I Rule in Rule Set AAORULM1</td>
</tr>
<tr>
<td>MSU002C</td>
<td>scheduled from within PM0010I at PAS initialization</td>
</tr>
<tr>
<td>MSU005C</td>
<td>scheduled from within MSU002C and schedules the various EXECs that set</td>
</tr>
<tr>
<td></td>
<td>timer EXECs</td>
</tr>
<tr>
<td>MJSU001C</td>
<td>starts daily events</td>
</tr>
<tr>
<td>MJSU002C</td>
<td>starts Sunday events</td>
</tr>
<tr>
<td>MJSU003C</td>
<td>starts Monday events</td>
</tr>
<tr>
<td>MJSU004C</td>
<td>starts Tuesday events</td>
</tr>
<tr>
<td>MJSU005C</td>
<td>starts Wednesday events</td>
</tr>
<tr>
<td>MJSU006C</td>
<td>starts Thursday events</td>
</tr>
<tr>
<td>MJSU007C</td>
<td>starts Friday events</td>
</tr>
<tr>
<td>MJSU008C</td>
<td>starts Saturday events</td>
</tr>
</tbody>
</table>

The Rule for the Job Scheduling solution is as follows:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM00101</td>
<td>this Rule is distributed disabled in Rule Set AAORULM1. You need to enable</td>
</tr>
<tr>
<td></td>
<td>this Rule to implement this solution.</td>
</tr>
</tbody>
</table>
MVS performance management solutions

The performance management solutions are designed to assist data center personnel monitor and adjust system parameters that affect system performance.

Load balancing solution

As workload type and activity on your system changes, you might want to adjust your job initiator configuration.

For example, you might want to take advantage of times when the system resources support additional initiators without adversely affecting system performance. Or you might want to limit access to the system at times when resources are constrained.

This solution lets the system automatically increase or decrease its workload (initiators) based on current system performance and job demand.

Variables

For this solution to function properly in your environment, you must establish values for the following variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBGPAGE</td>
<td>paging rate at which an attempt is made to decrease workload</td>
</tr>
<tr>
<td>LBGCPUH</td>
<td>CPU utilization percentage at which an attempt is made to decrease workload</td>
</tr>
<tr>
<td>LBGCPUL</td>
<td>CPU utilization percentage at which an attempt is made to increase workload</td>
</tr>
<tr>
<td>LBGIBEG</td>
<td>beginning initiator number to manage</td>
</tr>
<tr>
<td>LBGIEND</td>
<td>ending initiator number to manage</td>
</tr>
<tr>
<td>Variable name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>LBGIINC</td>
<td>number of initiators to start in an increase situation</td>
</tr>
<tr>
<td>LBGIDEC</td>
<td>number of initiators to stop in an decrease situation</td>
</tr>
<tr>
<td>LBGCLAS</td>
<td>class priority list from high to low</td>
</tr>
</tbody>
</table>

See “MVS solution variables” on page 355 for information about initializing the variables and default settings.

**Invocation**

This solution is operator-invoked.

To invoke, enter

```
%LBSTART
```

on the COMMAND line to begin load balancing on your system.

**Flow process**

Every five minutes, this solution compares the current CPU utilization and paging rate to the threshold variables.

If any threshold is crossed 3 times within 15 minutes, an action is taken.

If CPU utilization is below the low CPU threshold, an attempt is made to increase the workload by starting additional initiators.

Initiators (up to the maximum specified by variable LBGIINC) that have a status of either DRAINED or HALTED are started. Classes are assigned to the initiators based on the class priority list (variable LBGCLAS) and work waiting to execute.

If the CPU utilization is above the high CPU threshold or the page rate is above the page rate threshold, an attempt is made to decrease the workload by stopping initiators. Initiators (up to the maximum specified by variable LBGIDEC) that have a status of either active or INACTIVE are purged or drained.

**EXECs**

The following EXECs are for the load balancing solution:
<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBSTART</td>
<td>initializes CPU and page threshold counters, calls MPE003C to build initiator table, schedules MPE004C to execute in five minutes</td>
</tr>
<tr>
<td>MPE003C</td>
<td>retrieves initiator information from JES</td>
</tr>
<tr>
<td>MPE004C</td>
<td>manages CPU and page threshold counters, schedules MPE005C to execute</td>
</tr>
<tr>
<td>MPE005C</td>
<td>compares current CPU and page values against thresholds (if adjustment is necessary), and schedules MPE006C to execute or schedules MPE004C to execute in five minutes</td>
</tr>
<tr>
<td>MPE006C</td>
<td>starts/stops initiators and schedules MPE004C to execute again in five minutes</td>
</tr>
</tbody>
</table>

**Rules**

There are no Rules for the load balancing solution.

**DASD reserve analysis solution**

In shared DASD environments, one or more systems can be prevented from accessing an entire DASD volume due to hardware reserves by another system.

If a reserve remains in effect for an extended period of time, end-user response time can be severely affected.

This solution lets the system automatically determine which other system in the shared DASD configuration is preventing access to a shared device.

**Variables**

For this solution to function properly in your environment, you must establish values for the following variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSN</td>
<td>number of MVS systems in the shared DASD configuration</td>
</tr>
<tr>
<td>SYS1-SYSn</td>
<td>names of the primary MainView AutoOPERATOR subsystem on each MVS system in the shared DASD configuration</td>
</tr>
</tbody>
</table>

See “MVS solution variables” on page 355 for information about initializing the variables and default settings.
Invocation

This solution is invoked by a Rule for the following IOS message:

IOS071I UCB,CHPID,JOBNAME,START PENDING

Flow process

When the IOS071I message is received, a remote EXEC is scheduled on every other system in the shared DASD complex.

Each of the other systems use the RESOLVE RESERVE command to examine the DASD reserves it is currently holding. Any system that has a device reserved issues an operator ALERT back to the system that is being prevented access to the device.

EXECs

The EXECs for the DASD reserve analysis solution are

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOS071I</td>
<td>schedules remote EXECs</td>
</tr>
<tr>
<td>MPE001C</td>
<td>examines currently held reserves</td>
</tr>
<tr>
<td>MPE002C</td>
<td>issues ALERT</td>
</tr>
</tbody>
</table>

Rules

The Rule for the DASD reserve analysis solution is

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOS071I</td>
<td>text ID IOS071I</td>
</tr>
</tbody>
</table>

The Rule is distributed DISABLED in RULESET AAORULM1. You must enable this Rule to implement this solution.
Exception monitoring solution

If you use the MainView SYSPROG Services product, you can extend its Advanced Early Warning (AEW) capabilities.

This solution lets RESOLVE AEW messages be posted to the MainView AutoOPERATOR ALERT application and, in some instances, provides information about the causes of an exception condition.

Variables

For this solution to function properly in your environment, you must establish values for the following variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXCCPU</td>
<td>maximum complex CPU utilization percentage</td>
</tr>
<tr>
<td>MAXTCPU</td>
<td>maximum CPU utilization by TSO address spaces</td>
</tr>
<tr>
<td>MAXBCSA</td>
<td>maximum CSA utilization percentage below the 16M line</td>
</tr>
<tr>
<td>MAXACSA</td>
<td>maximum extended CSA utilization percentage</td>
</tr>
<tr>
<td>MAXDEVU</td>
<td>maximum device utilization percentage</td>
</tr>
</tbody>
</table>

See “MVS solution variables” on page 355 for information about initializing the variables and default settings.

Invocation

You must have the MainView SYSPROG Services product installed is required to use the MVS performance monitoring solution. To invoke the exception monitoring solution:

1. ENABLE the following Rules in the AAORULM1 Rule Set:

   - PWSCPU00 *WARNING* CPU 1 USAGE IS (x)%
   - PWSCPU01 *WARNING* CPU COMPLEX USAGE IS (x)%

   and enable the following Rules:

   - PWSCPU02 *WARNING* CPU USAGE IS (x)% FOR TSO
To install these solutions

a. Activate the AEW sampler.

COPY <PREFIX>.BBPARM(PWSCPMZZ) TO <PREFIX>.UBBPARM,DISP=SHR

b. Add the following statement to your BBI-SS JCL to activate the PWSCPM00 Rule.

//LIB DD DSN=<PREFIX>.UBBPARM,DISP=SHR

Restart your BBI subsystem after making those changes.

3. Set the variables for these solutions in <PREFIX>.UBBPROC member MSU002C.

Additional information is provided in MSU002C and PWSCPM00.

4. ENABLE the Rules in RULESET AAORULM1.

If you have only one CPU in the system in which you are running this BBI subsystem, ENABLE Rule PWSCPU00.

If there are multiple CPUs running, ENABLE Rule PWSCPU01.

Issue a RESOLVE CPU command to check the number of CPUs running in your system.

**Flow process**

These two Rules schedule EXECs that use the same variables in EXEC MSU002C and are mutually exclusive.

If either of the PWSCPU messages is received three or more times within 10 minutes, the CPU value in the message is compared against the appropriate CPU threshold variable. If the threshold is exceeded, an ALERT is issued and an additional monitor ($TOTCPU or $TSOCPU, depending on which threshold is exceeded) is invoked.

The $TOTCPU and $TSOCPU EXECs issue the RESOLVE CPU command to determine who is the current major user of the CPU ($TSOCPU limits the search to TSO users only). The EXECs then issue ALERTs to give the operator insight into which users are responsible for the exception condition.
When either the PWSCSA01 or PWSDEV01 message is received, the value in the message is compared against the appropriate threshold variable. If the threshold is exceeded, an ALERT is issued.

**EXECs**

The EXECs for the exception monitoring solution are listed in the following table.

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWSCPU00</td>
<td>monitors MAXCCPU threshold</td>
</tr>
<tr>
<td>PWSCPU01</td>
<td>monitors MAXCCPU threshold</td>
</tr>
<tr>
<td>PWSCPU02</td>
<td>monitors MAXTCPU threshold</td>
</tr>
<tr>
<td>PWSCSA01</td>
<td>monitors MAXBCSA and MAXACSA thresholds</td>
</tr>
<tr>
<td>PWSDEV01</td>
<td>monitors MAXDEVU threshold</td>
</tr>
</tbody>
</table>

**Rules**

The Rules for the exception monitoring solution are listed in the following table.

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWSCPU00</td>
<td>text ID PWSCPU00</td>
</tr>
<tr>
<td>PWSCPU01</td>
<td>text ID PWSCPU01</td>
</tr>
<tr>
<td>PWSCPU02</td>
<td>text ID PWSCPU02</td>
</tr>
<tr>
<td>PWSCSA01</td>
<td>text ID PWSCSA01</td>
</tr>
<tr>
<td>PWSDEV01</td>
<td>text ID PWSDEV01</td>
</tr>
</tbody>
</table>

The Rules are distributed DISABLED in RULESET AAORULM1. You must enable these Rules to implement this solution.
Error recovery solutions

Critical messages from CICS can be lost among the message traffic at the operator console.

Also, replies to those messages and the follow-up actions taken might not be consistent. The error recovery solutions assist data center personnel when critical errors occur in the CICS environment.

Storage violation solution

The storage violation solution notifies an operator when a storage violation has occurred.

Variables

For the storage violation solution to function properly in your environment, you must specify values for the variables listed in the following table:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCALPT</td>
<td>SSID of MainView AutoOPERATOR subsystem to receive all ALERTs</td>
</tr>
<tr>
<td>ONLSYSN</td>
<td>job name of the target system</td>
</tr>
<tr>
<td>ONLCNTN</td>
<td>subsystem ID of the controlling BBI-SS</td>
</tr>
<tr>
<td>ONLALTN</td>
<td>color for ALERT messages (can be BLUE, WHITE, RED, or GREEN)</td>
</tr>
<tr>
<td>ONLTYPPN</td>
<td>type of target or BBI-SS (can be CICS, DB2®, IMS, CICSAO, or IMSAO)</td>
</tr>
<tr>
<td>ONLALMN</td>
<td>indicates whether an alarm sounds (can be Y or N)</td>
</tr>
</tbody>
</table>

See “CICS solution variables” on page 357 for information on default settings and initializing the variables.
Invocation

This solution is invoked by the Rule for the following text ID:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0508</td>
<td>A STORAGE VIOLATION HAS OCCURRED</td>
</tr>
</tbody>
</table>

Flow process

The processing flow depends on the operator's reply to an ALERT.

When the DFH0508 message is received, the ALERT *STORAGE VIOLATION IN CICS cicsid, REPLY YES TO CANCEL is sent to ask the operator to confirm cancellation of the CICS region that suffered the storage violation.

If the operator replies yes, the region is canceled and this message is sent: *STORAGE VIOLATION IN CICS cicsid, BEING CANCELED NOW.

To modify the solution to automatically cancel the region without asking for operator confirmation, edit BBPROC member DFH0508. When automatic cancellation is in effect, the ALERT BEING CANCELED NOW is sent.

EXECs

The EXECs for the storage violation solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0508</td>
<td>handles DFH0508 message</td>
</tr>
<tr>
<td>CER001C</td>
<td>cancels the CICS system after a storage violation</td>
</tr>
</tbody>
</table>

Rules

The Rules for the storage violation solution are listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0508</td>
<td>text ID DFH0508</td>
</tr>
<tr>
<td>DFHSM010</td>
<td>text ID DFHSM010</td>
</tr>
</tbody>
</table>
The Rules are distributed DISABLED in RULESET AAORULC1. You must ENABLE these Rules to implement this solution. You must ENABLE DFHSM010 for CICS/ESA.

**CICS abnormal termination solution**

The CICS abnormal termination solution notifies an operator when an abend in a CICS region has been detected.

**Variables**

For the CICS abnormal termination solution to function properly in your environment, you must establish values for the following variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCALPT</td>
<td>SSID of MainView AutoOPERATOR subsystem to receive all ALERTs</td>
</tr>
<tr>
<td>ONLSYSN</td>
<td>job name of the target system</td>
</tr>
<tr>
<td>ONLCNTN</td>
<td>subsystem ID of the controlling BBI-SS</td>
</tr>
<tr>
<td>ONLALTN</td>
<td>color for ALERT messages (can be BLUE, WHITE, RED, or GREEN)</td>
</tr>
<tr>
<td>ONLTYPN</td>
<td>type of target or BBI-SS (can be CICS, DB2, IMS, CICSAO, or IMSAO)</td>
</tr>
<tr>
<td>ONLALMN</td>
<td>indicates whether an alarm sounds (can be Y or N)</td>
</tr>
</tbody>
</table>

See “CICS solution variables” on page 357 for information on default settings and initializing the variables.

**Invocation**

This solution is invoked by the Rule for the text ID listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0606</td>
<td>ABEND xxxx - xxxx HAS BEEN DETECTED</td>
</tr>
</tbody>
</table>
Flow process

In response to the DFH0606 message, an ALERT is sent to the FOCAL POINT target to indicate a catastrophic abend ended a CICS system.

EXECs

The EXEC for the CICS abnormal termination solution is listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0606</td>
<td>handles DFH0606 message</td>
</tr>
</tbody>
</table>

Rules

The Rule for the CICS Abnormal Termination is listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0606</td>
<td>text ID DFH0606</td>
</tr>
</tbody>
</table>

The Rule is distributed DISABLED in RULESET AAORULC1. You must enable this Rule to implement this solution.

VSAM subtask abnormal termination solution

The VSAM subtask abnormal termination solution automatically notifies an operator when a VSAM subtask abends.

Variables

For the VSAM subtask abnormal termination solution to function properly in your environment, you must establish values for the following variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCALPT</td>
<td>SSID of MainView AutoOPERATOR subsystem to receive all ALERTs</td>
</tr>
<tr>
<td>ONLSYSN</td>
<td>job name of the target system</td>
</tr>
</tbody>
</table>
### Variable name

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONLCNTN</td>
<td>subsystem ID of the controlling BBI-SS</td>
</tr>
<tr>
<td>ONLALTN</td>
<td>color for ALERT messages (can be BLUE, WHITE, RED, or GREEN)</td>
</tr>
<tr>
<td>ONLTYPN</td>
<td>type of target or BBI-SS (can be CICS, DB2, IMS, CICSAO, or IMSAO)</td>
</tr>
<tr>
<td>ONLALMN</td>
<td>indicates whether an alarm sounds (can be Y or N)</td>
</tr>
</tbody>
</table>

See “CICS solution variables” on page 357 for information on default settings and initializing the variables.

### Invocation

This solution is invoked by the Rule for the text ID listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0901</td>
<td>VSAM SUBTASK ABEND - DO YOU WANT TO CONTINUE IN A DEGRADED MODE OR ABEND? REPLY GO OR CANCEL.</td>
</tr>
</tbody>
</table>

This solution cannot be used for CICS/ESA.

### Flow process

When the DFH0901 WTOR is received, this solution replies GO to let CICS continue initialization. the following ALERT message is sent:

In addition, the following ALERT message is sent:*VSAM SUBTASK ABEND, CICS CONTINUING IN DEGRADED MODE*.

To modify the solution to reply CANCEL instead of GO, edit BBPROC member DFH0901. When automatic cancel is in effect, the solution replies CANCEL to the DFH0901 WTOR and issues the ALERT

*VSAM SUBTASK ABEND, CICS BEING CANCELLED AUTOMATICALLY*.

### EXECs

The EXEC for the VSAM subtask abnormal termination solution is listed in the following table:
**Rules**

The Rule for the VSAM subtask abnormal termination solution is listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0901</td>
<td>text ID DFH0901</td>
</tr>
</tbody>
</table>

The Rule is distributed DISABLED in RULESET AAORULC1. You must enable this Rule to implement this solution.

**CICS temporary storage suspensions solution (CICSTSS)**

This solution detects and automatically corrects suspensions of CICS temporary storage processing caused when auxiliary temporary storage data set is full.

When that is the case, CICS suspends processing for many transactions—not only transactions that are writing to temporary storage directly but many transactions (including system functions) that use temporary storage indirectly are also affected.

For example, the EXEC CICS START command with the DATA parameter causes CICS to use temporary storage to hold the data being passed to the Started Task (STC).

**Variables**

This solution does not use any specific variable requiring special initialization.

However, variables describing the CICS systems, as explained in “CICS solution variables” on page 357, should be set.
Invocation

The EXECs in this solution are scheduled when the following message is received:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT068*</td>
<td>NNNNNN TEMPORARY STORAGE SUSPENSIONS HAVE OCCURRED</td>
</tr>
</tbody>
</table>

Flow process

All EXECs in this solution are message-driven.

The EXECs create ALERT messages that may require an operator reply to initiate a follow-up EXEC, or may include extended help or follow-up commands. The IND column of the Alert Detail Display shows which options are available for each ALERT.

If an ALERT message has an E in the IND column, a follow-up EXEC is assigned. The message text (or extended help panel) explains what reply is requested. You should enter the message text in the LC column of the ALERT display to schedule the follow-on EXEC.

If an ALERT message has a H in the IND column, an Extended Help Panel is assigned. You should type the EXPAND command and press Enter (or use the EXPAND PFK, if assigned) after placing the cursor anywhere in the ALERT message text to display the help panel.

If an ALERT message has a C in the IND column, a follow-up Primary Command is assigned. You should type the TRANSFER command and press Enter (or use the TRANSFER PFK, if assigned) after placing the cursor anywhere in the ALERT message text to invoke the command. The follow-up command transfers you to a display with more information to help solve the problem.

EXECs

The EXECs for the CICSTSS solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT068</td>
<td>starts follow-up of temporary storage suspensions by scheduling CER004C and setting a timer for CER003C</td>
</tr>
</tbody>
</table>
## EXEC name  | Description
--- | ---
CER003C | executes the MainView for CICS Problem Display and looks for the FT068 message. If found, it compares the count in the message to the count saved in a variable. If the count has increased, the investigation EXECs are kicked off.
CER004C | executes the MainView for CICS Task Display and purges any tasks waiting on temporary storage processing (ATSP)
CER005C | executes the MainView for CICS Temporary Storage Unit Table (TSUT) display and purges some Auxiliary TSUT entries. It first purges any entries that start with CEBR, which typically contain transaction dump output. If the TSP suspension persists, on the next iteration, CER005C purges the largest TSUT entry. The largest entry is determined by examining the Data Length and PUTQ count fields.
CER006C | handles the reply from the ALERT created by CER004C

### Rules

Each of the messages that is listed in the Invocation section has one Rule.

The Rule-IDs equal the message-IDs. The Rules are distributed DISABLED in RULESET AAORULC1. You must enable the following Rules to implement this solution:

- CICSTRT
- CICSTERM
- FT426W
- FT425W
- FT435I

### Temporary storage (TS) data set extension failure solution

The temporary storage (TS) data set extension failure solution notifies a central operator and provides a follow-up display for investigating the problem.
Variables

For the temporary storage data set extension failure solution to function properly in your environment, you must establish values for the following variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCALPT</td>
<td>SSID of MainView AutoOPERATOR subsystem to receive all ALERTs</td>
</tr>
<tr>
<td>ONLSYSN</td>
<td>job name of the target system</td>
</tr>
<tr>
<td>ONLCNTN</td>
<td>subsystem ID of the controlling BBI-SS</td>
</tr>
<tr>
<td>ONLALTN</td>
<td>color for ALERT messages (can be BLUE, WHITE, RED, or GREEN)</td>
</tr>
<tr>
<td>ONLTYPN</td>
<td>type of target or BBI-SS (can be CICS, DB2, IMS, CICSAO, or IMSAO)</td>
</tr>
<tr>
<td>ONLALMN</td>
<td>indicates whether an alarm sounds (can be Y or N)</td>
</tr>
</tbody>
</table>

See “CICS solution variables” on page 357 for information on default settings and initializing the variables.

Invocation

This solution is invoked by the Rule for the text ID listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH1311</td>
<td>TEMPORARY STORAGE DATA SET IS FULL AND CANNOT BE EXTENDED</td>
</tr>
</tbody>
</table>

Flow process

In response to DFH1311, an ALERT is issued. In addition, the MainView for CICS TEMPSTOR command is associated with the ALERT.

The operator can use the command to map the temporary storage usage.

EXECs

The EXEC for the temporary storage data set extension failure solution is listed in the following table:
### Rules

The Rule for the temporary storage data set extension failure solution is listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH1311</td>
<td>text ID DFH1311</td>
</tr>
</tbody>
</table>

The Rule is distributed DISABLED in RULESET AAORULC1. You must enable this Rule to implement this solution.

### Terminal errors solution

The terminal errors solution resets the terminal status after a terminal error occurs.

### Variables

For the terminal errors solution to function properly in your environment, you must establish values for the following variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCALPT</td>
<td>SSID of MainView AutoOPERATOR subsystem to receive all ALERTs</td>
</tr>
<tr>
<td>ONLSYSN</td>
<td>job name of the target system</td>
</tr>
<tr>
<td>ONLCNTN</td>
<td>subsystem ID of the controlling BBI-SS</td>
</tr>
<tr>
<td>ONLALTN</td>
<td>color for ALERT messages (can be BLUE, WHITE, RED, or GREEN)</td>
</tr>
<tr>
<td>ONLTPYN</td>
<td>type of target or BBI-SS (can be CICS, DB2, IMS, CICSAO, or IMSAO)</td>
</tr>
<tr>
<td>ONLALMN</td>
<td>indicates whether an alarm sounds (can be Y or N)</td>
</tr>
</tbody>
</table>

See “CICS solution variables” on page 357 for information on default settings and initializing the variables.
Invocation

This solution is invoked by the Rule for the following text ID listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH3437I</td>
<td>termid tranid time NODE netname ACTION TAKEN: action1 action2... action8</td>
</tr>
</tbody>
</table>

Flow process

In response to DFH3437I, the terminal status is reset as shown in the following list:

- **CEMT**: NOCREATE is set to CREATE
- **OUTSERV** is set to INSERV.

After 10 retries, VTAM commands are entered to reset the node prior to attempting CEMT. After 15 retries, recovery is stopped and this ALERT is sent:

*TERM term NODE node STATUS status AFTER 15 RETRIES

EXECs

The EXECs for the terminal errors solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH3437I</td>
<td>handles DFH3437I message</td>
</tr>
<tr>
<td>CER002C</td>
<td>resets the terminal retry counter</td>
</tr>
</tbody>
</table>

Rules

The Rules for the terminal errors solution are listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH3437I</td>
<td>text ID DFH3437I</td>
</tr>
<tr>
<td>DFHZC343</td>
<td>text ID DFHZC3437</td>
</tr>
</tbody>
</table>
The Rules are distributed DISABLED in RULESET AAORULC1. You must ENABLE DFH3437I to implement this solution. ENABLE DFHZC343 for CICS/ESA.
CICS performance management solutions

The CICS performance management solutions help data center personnel achieve maximum availability and increased response time.

Checklist for using the CICS solutions

Use this checklist to identify the steps you must complete to implement the CICS solutions.

1. Start the MainView for CICS monitors. Refer to the MainView for CICS Monitors Guide for more information.


3. Enable the individual Rules which are part of the solution that you implement.

4. Copy the CSUINIT EXEC from the BBPROC library into the UBBPROC library and edit the CSUINIT EXEC (in the UBBPROC library) to set variables values used during operation of the solutions.

5. Create (or activate) an initialization Rule that enables the solution.

File degradation analysis solution

Most CICS response time problems are caused by I/O performance problems.

High I/O service times, as reported by the MainView for CICS Background Problem Service, are presented on various displays and logged to the BBI-SS Journal.
Constant investigation of I/O problems can be very time consuming. Further, because most I/O contention problems are transient, factors causing a problem seem to disappear before an investigation is started.

This solution automatically analyzes the factors causing disk I/O performance problems that affect CICS response time.

**Note**

This solution requires that the MainView SYSPROG Services product is also installed.

### Variables

For the file degradation analysis solution to function properly in your environment, you must establish values for the variables listed in the following table:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCALPT</td>
<td>SSID of MainView AutoOPERATOR subsystem to receive all ALERTs</td>
</tr>
<tr>
<td>ONLSYSN</td>
<td>job name of the target system</td>
</tr>
<tr>
<td>ONLCNTN</td>
<td>subsystem ID of the controlling BBI-SS</td>
</tr>
<tr>
<td>ONLALTN</td>
<td>color for ALERT messages (can be BLUE, WHITE, RED, or GREEN)</td>
</tr>
<tr>
<td>ONLTPYN</td>
<td>type of target or BBI-SS (can be CICS, DB2, IMS, CICSAO, or IMSAO)</td>
</tr>
<tr>
<td>ONLALMN</td>
<td>indicates whether an alarm sounds (can be Y or N)</td>
</tr>
</tbody>
</table>

See “CICS solution variables” on page 357 for information on default settings and initializing the variables.

### Invocation

This solution is invoked by a Rule for the text ID listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT095W</td>
<td>FILE filename EXCEEDED SERVICE TIME, n.nnn IS AVERAGE SERVICE TIME</td>
</tr>
</tbody>
</table>
Flow process

When the FT095W message is received, contention for logical and physical resources in the system where the target CICS is executing is analyzed using MainView for CICS and SYSPROG services.

EXECs also are scheduled to look for any contention from other systems in the shared DASD environments. Results of the analysis are written in the MainView AutoOPERATOR subsystem (SS) log.

All results messages written to the MainView AutoOPERATOR SS log are prefixed with a message-ID. The message-ID consists of a fixed literal (CAF#), followed by the EXEC sequence number (IMFEID); for example, CAF#1234.

Use the message-ID to search for all messages from a given file analysis in the log, even when they arrive from different systems at different times and are intermixed with other messages in the SS log.

The following messages are written to the MainView AutoOPERATOR SS log:

```
STARTING FILE DEGRADATION ANALYSIS FOLLOWING FT095W
FOR FILE filename IN CICS SYSTEM cicsid (smfid) 
ON VOLUME volser ON UNIT ucb WITH nnnnn EXCPS.

DATASET dsname IS ALSO ALLOCATED TO JOB(S):
jobname1, jobname2, .... ON SYSTEM smfid.

NO OTHER JOBS ON SYSTEM smfid ARE USING DATASET dsname 
IN CICS SYSTEM cicsid.

DEVICE ucb IS LESS THAN n% BUSY ON SYSTEM smfid.

DEVICE ucb WAS OBSERVED nnn% BUSY ON SYSTEM smfid 
WITH AN I/O RATE OF nnn PER SECOND 
AND AN AVERAGE WAIT TIME OF n.nnn MS.

nnn% OF THE ACTIVITY ON ucb IS FROM JOB jobname.

DEGRADATION ANALYSIS COMPLETE FOR FILE filename 
ON SYSTEM smfid, RESULTS FROM OTHER SYSTEM(S) 
WILL FOLLOW IN THE LOG WITH THE SAME MESSAGE ID.
```
**EXECs**

The EXECs for the file degradation analysis solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT095W</td>
<td>as primary EXEC, schedules all secondary CPEnnnx EXECs that follow</td>
</tr>
<tr>
<td>CPE001C</td>
<td>searches for the DD name corresponding to the filename when the file is an IMS database</td>
</tr>
<tr>
<td>CPE002C</td>
<td>reports on other jobs allocated to the CICS file data set name</td>
</tr>
<tr>
<td>CPE003C</td>
<td>reports on other jobs using the device where the CICS file is located</td>
</tr>
<tr>
<td>CPE004C</td>
<td>gets the data set name, volser number, and UCB for the CICS filename using the SYSPROG T10T command</td>
</tr>
</tbody>
</table>

**Rules**

The Rule for the file degradation analysis solution is listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT095W</td>
<td>text ID FT095W</td>
</tr>
</tbody>
</table>

The Rule is distributed DISABLED in RULESET AAORULC1. In addition to this Rule, the following Rules must be ENABLED to implement this solution:

- CICSTART
- CICSTERM
- FT426W
- FT425W
- FT435I

**File allocation - deallocation solution**

The batch window (the interval between the time CICS is stopped and started again) might be too short to accommodate the necessary batch jobs to extract, update, and back up files used by CICS.
If the window is too small, the operator must enter commands to close, disable, and free, and then later allocate, enable, and open files. This takes time, is tedious, and is prone to error. Syntax errors or a missed step in a complicated procedure can cause resources to be unavailable for use by CICS transactions.

This solution lets batch jobs allocate and deallocate files from a running CICS system. This lets files be freed from CICS, processed by other jobs, and returned to CICS, while CICS remains available for other users.

**Variables**

For the file allocation - deallocation solution to function properly in your environment, you must establish values for the variables listed in the following table:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCALPT</td>
<td>SSID of MainView AutoOPERATOR subsystem to receive all ALERTs</td>
</tr>
<tr>
<td>ONLSYSN</td>
<td>job name of the target system</td>
</tr>
<tr>
<td>ONLCNTN</td>
<td>subsystem ID of the controlling BBI-SS</td>
</tr>
<tr>
<td>ONLALTN</td>
<td>color for ALERT messages (can be BLUE, WHITE, RED, or GREEN)</td>
</tr>
<tr>
<td>ONLTYPN</td>
<td>type of target or BBI-SS (can be CICS, DB2, IMS, CICSAO, or IMSAO)</td>
</tr>
<tr>
<td>ONLALMN</td>
<td>indicates whether an alarm sounds (can be Y or N)</td>
</tr>
</tbody>
</table>

See “CICS solution variables” on page 357 for information on default settings and initializing the variables.

**Invocation**

Most aspects of this solution are invoked from batch JCL.

Edit two sample JCL members distributed in the BBUSER data set to specify file names and CICS region names.

<table>
<thead>
<tr>
<th>Sample JCL member name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAOALLOC</td>
<td>allocate, enable, and open a file in a CICS system from a batch job</td>
</tr>
<tr>
<td>CAODEALO</td>
<td>close, disable, and free a file in a CICS system from a batch job</td>
</tr>
</tbody>
</table>
The remaining aspects of this solution are invoked by Rules for the following text IDs:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT401E</td>
<td>ALLOC FAILED RC=XX EC=XX FILE filename DSN dsname</td>
</tr>
<tr>
<td>FT402I</td>
<td>ALLOC COMPLETE FOR FILE filename DSN dsname</td>
</tr>
<tr>
<td>FT403E</td>
<td>DEALLOC FAILED NOT ALLOCATED FILE filename</td>
</tr>
<tr>
<td>FT404I</td>
<td>DEALLOC COMPLETE FOR FILE filename</td>
</tr>
</tbody>
</table>

**Flow process**

When the batch job CAOALLOC is run, the file you specified is allocated, enabled, and opened on the CICS system that you specified.

When the batch job CAODEALO is run, the file you specified is closed, disabled, and freed on the CICS system that you specified.

**EXECs**

The EXECs for the file allocation-deallocation solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPEALLOC</td>
<td>allocates, enables, and opens a file in a CICS system</td>
</tr>
<tr>
<td>CPEDEALO</td>
<td>closes, disables, and deallocates a file in a CICS system</td>
</tr>
<tr>
<td>FT401E</td>
<td>stores the result from IMFEXEC ALLOC in a shared variable</td>
</tr>
<tr>
<td>FT402I</td>
<td>stores the result from IMFEXEC ALLOC in a shared variable</td>
</tr>
<tr>
<td>FT403E</td>
<td>stores the result from IMFEXEC FREE in a shared variable</td>
</tr>
<tr>
<td>FT404I</td>
<td>stores the result from IMFEXEC FREE in a shared variable</td>
</tr>
</tbody>
</table>

**Rules**

The Rules for the file allocation-deallocation solution are listed in the following table:
### VSAM control area split monitor solution

VSAM control area splits can cause I/O response times to be extended.

This solution assists the operator in monitoring which files are suffering control area splits.

### Variables

For the VSAM control area split monitor solution to function properly in your environment, you must establish values for the variables in the following table:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCALPT</td>
<td>SSID of MainView AutoOPERATOR subsystem to receive all ALERTs</td>
</tr>
<tr>
<td>ONLSYSN</td>
<td>job name of the target system</td>
</tr>
<tr>
<td>ONLCNTN</td>
<td>subsystem ID of the controlling BBI-SS</td>
</tr>
<tr>
<td>ONLALTN</td>
<td>color for ALERT messages (can be BLUE, WHITE, RED, or GREEN)</td>
</tr>
<tr>
<td>ONLTYPN</td>
<td>type of target or BBI-SS (can be CICS, DB2, IMS, CICSAO, or IMSAO)</td>
</tr>
<tr>
<td>ONLALMN</td>
<td>indicates whether an alarm sounds (can be Y or N)</td>
</tr>
</tbody>
</table>

The Rules are distributed DISABLED in RULESET AAORULC1. In addition to these Rules, you must enable the following Rules:

- CICSTART
- CICSTERM
- FT426W
- FT425W
- FT435I
See “CICS solution variables” on page 357 for information on default settings and initializing the variables.

**Invocation**

This solution is invoked by a Rule for the text ID listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT092S</td>
<td>FILE filename HAS HAD nnn Control AREA SPLITS</td>
</tr>
</tbody>
</table>

**Flow process**

This solution sends an ALERT in response to the FT092S message.

The text of the ALERT tells the operator what to do about a VSAM file with several control area splits: *

```
FT092S FILE file SHOULD BE RE-DEFINED (nnn C.A. SPLITS)
```

In addition, the MainView for CICSFILEXPND command is associated with the ALERT. The operator uses the command to display the status of the VSAM cluster.

**EXECs**

The EXEC for the VSAM control area split monitor solution is listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT092S</td>
<td>handles FT092S message</td>
</tr>
</tbody>
</table>

**Rules**

The Rule for the VSAM control area split monitor solution is listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT092S</td>
<td>text ID FT092S</td>
</tr>
</tbody>
</table>
The Rule is distributed DISABLED in RULESET AAORULC1. In addition to this Rule, you must enable the following Rules:

- CICSTART
- CICSTERM
- FT426W
- FT425W
- FT435I

Transaction response time monitor solution

This solution ensures that operators are aware of transactions exceeding service level objectives.

Variables

For the Transaction Response Time Monitor solution to function properly in your environment, you must establish values for the variables in the following table:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCALPT</td>
<td>SSID of MainView AutoOPERATOR subsystem to receive all ALERTs</td>
</tr>
<tr>
<td>ONLSYSN</td>
<td>job name of the target system</td>
</tr>
<tr>
<td>ONLCNTN</td>
<td>subsystem ID of the controlling BBI-SS</td>
</tr>
<tr>
<td>ONLALTN</td>
<td>color for ALERT messages (can be BLUE, WHITE, RED, or GREEN)</td>
</tr>
<tr>
<td>ONLTYPEPN</td>
<td>type of target or BBI-SS (can be CICS, DB2, IMS, CICSAO, or IMSAO)</td>
</tr>
<tr>
<td>ONLALMN</td>
<td>indicates whether an alarm sounds (can be Y or N)</td>
</tr>
</tbody>
</table>

See “CICS solution variables” on page 357 for information on default settings and initializing the variables.

Invocation

This solution is invoked by a Rule for the text ID listed in the following table:
Flow process

In response to the FT094W message, the ALERT message is sent:

*SERVICE LEVELS EXCEEDED, ENTER ON THIS MSG TO SEE ACTIVITY

In addition, the MainView for CICS MONITOR command is associated with the ALERT. The operator can use the command to display service level information in the CICS region.

EXECs

The EXEC for the transaction response time monitor solution is listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT094W</td>
<td>handles FT094W message</td>
</tr>
</tbody>
</table>

Rules

The Rule for the transaction response time monitor solution is listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT094W</td>
<td>text ID FT094W</td>
</tr>
</tbody>
</table>

The Rule is distributed DISABLED in RULESET AAORULC1. In addition to this Rule, you must enable the following Rules

- CICSTART
- CICSTERM
- FT426W
- FT425W
The MainView AutoOPERATOR for CICS option provides the CAODTAB facility that allows the CICS system programmer to limit the volume of messages processed by MainView AutoOPERATOR.

This function is important because it reduces the amount of overhead that is necessary to perform automation in the CICS region. The CAODTAB is a CSECT that is linked to Transient Data exit CMRTDPXn and contains user settings that indicate:

- whether Extrapartition or Intrapartition transient data queues should be considered for automation
- the type of tables specified for Extrapartition and Intrapartition queues: include or exclude tables

Include means that all of the queues specified are included for automation and exclude means that all the queues are excluded for automation for the respective queue type that can be Extrapartition or Intrapartition.

- which transient data queues should have the above specifications applied and are then considered or rejected from automation

Refer to the chapter, "MainView AutoOPERATOR for CICS Optional Tailoring" in the MainView AutoOPERATOR Customization Guide for more information about using the CAODTAB facility.

**Note**

This solution requires that the MainView for CICS and MainView SYSPROG Services products are both installed in the same BBI-SS PAS as this solution.

---

**CAODEBUG EXEC implementation**

Use the BBPROC CAODEBUG EXEC to help determine the settings that were used to create the CAODTAB table.

This EXEC uses the MainView for CICS and SYSPROG Services products to find and display the CAODTAB table contents.

You should use the CAODEBUG EXEC
- when you have a performance problem in CICS and suspect that the transient
data exit is processing more messages than is required for the automation being performed

Use this solution to see which queues MainView AutoOPERATOR is processing and which queues are not being processed. You can then determine if you need to make changes to the CAODTAB table to remove unwanted queues and reassemble and relink the table.

- when a transient data message from CICS is not being processed by a MainView AutoOPERATOR Rule (because the Rule is not firing) and you believe that CAODTAB was created to handle the queue to which that message is destined

With this solution, you can determine if the queue in question is actually processed.

- when you are having a problem with a Rule not firing and you do not know if you have a CAODTAB created

This solution tells you whether or not a CAODTAB table exists.

The CAODEBUG EXEC accepts several parameters that are described in the following table but only the CICS region name (specified with R parameter) is a required parameter.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
</table>
| R( region)     | specifies the CICS Region name  
                    This is a required parameter. |
| P( ptf)        | (optional) specifies the PTF level of CMRTDPXn module (for example, BPOxxxx)  
                    You can also specify BASExxxx is also allowed, where xxxx = AutoOPERATOR release level (6400, 6500, and so on).  
                    The default value is the current PTF level of CMRTDPXn. |
| O( offset)     | (optional) specifies the offset to the CAODTAB table at the end of CMRTDPXn  
                    This parameter is not often needed you might need to use it when the PTF level (specified with P parameter) does not produce the correct results. You must contact BMC Software customer support to receive this offset value. |
| Status         | (optional) specifies that status messages are produced in the BBI journal during processing |
| TRACE          | (optional) specifies that the REXX TRACE command is to be used |
| TRACEI         | (optional) specifies that the REXX TRACE I command is to be used |
| HELP           | (optional) specifies that Help information will be written to the BBI Journal  
                    In the event that there is missing required input, the HELP output is automatically sent to the BBI Journal.
Following are some examples of invoking the CAODEBUG EXEC with different parameters and the resulting output.

Example 1: This example shows how to use the HELP parameter to get help with the solution.

```
%CAODEBUG HELP
CAODEBUG: This EXEC displays the contents of the CAODTAB
CAODEBUG: table in the BBI journal. It requires the use
CAODEBUG: of the SYSPROG Services DUMP command and
CAODEBUG: MainView for CICS commands.
CAODEBUG: Possible inputs (in any order) are:
CAODEBUG: R() - CICS region name, required
CAODEBUG: P() - PTF level of CMRTDPXn (default BPO7428)
CAODEBUG: O() - Offset to CAODTAB table (used only if
CAODEBUG: use of PTF level does not produce correct
CAODEBUG: results).
CAODEBUG: Status - issue informational and status msgs
CAODEBUG: Trace - use REXX TRACE R command
CAODEBUG: Tracei - use REXX TRACE I command
CAODEBUG: Example - %CAODEBUG R(CICSPROD)
CAODEBUG: Example - %CAODEBUG R(CICSPROD) P(BPO7427) STATUS
CAODEBUG: Terminating.
```

Example 2: This example shows the result when you use the CAODEBUG EXEC and specify only the region name parameter.

```
%CAODEBUG R(AAOCT31J)
%CAODEBUG R(AAOCT31J)
CAODEBUG: Table Specifications:
CAODEBUG: -----------------------
CAODEBUG: Extrapartition messages are processed.
CAODEBUG: Intrapartition messages are processed.
CAODEBUG: An INCLUDE table for Extrapartition queues.
CAODEBUG: An EXCLUDE table for Intrapartition queues.
CAODEBUG: Blank-delimited list of queues in CAODTAB:
CAODEBUG: -----------------------------------------
CAODEBUG: TEST CSMT CADL INTR QUE1 QUE2 QUE3 QUE4 QUE5 QUE6 QUE7
CAODEBUG: QUE8 QUE9 QU10 QU11 QU12 QU13 QU14 QU15 QU16 QU17 QU18
CAODEBUG: QU19 QU20 QU21 QU22 QU23 QU24 QU25 QU26 QU27 QU28
CAODEBUG: Terminating.
```

Example 3: This example uses the region name and the status parameters to show detailed processing information.

```
%CAODEBUG R(AAOCT31J) STATUS
CAODEBUG: Checking for PTF level and offset.
CAODEBUG: PTF level for CMRTDPXn = BPO7428.
CAODEBUG: Table offset in CMRTDPXn = 00000B0C.
CAODEBUG: Checking for CICS region name.
CAODEBUG: CICS region name = AAOCT31J.
CAODEBUG: Issuing QUERY to get program attributes.
CAODEBUG: Pgm = CMRTDPX5, Address = 00064000, Length = 3,016.
CAODEBUG: Table address = 00064B40.
CAODEBUG: Storage dumped by SYSPROG Services.
CAODEBUG: Flag bytes = 0152.
CAODEBUG: Formatting table data...
CAODEBUG: Table Specifications:
```
Example 4: This example specifies a CICS region that does not contain a CAODTAB table.
VTAM NCP solution

Much of the work in data centers is performed interactively using online systems and terminal networks. However, the commands and replies used to communicate with the network software can be cumbersome to use.

BMC provides the TCAS reply solution, which provides automatic replies to TCAS WTORs.

TCAS reply solution

When TCAS ends either normally or abnormally, WTORs are issued.

If these WTORs are not replied to correctly and in a timely manner, the down time for TSO can be extended unnecessarily. This solution ensures that the TCAS WTORs are replied to correctly and immediately.

Variables

This solution does not require any values to be set for variables.

Invocation

This solution is invoked by Rules for the TCAS messages listed in the following table:

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IKT001D</td>
<td>nnn Users Active Reply U, SIC or FSTOP</td>
</tr>
<tr>
<td>IKT010D</td>
<td>nnn Users Active Reply SIC or FSTOP</td>
</tr>
<tr>
<td>IKT012D</td>
<td>TCAS termination in progress - specify U or DUMP</td>
</tr>
</tbody>
</table>
Flow process

When either of the WTORs listed in the following table is received, a reply of SIC is generated:

<table>
<thead>
<tr>
<th>WTOR ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IKT001D</td>
<td>\textit{nnn} Users Active Reply U, SIC or FSTOP \hspace{1cm} (received when TCAS is started after abnormal termination)</td>
</tr>
<tr>
<td>IKT010D</td>
<td>\textit{nnn} Users Active Reply SIC or FSTOP \hspace{1cm} (received when TCAS is stopped)</td>
</tr>
</tbody>
</table>

When the IKT012D TCAS Termination In Progress - Specify U or DUMP WTOR is received, a reply of U is generated.

EXECs

The EXECs for the TCAS reply solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IKT001D</td>
<td>Handle IKT001D message</td>
</tr>
<tr>
<td>IKT010D</td>
<td>Handle IKT010D message</td>
</tr>
<tr>
<td>IKT012D</td>
<td>Handle IKT012D message</td>
</tr>
</tbody>
</table>

Rules

The Rules for the TCAS reply solution are listed in the following table:

<table>
<thead>
<tr>
<th>Rule name</th>
<th>Fires for</th>
</tr>
</thead>
<tbody>
<tr>
<td>IKT001D</td>
<td>text ID IKT001D</td>
</tr>
<tr>
<td>IKT010D</td>
<td>text ID IKT010D</td>
</tr>
<tr>
<td>IKT012D</td>
<td>text ID IKT012D</td>
</tr>
</tbody>
</table>

The Rules are distributed DISABLED in RULESET AAORULM1. You must enable these Rules to implement this solution.
Before you can use the automation features of the DB2 solutions, you must customize your automation environment to accommodate them.

After performing the customization steps, you can implement the solutions described in the section “DB2 solution variables and parameters” on page 361.

Customizing for DB2 solutions

Complete the following procedure.

1. Customize EXECs.

Make these changes to the following EXECs:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSU101C</td>
<td>change the value specified for variable DB2 to the name of one of your DB2 subsystems</td>
</tr>
<tr>
<td>DSU101C</td>
<td>verify that the variables MSTR, DBM1, IRLM and DIST correctly identify the names of the Started Tasks for the DB2 you named</td>
</tr>
<tr>
<td>DSU101C</td>
<td>verify and change the value specified for variable ( D_&amp;DB2_DB2RELEASE ) \ This variable identifies the version of DB2 to match the database named in variable &amp;DB2. The default is 8.1.</td>
</tr>
</tbody>
</table>

For the initial installation, you can accept the default values for the rest of the variables in this EXEC.

2. Determine number of DB2 subsystems.

You can monitor up to nine different DB2 subsystems with the DB2 solutions. However, changing the number of DB2 subsystems that your BBI subsystem (BBI-SS PAS) monitors requires that you also change the settings of some DB2 variables. (See “All DB2 solutions” on page 362.)
If your BBI-SS PAS is monitoring more than one subsystem, follow these steps:

a  Set the value of n in D_DB2NUMn to the number of DB2 subsystems in DSU001C.

b  Create as many copies of EXEC DSU101C as the number of subsystems you plan to monitor and name the copies as follows:

   DSU101C, DSU102C...DSU10nC

   where n equals the value of D_DB2NUM in step one.

c  Set each copy's variable to identify a unique DB2 subsystem and its release number.

d  Verify that the following variables' values correctly identify the names of the Started Tasks for this DB2 subsystem:

   - MSTR
   - IRLM
   - DBM1
   - DIST
   - SPAS

3 Update and activate Rules.

Rules are distributed initially DISABLED with MainView AutoOPERATOR solutions. You must enable the Rules that affect the solutions you plan to use. In addition, you must update certain parameters in BBPARM to reflect the solutions Rules you enabled.

To activate and update DB2 solutions Rules, follow these steps:

a  Activate the Rules in each set of Rules for the DB2 solutions by using a text editor to change the Rule status from STATUS(DISABLED) to STATUS(ENABLED).

b  Enable the Rule for message PM0010I found in Rule Set AA0RUL00 of the MainView AutoOPERATOR product.

c  Activate the following DB2 solutions sets of Rules from the MainView AutoOPERATOR Rule Processor Detail Control panel:

   - AAORULD2
- AAORULD3
- AAORULD5
- AAORULD6
- AAORULD7
- AAORULD8

d Update the RULESCAN and RULESET parameters in BBPARM member AAOPRM00 to include the DB2 solutions Rules and make these sets of Rules permanently active.

A sample is provided in BBPARM member AAOPRMD1.

4 Add the required Time-Initiated EXEC:

a Add the monitors and time-initiated EXECs for the DB2 solutions to your BLK member, as defined in the TARGET=xxx,BLK=xxx parameter in BBPARM member BBIISP00. An example is shown in the sample that is provided in BBPARM member BBISSPD1.

b To activate these monitors and time-initiated EXECs you will need to either:

- Restart the BBI-SS PAS
- Dynamically start the BLK by using the MainView AutoOPERATOR Time-Initiated EXEC panel (option 8.8) by specifying:

  \[ BLK=DPE002B \]

  and

  \[ BLK=DPE030B \]

5 Determine threshold levels.

When running DB2 applications, DB2 informational messages appear frequently on your console screen. For the DB2RNWY solution (see “DB2RNWY threshold parameters” on page 363), the frequency of a message's appearance is directly related to any threshold levels that you set before using this solution:

a Browse hilevel.BBPARM(DMRBEX00) to review DB2RNWY threshold level parameters.

b Browse hilevel.BBPARM(DMRBEXD1) to review the BMC recommended settings for these parameters.
c Copy member DMRBEXD1 into member DMRBEX00 and set each threshold parameter in DMRBEXD1 to the value of your choice.

If you have more than one DB2 subsystem, you must copy the thresholds for each DB2 TARGET statement.

d Put these changes into effect by issuing the command

```
.E P DMRBEX00
```

from any command line.

6 Modify your ALERT profile.

To display the DB2 ALERT Queue on the MainView AutoOPERATOR ALERT Overview application, you must modify the ALERT Profile.

Use the primary command PROFILE and set the name of the ALERT Queue to match what you used for the shared variable D_ALERT_Q.
DB2 global operations solutions

The DB2 global operations solutions provide initialization routines as well as utilities and minimum alert management for DB2 subsystems.

The MainView for DB2 product is a prerequisite for several DB2 solutions as indicated in the individual solution description.

DB2 major messages (DB2ALRT solution)

During its operations, DB2 issues a number of messages with various severity levels; some of these messages require immediate attention.

The DB2ALRT solution transforms these messages into MainView AutoOPERATOR ALERTs, assigning a color to each message based on the severity level of the DB2 problem.

Variables

You do not need to set any variable values for this solution.

Invocation

This solution is invoked by Rules for these major DB2 messages:

- DSN3201I ABNORMAL EOT IN PROGRESS FOR USER= CON-ID= COR-ID=
- DSNB200I UPDATE VVDS FAILED
- DSNB204I OPEN OF DATA SET FAILED. DSNAME = dsn
- DSNB207I DYNAMIC ALLOCATION OF DATA SET FAILED. REASON = rrr DSNAME = dsn
- DSNB217I csect-name - ONLINE RECOVERY FOR AN INCONSISTENT PAGE
WAS UNSUCCESSFUL FOR DBNAME = dbn, SPACENAME = spn,
PAGE NUMBER = X'pno'

DSNB224I csect-name - BUFFER MANAGER I/O ERROR DURING function
DBNAME=dbn, ...

DSNB225I BUFFER MANAGER I/O ERROR

DSNB226I BUFFER MANAGER DETECTED INVALID PAGE

DSNB227I DFHSM RECALL FAILED

DSNB551I BSDS READ ERROR

DSNB552I BSDS WRITE ERROR

DSNB553I BSDS INSERT ERROR

DSNB601I BUFFER POOL nn FULL

DSNB602I UNABLE TO CREATE BUFFERPOOL

DSNB603I INSUFFICIENT STORAGE FOR BUFFERPOOL EXPAND/CREATE

DSNB605I INSUFFICIENT VIRTUAL STORAGE FOR BUFFERPOOL

DSNB606I INSUFFICIENT STORAGE FOR HIPERPOOL

DSNB607I UNABLE TO CREATE HIPERPOOL - NO EXPANDED

DSNB608I UNABLE TO CREATE HIPERPOOL

DSNB609I VIRTUAL BUFFERPOOL IS ZERO - DEFAULT USED

DSNB610I UNABLE TO CREATE HIPERPOOL - NO ADMF

DSNB611I HIPERPOOL DELETED - ADMF INACTIVE

DSNC001I CICS UR INDOUBT RESOLUTION IS INCOMPLETE FOR name

DSNC030E ERROR WRITING TO TRANSIENT DS

DSNC034I INDOUBT RESOLUTION FOR ur-id IS INCORRECT

DSNC035I INDOUBT RESOLUTION INCOMPLETE

DSNC036I INDOUBT RESOLUTION FOR ur-id IS INCONSISTENT

DSNC901I UNRECOVERABLE I/O ERROR IN DSNCCOM1

DSNI001I RESTART HAS BEEN DEFERRED

DSNI007I UNABLE TO DIRECT READ A LOG RECORD

DSNI010I BROKEN PAGE ACCESSED TYPE type NAME name MODNAME
csect-name CONN-ID id CORR-ID id

DSNI012I PAGE LOGICALLY BROKEN TYPE type NAME name MODNAME
modname ERQUAL erqual

DSNJ004I ACTIVE LOG COPY n INACTIVE, LOG IN SINGLE MODE, ENDRBA=...
DSNJ008E nn OF mm ACTIVE LOGS ARE FULL sname NEEDS ARCHIVE SCRATCH. REPLY YY WHEN DEVICE READY OR N TO CANCEL

DSNJ013I TERMINAL ERROR ccc IN BUFFER rrr BEFORE ACTIVE LOG WRITE

DSNJ014I TERMINAL ERROR ccc IN BUFFER rrr AFTER ACTIVE LOG WRITE

DSNJ073I LOG ARCHIVE UNIT ALLOCATION FAILURE DETECTED.
   RETURN CODE = nnnn. ALLOCATION OR OFF-LOAD OF ARCHIVE LOG DATA SET MAY FAIL

DSNJ100I csect-name ERROR OPENING BSDSn DSNAME=..., ERROR STATUS=...

DSNJ102I LOG RBA CONTENT OF LOG DATA SET DSNAME=... STARTRBA=.. ENDRBA=.. DOES NOT AGREE WITH BSDS INFORMATION

DSNJ103I LOG ALLOCATION ERROR DSNAME=dsname, ERROR STATUS=eeeeiiii

DSNJ104I csect-name RECEIVED ERROR STATUS nnn FROM macro-name FOR DSNAME dsname

DSNJ105I csect-name LOG WRITE ERROR DSNAME=..., LOGRBA=...,ERROR STATUS=ccccffss

DSNJ106I LOG READ ERROR DSNAME=..., LOGRBA=...,ERROR STATUS=ccccffss

DSNJ107I READ ERROR ON BSDS DSNAME=... ERROR STATUS=...

DSNJ108I WRITE ERROR ON BSDS DSNAME=... ERROR STATUS=...

DSNJ109I OUT OF SPACE IN BSDS DSNAME=...

DSNJ110E LAST COPYn ACTIVE LOG DATA SET IS nnn PERCENT FULL

DSNJ111E OUT OF SPACE IN ACTIVE LOG DATA SETS

DSNJ115I OFFLOAD FAILED FOR ARCHIVE

DSNJ117I INITIALIZATION ERROR READING BSDS DSNAME=..., ERROR STATUS=...

DSNJ120I DUAL BSDS DATA SETS HAVE UNEQUAL TIME STAMPS, BSDS1 SYSTEM=.... UTILITY=.... BSDS2 SYSTEM=.... UTILITY=...

DSNJ124I OFFLOAD OF ACTIVE LOG SUSPENDED FROM RBA xxxx TO RBA yyyy DUE TO I/O ERROR

DSNJ126I BSDS ERROR FORCED SINGLE MODE

DSNJ150E LOG CAPTURE EXIT ABEND

DSNL007I DDF IS ABNORMALLY TERMINATING

DSNL033I DDF TERMINATION BECAUSE OF ABEND

DSNL400E INDOUBT THREAD HEURISTIC DAMAGE
DSNL401E INDOUBT THREAD REMOTE ABORT - HEURISTIC DAMAGE
DSNL402I INDOUBT THREAD REMOTE COMMIT - HEURISTIC DAMAGE
DSNL403I INDOUBT THREAD REMOTE ABORT - HEURISTIC DAMAGE
DSNL404E PROTOCOL ERROR
DSNL405I THREAD PLACED INDOUBT
DSNL406I THREAD MAY BE INDOUBT - COMM FAILURE
DSNL408I INDOUBT THREAD HEURISTIC DAMAGE - COORDINATOR
DSNL409I INDOUBT THREAD HEURISTIC DAMAGE - COORDINATOR
DSNL411E COLD START BY COORDINATOR - MANUAL RESOLUTION
DSNL412I PROTOCOL ERROR DURING SYNCHPOINT
DSNL413I PROTOCOL ERROR DURING SYNCHPOINT
DSNL414E PROTOCOL ERROR DURING INDOUBT
DSNL420I COLD START BY PARTICIPANT - POSSIBLE DAMAGE
DSNL421I SNA XLN PROTOCOL VIOLATION
DSNL500I CONVERSATION FAILED TO LOCATION locname ...
DSNL501I CNOS PROCESSING FAILED
DSNL502I SYSTEM CONVERSATION FAILED TO LOCATION=locname ...
DSNM002I IMS/VS xxxx DISCONNECTED FROM SUBSYSTEM yyyy RC=rc
DSNM004I RESOLVE INDOUBT ENTRY(S) ARE OUTSTANDING FOR SUBSYS xxxx
DSNM005I RESOLVE INDOUBT SYNCHRONIZATION PROBLEM WITH SUBSYS xxxx
DSNP001I DSNPmmm - dsn IS WITHIN n KBYTES OF AVAILABLE SPACE
              RC=r CONNECTION-ID=id CORRELATION-ID=id
DSNP007I DSNPmmm - EXTEND FAILED FOR dsn
              RC=r CONNECTION-ID=id CORRELATION-ID=id
DSNP011I DSNPmmm - MEDIA MANAGER SERVICES ERROR FOR dsn. MMRC=C
              DSMRC=r CONNECTION-ID=id CORRELATION-ID=id
DSNP012I DSNPmmm - ERROR IN VSAM CATALOG LOCATE FUNCTION FOR dsn
              CTLGRC=r CTLGRSN=r CONNECTION-ID=id CORRELATION-ID=id
DSNP015I IRLM MANUAL UNLOCK FAILED
DSNP028I HSM RECALL FAILED
DSNT377I PLAN plan-id1 WITH CORRELATION ID id1 CONNECTION ID id2
              IS IN CONFLICT WITH AN INDOUBT THREAD
DSNT500I csect-name RESOURCE UNAVAILABLE REASON = r TYPE = t NAME = n
DSNT501I csect-name RESOURCE UNAVAILABLE CORRELATION-ID = id1 CONNECTION-ID = id2 REASON = r TYPE = t NAME = n
DSNV086E DB2 ABNORMAL TERMINATION REASON = xxxxxxxx
DXR013E ABEND UNDER IRLM TCB/SRB
DXR016E BUCK PROCESS TIME EXCEEDS 30 SECONDS
DXR019E STORAGE POOL DESTROYED
DXR021E ABEND VTAM ERROR
DXR022E SUBTASK FAILURE
DXR023E INTERNAL OR MVS ERROR
DXR024E VTAM REQUEST UNSUCCESSFUL
DXR027A SESSION LOST
DXR031E ABEND UNDER PTB SRB
DXR050I OUT OF STORAGE DETECTED DURING DEADLOCK
DSNX905I csect-name PROCEDURE OR FUNCTION name TERMINATED ABNORMALLY ASID = asid WLM_ENV = wlmenv
DSNX906I csect-name PROCEDURE OR FUNCTION name TERMINATED ABNORMALLY. THE PROCEDURE HAS BEEN STOPPED. ASID = asid WLM_ENV = wlmenv
DSNX908I csect-name PROCEDURE OR FUNCTION name WITH LOAD MODULE module EXCEEDED CPU RESOURCE LIMIT SSN = ssn PROC = jclproc ASID = asid WLM_ENV = wlmenv
DSNX921I csect-name SOME SQL CALL STATEMENTS CANNOT BE PROCESSED, BECAUSE REQUIRED RESOURCES ARE NOT AVAILABLE REASON = reason TYPE = type NAME = name
DSNX922I csect-name BEGINNING DISCONNECTION OF STORED PROCEDURE ADDRESS SPACES
DSNX923I csect-name ALL STORED PROCEDURE ADDRESS SPACES ARE NOW DISCONNECTED.
DSNX944I csect-name THE STORED PROCEDURE FUNCTION IS NOT AVAILABLE
DSNX950I csect-name PROCEDURE procname HAS NOT BEEN ACCESSED
DSNX953E csect-name STORED PROCEDURE JOBNAME OR STARTED TASK NAME DOES NOT MATCH ZPARM
DSNX954I csect-name ATTEMPT TO START DB2-Established STORED PROCEDURE ADDRESS SPACE PROC = jclproc FAILED
DSNX960E csect-name SOME SQL CALL STATEMENTS CANNOT BE PROCESSED, BECAUSE THE Language Environment ROUTINE CEEPIPI CANNOT BE LOADED SSN = ssn PROC = jclproc ASID = asid WLM_ENV = wlmenv
DSNX962I csect-name THE Language Environment CEEPIPI SERVICE RETURNED AN UNEXPECTED RETURN CODE rc FOR FUNCTION func SSN = ssn PROC = jclproc ASID = asid WLM_ENV = wlmenv
DSNX964I csect-name THE DB2-Established STORED PROCEDURES ADDRESS SPACE FOR SUBSYSTEM subs IS STARTED
DSNX965I csect-name THE DB2-Established STORED PROCEDURES ADDRESS SPACE FOR SUBSYSTEM subs IS STOPPING
DSNX966I csect-name STORED PROCEDURE OR FUNCTION sp FAILED. Language Environment ENCLAVE RC = erc, FEEDBACK INFORMATION = fi SSN = ssn PROC = jclproc ASID = asid WLM_ENV = wlmenv
DSNX967I csect-name ATTEMPT TO PERFORM WLM FUNCTION func FAILED WITH WLM RC = rc RSN = rsn SSN = ssn PROC = jclproc ASID = asid WLM_ENV = wlmenv
DSNX968I csect-name STORED PROCEDURE ADDRESS SPACE IS UNABLE TO CONNECT TO WLM BECAUSE WLM_ENV = wlmenv IS STOPPED OR QUIESCED
Flow process

When any of these messages are received, MainView AutoOPERATOR issues an ALERT and assigns a color based on the known severity of the attached message.

EXECs

The EXEC for this solution is listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPE992C</td>
<td>creates an ALERT for any message that invokes DPE992C</td>
</tr>
</tbody>
</table>

Rules

There is one Rule for each of the messages listed in the Invocation section.

The Rules are distributed DISABLED in RULESET AAORULD8. You must enable these Rules to implement this solution.

DB2 environment set-up (DB2INIT solution)

Before you use the DB2 solutions, you must provide some information about the DB2 subsystems you plan to use.

This section describes how you provide this information to your automation environment.

Variables

To use the DB2INIT solution, you must establish values for both the general DB2 variables and the DB2 subsystem variables.

The general DB2 variables are listed in the following table:
<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_ALERT_TGT</td>
<td>target to receive ALERTS</td>
</tr>
<tr>
<td>D_ALERT_Q</td>
<td>name of the ALERT queue for all ALERTS generated by the solutions</td>
</tr>
<tr>
<td>D_ALERT_IDS</td>
<td>TSO IDs to receive notification of ALERTS through TSO SEND commands</td>
</tr>
<tr>
<td>D_ALERT_SEND</td>
<td>indicates whether to issue the send command (Y/N)</td>
</tr>
<tr>
<td>D_ALERT_SENDOPT</td>
<td>MVS SEND option; either LOGON or NOW</td>
</tr>
<tr>
<td>D_DB2NUM</td>
<td>number of DB2s defined to this BBI-SS</td>
</tr>
</tbody>
</table>

Set these variables by modifying EXEC DSU001C. For a complete list of all the DB2 solutions variables, see “DB2 solution variables and parameters” on page 361

For the DB2 subsystems variables, one set of the following variables should be defined for each DB2 system you intend to control through solutions:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_DB2n</td>
<td>nth DB2 target name</td>
</tr>
<tr>
<td>D_MSTRn</td>
<td>nth DB2 MSTR address space name</td>
</tr>
<tr>
<td>D_DBM1n</td>
<td>nth DB2 DBM1 address space name</td>
</tr>
<tr>
<td>D_IRLMn</td>
<td>nth DB2 IRLM address space name</td>
</tr>
<tr>
<td>D_DISTn</td>
<td>nth DB2 DIST address space name</td>
</tr>
<tr>
<td>D_SPASn</td>
<td>nth DB2 SPAS address space name</td>
</tr>
</tbody>
</table>

Set these variables by modifying EXEC DSU10 n C, where n is the number of the associated DB2 subsystem. If you start more than one DB2 subsystem, you must duplicate the sample EXEC DSU101C provided. Other EXECS, up to DSU105C, are included with only the comments box.

If you try to display the DB2 variables names and content by using the MainView AutoOPERATOR facilities, be aware that there might be several DB2 subsystems.

**Invocation**

DB2INIT consists of the EXEC DSU001C and one (or several) of the EXECS DSU10 n C.

Schedule these EXECS to run as soon as possible after the host BBI-SS is started.
Using a Rule that fires from the PM0010I message, which is the primary MainView AutoOPERATOR message is one way to ensure that the EXECs are scheduled. The Rule should schedule a start-up EXEC and you need to add the subsequent calls to additional EXECs, listed in “Variables” on page 232, to that start-up EXEC; for example:

\[
\text{IMFEXEC SELECT EXEC(DSU001C) WAIT(Y)}
\]

**Flow process**

The DB2INIT solution is executed only once at BBI-SS start-up.

If you need to run DB2INIT dynamically (such as the first time, or after a variable change), you can schedule the processing EXECs from any MainView AutoOPERATOR workstation.

**EXECs**

The EXECs for the DB2INIT solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSU001C</td>
<td>initializes shared variables for solutions</td>
</tr>
<tr>
<td>DSU10nC</td>
<td>initializes shared variables for a particular DB2 subsystem where ( n ) is the number of DB2 subsystems monitored by this BBI-SS</td>
</tr>
</tbody>
</table>

**Rules**

There are no Rules for the DB2INIT solution.

**MainView for DB2 major messages (DMRALRT solution)**

MainView for DB2 (DMR) provides DB2 performance information for the system programmer and the Database Administrator (DBA).

The DMRALERT solution requires that the MainView for DB2 product is installed.
Several MainView for DB2 components issue messages with various severity levels. These messages have been sorted and a set of those requiring immediate attention have been selected. In fact, only the messages reported as severe exceptions have been retained, along with their associated clearing message.

The DB2 solutions were developed from a subset of those messages that are considered to be candidates for automation.

The remaining messages are listed in “Invocation” on page 235. DMRALRT transforms the messages in this set into MainView AutoOPERATOR ALERTs, assigning a color to each message based on the severity level of the problem.

**Variables**

To use the DMRALRT solution, you must establish values for both the general DB2 variables and the DB2 subsystem variables defined in the DB2INIT solution.

**Invocation**

This solution is invoked by Rules for the messages listed in the following sample:

- **DZ1010S** - INDOUBT THREAD
- **DZ1011I** - EXCEPTION CLEARED: INDOUBT THREAD
- **DZ1040S** - EDM POOL FULL FAILURES
- **DZ1041I** - EXCEPTION CLEARED: EDM POOL FULL FAILURES
- **DZ1050S** - BP(n) DM CRITICAL THRESHOLD REACHED
- **DZ1051I** - EXCEPTION CLEARED: BP(n) DM CRITICAL THRESHOLD REACHED
- **DZ1060S** - BP(n) IMMEDIATE WRITE THRESHOLD REACHED
- **DZ1061I** - EXCEPTION CLEARED: BP(n) IMMEDIATE WRITE THRESHOLD REACHED
- **DZ1070S** - BP(n) EXPANSION FAILURE, MAXPAGES REACHED
- **DZ1071I** - EXCEPTION CLEARED: BP(n) EXPANSION FAILURE, MAXPAGES REACHED
- **DZ1080S** - BP(n) EXPANSION FAILURE, VIRTUAL STORAGE SHORTAGE
- **DZ1081I** - EXCEPTION CLEARED: BP(n) EXPANSION FAILURE, VIRT. ST. SHORTAGE
- **DZ1090S** - FINAL ACTIVE LOG DATASET 75% FULL
- **DZ1091I** - EXCEPTION CLEARED: FINAL ACTIVE LOG DATASET 75% FULL
- **DZ1100S** - ACTIVE LOG REDUCED TO SINGLE MODE
Flow process

When any of the messages are received, MainView AutoOPERATOR creates an ALERT.

MainView AutoOPERATOR assigns a color to each ALERT based on the known severity of the attached message. When a clearing message is received, the ALERT is cleared.

EXECs

The EXEC for this solution is listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPE994C</td>
<td>issues or deletes the ALERT associated with any of the messages found in “Invocation” on page 235.</td>
</tr>
</tbody>
</table>

Rules

There is one Rule for each of the messages listed in the Invocation section.

The Rule-IDs equal the message-IDs.

The Rules are distributed DISABLED in RULESET AAORULD3. You must enable these Rules to implement this solution.
MainView for DB2 utilities (DMRUTIL solution)

MainView for DB2 (DMR) provides DB2 performance information for the system programmer and the Database Administrator (DBA).

With the MainView architecture, when MainView AutoOPERATOR and MainView for DB2 are installed in the same system, the products can easily access each others functions. This also allows the DMRUTIL solution to see what is happening within DB2. This communication is based on an internal exchange of a MainView for DB2 screen image in reply to a received command.

To shorten development time, MainView AutoOPERATOR provides service routines for the major MainView for DB2 services that let you access DB2 data from local variables. These routines are included in the DMRUTIL solution.

Variables

This solution does not require you to set any variables values.

Invocation

The EXECs from this solution are called by various DB2 solutions.

Therefore, any MainView for DB2 solution requires that DMRUTIL is enabled.

EXECs

The EXECs for the DB2UTIL solution are listed in the following table.

Each utility EXEC is completed by an example of how to schedule it.

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPE900C</td>
<td>DB2ST formatter - breaks DB2ST display into variables</td>
</tr>
<tr>
<td>DPE900D</td>
<td>Driver example for DPE900C and DPE901C</td>
</tr>
<tr>
<td>DPE901C</td>
<td>USERS formatter - breaks USERS display into variables</td>
</tr>
<tr>
<td>DPE901D</td>
<td>Driver example for DPE901C</td>
</tr>
<tr>
<td>DPE902C</td>
<td>Breaks DB2EX display into variables</td>
</tr>
<tr>
<td>EXEC name</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>DPE902D</td>
<td>Driver example for DPE902C</td>
</tr>
<tr>
<td>DPE905C</td>
<td>BFRPL formatter - breaks BFRPL display into variables</td>
</tr>
<tr>
<td>DPE905D</td>
<td>Driver example for DPE905C</td>
</tr>
<tr>
<td>DPE908C</td>
<td>DBTS formatter - breaks DBTS display into variables</td>
</tr>
<tr>
<td>DPE908D</td>
<td>Driver example for DPE908C</td>
</tr>
<tr>
<td>DPE908DR</td>
<td>Driver for DPE908C (written in REXX)</td>
</tr>
<tr>
<td>DPE909C</td>
<td>RIDPL Formatter</td>
</tr>
<tr>
<td>DPE909D</td>
<td>Driver for DPE909C</td>
</tr>
<tr>
<td>DPE932C</td>
<td>DLOGS formatter - breaks DLOGS display into variables</td>
</tr>
<tr>
<td>DPE932D</td>
<td>Driver for DPE932C</td>
</tr>
</tbody>
</table>

**Rules**

There are no Rules for this solution.
DB2 resource contention analysis solutions

The DB2 resource contention analysis solutions concentrate on a few major resources that are critical to DB2 operations.

DB2 table space filling up solution (DB2TFUL)

Many users access DB2 databases during the day, and some users might continue to add data until they run out of space.

When DB2 reports that all the space has been used up, it is too late to avoid an outage. DB2TFUL provides an early warning of a table space or a partition filling up.

DB2TFUL analyzes open table spaces and imposes user-specified thresholds on them. If a threshold is exceeded, an ALERT is issued.

Looking at many DB2 databases takes time and resources, so the attached analysis is run only every half-hour, through a timer request you set in MainView AutoOPERATOR.

This solution requires that MainView for DB2 is installed in the same system as MainView AutoOPERATOR.

Variables

To use DB2TFUL, you must establish values for the general DB2 variables and the DB2 subsystem variables defined in the DB2INIT solution.

Additionally, you can define values for the following thresholds:
<table>
<thead>
<tr>
<th>Threshold</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_EXTS</td>
<td>upper bound for the number of extents for a tablespace unless overridden by an exception. If zero, this threshold is ignored.</td>
</tr>
<tr>
<td>MAX_UTIL</td>
<td>upper bound for the percent utilization for a tablespace unless overridden by an exception. If zero, this threshold is ignored.</td>
</tr>
<tr>
<td>NUM</td>
<td>number of exceptions for this DB2. Must be equal to the highest suffix for variable NAME.</td>
</tr>
<tr>
<td>NAME n</td>
<td>the name of the nth exception. Must be specified as a tablespace. Can be generic.</td>
</tr>
<tr>
<td>MAX_EXTS n</td>
<td>the extent threshold for the nth exception name. If zero, this threshold is ignored.</td>
</tr>
<tr>
<td>MAX_UTIL n</td>
<td>the utilization threshold for the nth exception name. If zero, this threshold is ignored.</td>
</tr>
</tbody>
</table>

These variables should be set by reviewing EXEC DSU10 n C, attached to initializing values pertaining to each specific DB2 subsystem. These EXECs are included in the DB2INIT solution.

For example, suppose the following variables are set in EXEC DSU101C:

```
SET MAX_EXTS = 0
SET MAX_UTIL = 0
SET NUM      = 2
SET NAME1    = &STR(DSN*)
SET EXTS1    = 15
SET UTIL1    = 0
SET NAME2    = &STR(DSNDBO6_SYS*)
SET EXTS2    = 5
SET UTIL2    = 90
```

Setting both MAX_EXTS and MAX_UTIL to zero tells the solution to check only the exception thresholds.
The first exception states that any tablespace in a database starting with DSN that has more than 15 extents creates an ALERT.

The second exception states that an ALERT is generated for any tablespace starting with SYS in database DSND806 that has more than 5 extents or is over 90% utilized on the fifth extent.

**Invocation**

Usually, DB2TFUL is initiated through a time-initiated request.

A sample timer is provided in BBPARM member DPE030B. You also should create a call to this timer in your BBIISPxx member. A sample entry is provided in BBPARM member BBIISPD1.

**Flow process**

Using the timer facility described in the Invocation section, this solution is intended to run in the background and sample data periodically.

However, both DPE030C and DPE031C can be executed from any COMMAND line of any MainView screen display by using the command character percent sign (%).

**EXECs**

The EXECs for the DB2TFUL solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPE030C</td>
<td>threshold analysis of the DBTS display for a particular DB2</td>
</tr>
<tr>
<td>DPE031C</td>
<td>EXEC that calls EXEC DPE030C for each defined DB2</td>
</tr>
</tbody>
</table>

**Rules**

There are no Rules for this solution.
DB2 thread control solution (DB2THRD)

DB2 offers its users a number of access paths, referred to as threads.

DSNZPARM specifies the maximum number of threads DB2 can have. When this limit is reached, DB2 queues each request until a thread is available. Once this queuing process is initiated, it causes delays to users.

For transaction-oriented systems such as IMS and CICS, waiting for a DB2 thread can degrade the throughput of the entire system and should be avoided.

In a data center using DB2, users often encounter a shortage of threads. In response to this situation, the DB2THRD solution attempts to identify a TSO thread that can be made available for IMS or CICS to use.

DB2THRD requires that the MainView for DB2 product is also installed.

Variables

To use DB2THRD, you must establish values for the general DB2 variables and the DB2 subsystem variables defined in the DB2INIT solution.

Additionally, you should define the variable listed in the following table:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q_VERIFY</td>
<td>indicates (Y/N) whether to issue an ALERT to verify the TSO cancel</td>
</tr>
</tbody>
</table>

Invocation

This solution is invoked by Rules for the major messages listed in the following sample:

- DZ1020S - IMS TASK(S) QUEUED FOR THREAD
- DZ1021I - EXCEPTION CLEARED: IMS TASK(S) QUEUED FOR THREAD
- DZ1030S - CICS TASK(S) QUEUED FOR THREAD
- DZ1031I - EXCEPTION CLEARED: CICS TASK(S) QUEUED FOR THREAD
Flow process

If \( Q\_VERIFY = Y \), an ALERT is created when the message DZ1020W or DZ1030W appears.

This ALERT has an extended help panel (DPE040A) associated with it that can be modified by your site to give the operator-specific instructions for this situation. When the queued for thread condition clears, the solution intercepts messages DZ1021I and DZ1031I and deletes the ALERT.

If \( Q\_VERIFY = N \), EXEC DPE041C is invoked to identify the TSO user with the shortest elapsed time. EXEC DPE040C then cancels that TSO user so that a thread can be freed up for CICS or TSO to use. A message is written to the journal identifying the canceled TSO user.

EXECs

The EXECs for the DB2THRD solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPE040C</td>
<td>analyzes the message it has been scheduled from</td>
</tr>
<tr>
<td>DPE041C</td>
<td>analyzes MainView for DB2 USERS service output to locate the TSO user to be canceled</td>
</tr>
</tbody>
</table>

Rules

There is one Rule for each of the messages listed in the Invocation section.

The Rule-IDs equal the message-IDs.

The Rules are distributed DISABLED in RULESET AAORULD6. You must ENABLE these Rules to implement this solution.
DB2 performance management solutions

The following DB2 performance management solutions help data center personnel achieve maximum availability and increased response time.

For several DB2 solutions, you must also have the MainView for DB2 product installed.

DB2 runaway query control solution (DB2RNWY)

Runaway queries are one of the major problems affecting DB2 performance.

MainView for DB2 dynamic and static SQL calls and warning capabilities made it possible to develop the DB2RNWY solution, which provides selective cancel capabilities over such runaway queries.

This solution requires

- MainView for DB2
- MainView AutoOPERATOR for IMS Online
- MainView AutoOPERATOR for CICS (to cancel a runaway CICS query)

Note
DB2 provides limited protection from runaway queries with the Resource Limit Facility (RLF), available with DB2 version 2.1. However, this protection applies only to dynamic SQL. Moreover, the result from RLF is a query that has stopped with an SQL error code (-905 expected).
Variables

To use DB2RNWY, you must establish values for the global DB2 variables that are defined in the DB2INIT solution.

Invocation

This solution is invoked by Rules for the following messages:

- **DZ0610W** RUNAWAY IMS <threshold> PST= TRAN= USER= J= TYPE= CRGN=
- **DZ0611I** EXCEPTION CLEARED: RUNAWAY IMS ...
- **DZ0620W** RUNAWAY CICS TRANSACTION USER=u <threshold> J= TRAN= TASK=
- **DZ0621I** EXCEPTION CLEARED: RUNAWAY CICS TRANSACTION ...
- **DZ0630W** RUNAWAY TSO QUERY USER= <threshold exceeded>
- **DZ0631I** EXCEPTION CLEARED: RUNAWAY TSO QUERY ...
- **DZ0640W** RUNAWAY BATCH USER= <threshold exceeded> J=
- **DZ0641I** EXCEPTION CLEARED: RUNAWAY BATCH ...
- **DZ0650W** RUNAWAY CAF, USER= <threshold exceeded> J=
- **DZ0651I** EXCEPTION CLEARED: RUNAWAY CAF ...
- **DZ0660W** RUNAWAY UTILITY FUNCTION, USER= <threshold exceeded> J=
- **DZ0661I** EXCEPTION CLEARED: RUNAWAY UTILITY ...

Flow process

When MainView for DB2 detects a runaway query, as determined by thresholds set in BBPARM member DMRBEX00, a message is sent to the LOG and filtered to be echoed as an ALERT on the MainView centralized screen.

A sample member named DMRBEXD1 is provided in the BBPARM data set.

Operators must confirm the cancel of a runaway query job by entering CAN in the response field for the ALERT; the ALERT is cleared when the operator cancels the runaway query job, or when the runaway condition ends.

If the runaway query is

- a job or TSO user, the cancel is done through the appropriate MVS cancel command
- an IMS region, the solution verifies that the region name, number, and trancode match those in the warning message and that the region status is ACTV-USR or ACTV-DB2

It then issues a /STOP REG ABDUMP command followed by a /STOP REGION CANCEL command.

- a CICS transaction, the solution issues a KILL TASK xxxx FORCE

Note: This solution does not cancel a runaway query automatically. It always requests operator agreement by the ALERT.

**EXECs**

The EXECs for the DB2 runaway query control solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPE020C</td>
<td>create ALERTs for runaway query</td>
</tr>
<tr>
<td>DPE021C</td>
<td>delete ALERTs for runaway query</td>
</tr>
<tr>
<td>DPE022C</td>
<td>schedules MVS cancel EXEC DPE024C</td>
</tr>
<tr>
<td>DPE023C</td>
<td>schedules IMS cancel EXEC DPE025C</td>
</tr>
<tr>
<td>DPE024C</td>
<td>performs MVS cancel</td>
</tr>
<tr>
<td>DPE025C</td>
<td>performs IMS cancel</td>
</tr>
<tr>
<td>DPE026C</td>
<td>schedules CICS cancel EXEC DPE027C</td>
</tr>
<tr>
<td>DPE027C</td>
<td>performs CICS cancel</td>
</tr>
</tbody>
</table>

**Rules**

The Rules are distributed disabled in RULESET AAORULD2.

You must enable these Rules to implement this solution.
DB2 response time solution (DB2RESP)

Because transaction response time for all CICS, IMS, and TSO users relies on DB2's ability to reply to requests in the shortest possible time, it is important to obtain data about DB2 response time as early as possible.

This solution examines several MainView for DB2 (DMR) services for potential performance problems by comparing the observed values to user-defined thresholds and by reporting any observations that are above threshold levels.

DB2RESP is initiated by certain warning messages for elapsed and CPU time monitors or invoked directly by the operator.

The usefulness of this solution depends in large part on the thresholds chosen, so these must be chosen carefully. BMC has supplied defaults with this solution, but you may want to make some modifications to fit your installation.

In the process of determining your installation thresholds, you should become aware of the MainView for DB2 data items that are good performance indicators for your system.

This solution requires the DB2INIT and DMRUTIL solutions.

Variables

To use DB2RESP, you must establish values for the general DB2 variables and the DB2 subsystem variables defined in the DB2INIT solution.

This solution makes extensive use of thresholds for many domains of DB2 performance. For each DB2 subsystem, up to 35 important figures are analyzed. You can change any of the default thresholds. They are available for each DB2 system, and are located in the DSU10 n C EXECs.

For more information on setting these threshold parameters, see “DB2 solution variables and parameters” on page 361.

Invocation

This solution is invoked by Rules for the following messages:

- DW0120W (nn) hh:mm: AVG ELAPSED TIME(parm) = nnn (>ppp) *****
- DW0150W (nn) hh:mm: AVG ELAPSED IN DB2(parm) = nnn (>ppp) *****
- DW0180W (nn) hh:mm: AVERAGE CPU USED(parm) = nnn (>ppp) *****
where

parm \{ALL | IMS | CICS | TSO\}

To get these messages out of MainView for DB2, you must activate some monitors as listed in member DPE002B of the BBPARM data set. This activation is done in BBPARM member BBIISPxx by using the TARGET statement, as shown in BBIISPD1 of BBPARM.

For example, to set the DB2 monitors, specify the following command:

```
TARGET=DB2D, BLK=DPE002B, USRID=xxxxxxx
```

You should adjust the WVAL values to values appropriate to your installation. Information about WVAL values can be found in the "Set Timer Request" chapter of the MainView for DB2 User Guide.

DB2RESP processes warning messages with an identifier of IMS, CICS, TSO, or ALL. These identifiers are required and cannot be modified.

---

**Flow process**

The DB2RESP solution is triggered by one of the following events:

- Messages that are listed in “Invocation” on page 248
- Invoking the solution from any MainView COMMAND line by entering the following command:

  `%DPE002C db2name`

Here, `db2name` is the name of the target DB2.

You can view online help is available by entering the following command:

 `%DPE002C ?`

The positional parameters for DPE002C are listed in the following table:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARM1</td>
<td>DB2 TARGET (REQUIRED)</td>
</tr>
</tbody>
</table>
## Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARM2</td>
<td>OUTPUT DESTINATION (OPTIONAL) DEFAULT = DB2 TARGET</td>
</tr>
<tr>
<td>PARM3</td>
<td>CONNECTION TYPE (OPTIONAL)</td>
</tr>
</tbody>
</table>

Values can be ALL (the default), IMS, CICS, or TSO.

### EXECs

The EXECs for the DB2RESP solution are in the following table:

<table>
<thead>
<tr>
<th>EXEC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPE002C</td>
<td>performs checking of thresholds for DB2</td>
</tr>
<tr>
<td>DPE005C</td>
<td>drives DPE002C from DB2 monitor messages</td>
</tr>
<tr>
<td>DPE007C</td>
<td>displays the governor for solution driven from monitor warning message</td>
</tr>
</tbody>
</table>

### Rules

The Rule-IDs equal the message IDs.

There is one Rule for each of the messages listed under “Invocation” on page 248.

The Rules are distributed disabled in RULESET AADORULD5. You must enable these Rules to implement this solution.
E-mail solution

MainView AutoOPERATOR provides a pair of sample EXECs that can be used to send an e-mail message from a REXX EXEC.

E-mail solution

The following sections describe how to implement the E-mail solution.

EXECs

The EXECs for the E-mail solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QAOSMTP1</td>
<td>resides in BBSAMP</td>
</tr>
<tr>
<td>QAOSMTP2</td>
<td>resides in BBPROC</td>
</tr>
<tr>
<td>QAOEMAIL</td>
<td>resides in BBPROC and can be modified and scheduled by a Rule or EXEC to send an E-mail</td>
</tr>
<tr>
<td>QAOSMTP3</td>
<td>resides in BBSAMP and is a sample of the QAOEMAIL EXEC that you can copy and customize</td>
</tr>
</tbody>
</table>

Note

Browse the MainView AutoOPERATOR product libraries to view the most current version of these sample EXECs. BMC might ship PTFs at any time to enhance them. This SMTP sample uses REXX sockets, which are part of the IBM TCP/IP product.
QAOSMTP1

QAOSMTP1 is a sample EXEC that demonstrates how you can pass e-mail data to QAOSMTP2, which performs the actual e-mail transmission.

You might not have to make any modifications to the QAOSMTP2 EXEC.

However, to use the solution, you must customize QAOSMTP1. Make a copy of QAOSMTP1, rename the EXEC, and store it in UBBPROC. In UBBPROC, edit the EXEC to use values from your system.

You can also incorporate the logic in QAOSMTP1 into your own REXX application to generate e-mail messages.

The areas that need to be modified in QAOSMTP1 are listed in the following table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP_SERVER</td>
<td>is the name of the local mail server</td>
</tr>
<tr>
<td></td>
<td>This server distributes mail to the POP3 and Exchange mail servers.</td>
</tr>
<tr>
<td></td>
<td>You can manually specify the SMTP_SERVER name in the EXEC or you can redefine</td>
</tr>
<tr>
<td></td>
<td>the value in a PROFILE variable via the windows-mode EZAOSOL menu. Using a</td>
</tr>
<tr>
<td></td>
<td>PROFILE variable is preferable, as the values can be easily changed in the future</td>
</tr>
<tr>
<td></td>
<td>without having to edit the EXEC source code. For more information about the</td>
</tr>
<tr>
<td></td>
<td>EZAOSOL menu, see the MainView AutoOPERATOR Basic Automation Guide Volume 2.</td>
</tr>
<tr>
<td>SMTP_REPLYTO</td>
<td>is the e-mail message address that receives replies when responding to this e-mail message</td>
</tr>
<tr>
<td></td>
<td>It must be a valid address (or example, <a href="mailto:yourName@yourCompany.com">yourName@yourCompany.com</a>).</td>
</tr>
<tr>
<td>Sender of the e-mail message</td>
<td>is the first line queued (your name), which might be different from the REPLYTO e-mail ID</td>
</tr>
<tr>
<td></td>
<td>Valid formats are dependent upon the mail server that is being used. Most mail servers accept '<a href="mailto:user@yourCompany.com">user@yourCompany.com</a>'. Other typical formats that are accepted are &quot;your name&quot; and &quot;your name&quot;<a href="mailto:id@bid.company.com">id@bid.company.com</a>' . Verify with your site's SMTP documentation for acceptable values.</td>
</tr>
<tr>
<td>Recipient(s)</td>
<td>is a blank-delimited list</td>
</tr>
<tr>
<td></td>
<td>It is queued second. Do not use the format &quot;name&quot;<a href="mailto:user@bigCompany.com">user@bigCompany.com</a>.</td>
</tr>
<tr>
<td>Email Subject</td>
<td>is the text for the subject line of the e-mail</td>
</tr>
<tr>
<td>Email text</td>
<td>is a line or multiple lines of text that will be sent to the recipients</td>
</tr>
<tr>
<td></td>
<td>This information is displayed in the body of the e-mail. message</td>
</tr>
<tr>
<td>CC_USERS</td>
<td>is a list of users who will receive a copy of this e-mail message</td>
</tr>
<tr>
<td></td>
<td>Use the same syntax as used for the Recipient(s) field.</td>
</tr>
</tbody>
</table>
Note
A sender's name, subject, and at least one line of message text are required. Keep line lengths reasonable so that the e-mail message has a readable format. The utility EXEC (QAOSMTP2) inserts a CrLf after each text line in the generated e-mail message.

A QAOSMTP1 sample EXEC follows:

```rexx
/* rexx qaosmtp1 */
/* DOC GROUP(AO) FUNC(SAMPLE) AUTHOR(BMC) */
/* DOC DESC(sample EXEC for sending email via a SMTP server) */
/*--------------------------------------------------------------------
* Sample EXEC which demonstrates sending an e-mail via a SMTP mail server. *
* This EXEC constructs the e-mail data then calls QAOSMTP2 which reformats the data and sends it using the SMTP protocol.
* QAOSMTP2 talks to the SMTP server at port 25 using IBM REXX Sockets. NOTE: QAOSMTP2 resides in BBPROC.
* Change Log
* 10-mar-05 bpoxxxx Add CC: list
*  9-mar-01 bpo4962 Use queue command instead of push
*  9-feb-01 bpo4874 shipped to make bpo4874 prereq bpo4851
* 30-jan-01 bpo4851 update comments to note QAOSMTP2 is in BBPROC
* 27-aug-99 bpo4319 initial release
*/
trace n
"IMFEXEC MSG '.QAOSMTP1 EID" IMFEID "Started'
address mvs newstack                       /* make a new stack */
 /*--------------------------------------------------------------------
* ----- Beginning of lines requiring changes ----------------------- *
* The external data queue is used to communicate the message text, subject and sender's name to the utility EXEC (qaosmtp2). *
* smtp_server is your local SMTP server. This server will distribute mail to the POP3 and exchange mail servers. Many mail servers will have "mail" as the first part of their network name but is not required to follow this standard. *
* smtp_replyto is the Email address that will receive replies when responding to this Email. You should have a valid address here. *
* Sender of the Email is the first line queued. It will be your name which may be different from the ReplyTo e-mail id. *
* The formats allowed are dependant upon the mail server in use. '<user@your-company.com>' is accepted by most mail servers. Some other typical formats that are accepted '"your name" and '"your name"@<id@bid.company.com>'. Check your SMTP documentation for acceptable values. *
* Recipient(s) is a blank delimited list. It is queued second. *
* Do not use the form "name"<user@big.company.com>.
* Copied users is a blank delimited list. It is passed to QAOSMPT2 instead of being queued. *
* Subject line is the third queued line. */```
Email message text will be the remainder of lines queued.

There must be a sender's name, a subject and at least one line of message text.

Keep line lengths reasonable so that your e-mails will look nice.

The utility EXEC will insert a CrLf after each text line in the generated e-mail.

smtp_server = 'mail.your-company.com' /* your SMTP mail server */
smtp_replyto = 'YourName@your-company.com' /* Reply-to Email id */
queue '<<strip(imforgn)>'_AutoOPERATOR@your-company.com>' /* sender */
queue 'user1@xxx.com user2@yyy.com group1@zzz.com' /* recipients */
cc_users = '' /* copied users */
queue 'Email subject' /* Emails subject line */
queue 'Email text line 1' /* Email body */
queue 'Email text line 2'
queue 'Email text line 3'

smtp_debug = 'n' /* optional debug parm */
smtp_error = 0 /* preset our error flag */
if words(smtp_server) /= 1 then
  do
    "IMFEXEC MSG '.QAOSMTP1 EID" IMFEID "Error, blanks not allowed in",
    "smtp_server="smtp_server""
  smtp_error = 1
  end
if words(smtp_replyto) /= 1 then
  do
    "IMFEXEC MSG '&QAOSMTP1 EID" IMFEID "Error, blanks not allowed in",
    "smtp_replyto="smtp_replyto""
  smtp_error = 1
  end
if smtp_error = 1 then
  do
    "IMFEXEC MSG '.QAOSMTP1 EID" IMFEID "Ended with error(s)"
    "IMFEXEC EXIT CODE(8)"
  exit
  end
if cc_users = "CC_USERS" then cc_users ="
if cc_users <> "" then cc_users = "CC:" cc_users
/* call the utility EXEC to send the e-mail */
call qaosmtpl smtp_server smtp_replyto smtp_debug , cc_users
exec_rc = result /* save qaosmtpl2 rc */
address mvs delstack /* delete our stack */
if exec_rc /= 0 then                       /* clean up and exit     */
   "IMFEXEC MSG '.QAOSMTP2 ended with rc=" exec_rc"'
"IMFEXEC MSG '.QAOSMTP1 EID" IMFEID "Ended'"

Some SMTP servers depend on its users to supply the correct time offset from UTC (GMT). Other SMTP servers supply this time offset internally and do not depend on its users. If you see an inaccurate "sent" time on an Email that you received, this is a common symptom of an incorrect time offset.

The EXEC QAOSMTP2 obtains the local time offset from MVS and appends this value to the Date and Timestamp. The example below shows a timestamp for an email generated on a MVS with local time 7 hours behind UTC.

**Date: Wed, 12 Oct 2005 16:16:58 -0700**

If your MVS system does not have the correct time offset from UTC (for example, this applies to some customers who run MVS with the local time set to the GMT), you may need to customize the QAOSMTP2 EXEC to hard code the appropriate time offset. You might also have to do this if you change the time to observe Daylight Savings Time (DST).

To view the local time offset on your MVS, issue the command **MVS D T** (display time).

# QAOEMAIL

The QAOEMAIL EXEC is a ready-to-use EXEC that you can schedule from an EXEC or a Rule to send E-mails.

To use this EXEC, you must specify the SMTP server name from the EZAOSOL menu.

For more information about the EZAOSOL menu, see the *MainView AutoOPERATOR Basic Automation Guide: Volume 2*.

To schedule the QAOEMAIL (or QAOSMTP3) EXEC from a Rule, you must enter the EXEC name in the **EXEC/PARMS** field on the Action Specification panel (A1) for the Rule. You must also enter the following parameters (in order) where each parameter is separated by a blank space: **to replyTo subject**.

The following table describes the parameters:
<table>
<thead>
<tr>
<th><strong>Parameter</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| to           | Is the E-mail address(es) of the E-mail recipient(s)  
You can specify multiple E-mail address with this parameter by separating each E-mail address with a comma. Use the standard E-mail address for the recipient; for example: `name@emailServer.sfx` |
| replyTo      | Is the E-mail address of the sender  
You can specify only one E-mail address for this parameter. Use the standard E-mail address for the sender; for example: `name@emailServer.sfx` |
| subject      | Is one or more words, separated with blank spaces, that represent the Subject line of the E-mail |
MainView AutoOPERATOR provides sample solutions that you can use to generate SNMP Traps:

- The QAOSNMP EXEC that is distributed as a ready-to-use EXEC that can be scheduled by an EXEC or Rule
- The QAOSNMP1 and QAOSNMP2 EXECs generate Traps that contain one OID
- The QAOSNMP3 and QAOSNMP4 EXECs support multiple OIDs in each Trap

These two solutions have different calling conventions.

**QAOSNMP**

QAOSNMP is a ready-to-use EXEC that you can schedule it from an EXEC or a Rule to generate a SNMP Trap.

To use this EXEC, you must specify the values for the SNMP parameters from the EZAOSOL menu.

For more information about the EZAOSOL menu, see the *MainView AutoOPERATOR Basic Automation Guide: Volume 2*.

To schedule the QAOSNMP EXEC from a Rule, you must enter the EXEC name in the **EXEC/PARMS** field on the Action Specification panel (A1) for the Rule. You must also enter the following parameters (in order) where each parameter is separated by a blank space: *ObjectId* OIDText.

The following table describes the parameters:
EXECs for traps that contain one OID

The EXECs for this solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QAOSNMP1</td>
<td>resides in BBSAMP</td>
</tr>
<tr>
<td>QAOSNMP2</td>
<td>resides in BBPROC</td>
</tr>
</tbody>
</table>

**Note**
Browse the MainView AutoOPERATOR product libraries to view the most current version of these sample EXECs. BMC might ship PTFs at any time to enhance them. This SNMP sample uses REXX sockets, which are part of the IBM TCP/IP product.

**QAOSNMP1**

QAOSNMP1 is a sample REXX EXEC that demonstrates how to generate a SNMP Trap.

The actual Trap generation logic is encapsulated in QAOSNMP2, which should be usable without modification by most customers.

To generate an SNMP Trap, copy QAOSNMP1 into UBBPROC using a new EXEC name. Replace the data in the user-input fields with the appropriate data for your installation.

You can also incorporate the logic from QAOSNMP1 into your own REXX application to generate SNMP Traps.
SNMP Traps are directed to SNMP Managers. Examples of SNMP Managers are PATROL Enterprise Manager, Tivoli Enterprise Console, and so on. You must coordinate your efforts with the administrators of your SNMP Manager. Consult them for any SNMP values of which you are unsure.

The user-input fields are as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Community</td>
<td>is a text string, usually used as a simple security mechanism</td>
</tr>
<tr>
<td>SnmpManager</td>
<td>is the IP address or host name of your SNMP Manager</td>
</tr>
<tr>
<td>SnmpPort</td>
<td>is the listening port number used by your SNMP Manager</td>
</tr>
<tr>
<td>Enterprise</td>
<td>is an object identifier that identifies the device that generates the Trap</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: Nodes must be less than 128.</td>
</tr>
<tr>
<td>GenericTrap</td>
<td>is an integer, specified from the SNMP-defined values for a generic trap</td>
</tr>
<tr>
<td>SpecificTrap</td>
<td>is an integer, specified for a specific trap</td>
</tr>
<tr>
<td>TimeTicks</td>
<td>is an integer; it usually represents the device uptime in hundredths of a second</td>
</tr>
<tr>
<td>Object Identifier</td>
<td>is a unique identifier for this event trap</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: Nodes must be less than 128.</td>
</tr>
<tr>
<td>OIDtype</td>
<td>is the type of data; QAOSNMP2 supports Octet String (4) only</td>
</tr>
<tr>
<td>OIDtext</td>
<td>is the trap text</td>
</tr>
</tbody>
</table>

The following is an example of a QAOSNMP1 EXEC.

```rexx
/* rexx */
00010000 /* */
00020000 /* */
00030000 /* Sample EXEC which calls QAOSNMP2 to issue a */
00040000 /* SNMP v1 Trap */
00050000 /* */
00060000 /* Nearly all parameters are passed to QAOSNMP2 */
00070001 /* as positional parameters. QAOSNMP2 resides */
00071001 /* in BBPROC. */
00080000 /* */
00090000 /* Community and OIDtext are passed on the data */
00100000 /* stack. */
00110000 /* */
00120000 /* Note: Object Identifiers must begin with 1.3.x */
```
/*     Subsequent nodes must be less than 128     */
01300000
/*     eg. valid = 1.3.6.2.4     */
01400000
/*     invalid = 2.9.1     */
01500000
/*     invalid = 1.3.200     */
01600000
/*     */
01700000
/*     change log: */
01800000
/*     */
01900000
/*     */
02000000
/*     */
02100000
02200000
/*     this is a string ... primitive SNMP v1 security */
02300000
Community = 'public'
02400000
/*     this can be any valid IP address or host name */
02500000
/*     eg. 132.60.1.3 or manager.bigcompany.com */
02600000
SnmpManager = 'snmpMgr.BigCompany.com'
02700000
/*     this can be any valid port number (integer) */
02800000
/*     normally it is 162 */
02900000
SnmpPort = 162
03000000
/*     this Object Identifier identifies the 'device' */
03100000
/*     which is generating the Trap */
03200000
/*     note: nodes must be less then 128 */
03300000
Enterprise = '1.3.6.1.4.2'
03400000
/*     SNMP defined values for Generic Trap are */
03500000
/*     0 = cold start */
03600000
/*     1 = warm start */
03700000
/*     2 = link down */
/* 3 = link up */
/* 4 = authentication failure */
/* 5 = egp neighbor loss */
/* 6 = enterprise specific */

GenericTrap = 6

/* specify an integer for Specific Trap */
SpecificTrap = 0

/* Time Ticks is an integer, it usually represents */
/* the device uptime in hundredths of a second */
TimeTicks = 5100000

/* Object Identifier which uniquely identifies */
/* this event (Trap) */
/* note: nodes must be less then 128 */
ObjectId = '1.3.6.1.4.1.1.2'

/* OIDtype represents the type of data, */
/* QAOSNMP2 supports Octet String (4) only */
OIDtype = 4

/* This is the text of the Trap */
OIDtext = '123 this is the text of the Alert'

/* Issue NewStack to insulate us from any callers */
address mvs 'newstack'

/* put on data stack */
push Community
push OIDtext

Chapter 23  SNMP solution  261
EXECs for traps that contain multiple OIDs

The EXECs for this solution are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QAOSNMP3</td>
<td>resides in BBSAMP</td>
</tr>
<tr>
<td>QAOSNMP4</td>
<td>resides in BBPROC</td>
</tr>
</tbody>
</table>

*Note*
Browse the MainView AutoOPERATOR product libraries to view the most current version of these sample EXECs. BMC might ship PTFs at any time to enhance them. This SNMP sample uses REXX sockets, which are part of the IBM TCP/IP product.

QAOSNMP3

QAOSNMP3 is a sample REXX EXEC that demonstrates how to generate a SNMP Trap.

The QAOSNMP4 EXEC contains the Trap generation logic.

To generate an SNMP Trap, copy and rename the QAOSNMP3 EXEC into UBBPROC. Edit the user-input fields and enter values that will work for your system.

You can also incorporate the logic from QAOSNMP3 into your own REXX application to generate SNMP Traps.
Variables for QAOSNMP1 and QAOSNMP3

The following table lists the variables you can use for configuring the QAOSNMP1 or QAOSNMP3 EXECs.

You can manually specify the variable values listed in the table below directly in the EXEC or, you can predefine the value in a PROFILE variable via the windows-mode EZAOSOL menu. Using a PROFILE variable is preferable, as the values can be easily changed in the future without having to edit the EXEC source code. For more information about the EZAOSOL menu, see the MainView AutoOPERATOR Basic Automation Guide: Volume 2.

<table>
<thead>
<tr>
<th>EXEC variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SnmpManager</td>
<td>Is the IP address or host name of your SNMP Manager</td>
</tr>
<tr>
<td>SnmpPort</td>
<td>Is the listening port number used by your SNMP Manager</td>
</tr>
<tr>
<td>Community</td>
<td>Is a simple string that provides primitive V1 SNMP security. For example, you can use a value of &quot;public&quot;.</td>
</tr>
<tr>
<td>Enterprise</td>
<td>Is a node address that presents the source of the SNMP trap. For example, you can use a value such as 1.3.6.2.1. Values in each node must be less than 128.</td>
</tr>
</tbody>
</table>
Paging solution

MainView AutoOPERATOR provides sample code that enables you to communicate with one or two-way wireless devices, including both text and numeric pagers.

You can also obtain confirmations and responses if the destination device and its provider offer these capabilities.

This sample was developed by using the Simple Network Paging Protocol (SNPP rfc 1861) as implemented by SkyTel Communications, Inc.

EXECs

The EXECs for the paging sample are listed in the following table:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QAOSNPP1</td>
<td>contains the calling communication logic</td>
</tr>
<tr>
<td>QAOSNPP2</td>
<td>contains the communication logic</td>
</tr>
</tbody>
</table>

The EXECs are not compiled, so you can modify them for your application.

QAOSNPP1

The QAOSNPP1 EXEC is the calling EXEC and does not have input parameters.

QAOSNPP2

The QAOSNPP2 EXEC can have from 0 to 9 parameters, depending on the mode.

Each parameter (except the message text) has the following format:
PARAMETER_NAME(value)

You must separate the parameters with blank spaces and you can enter them in any order. The parameters are described in the following table.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HELP</td>
<td>(optional) The default value is HELP(NO). If you specify HELP(YES), and the program is called without parameters, or if HELP is specified with no parameters, the calling format is printed to the BBI Journal.</td>
</tr>
</tbody>
</table>
| MODE      | (optional) The default value is MODE(SEND). The following lists the valid values for the MODE parameter:  
  - SEND: send the message.  
  - STATUS: check the message status. |

Parameters for MODE(SEND)

If you specify MODE(SEND), additional parameters are available that you can set. These parameters are described in the following table.

Table 21: Parameters for MODE(SEND)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVER</td>
<td>Specify the Internet name or address of the message server provider. For example, for SkyTel, the Internet name is snpp.skytel.com.</td>
</tr>
<tr>
<td>PORT</td>
<td>Specify the message server provider port to which the program should connect. In the sample, the port is set to the value 7777.</td>
</tr>
<tr>
<td>PIN</td>
<td>Specify the service provider access PIN or a direct telephone number.</td>
</tr>
<tr>
<td>USER</td>
<td>(optional) specify the name of the user to whom you are sending a message. This value is used in the messages and ALERTs issued by the program. Default is NONE.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>(optional) Specify the time (in seconds) that the program can wait for the TCP/IP response before timing out. Default is 10 seconds.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>CONF</td>
<td>(optional) Specify CONF(YES) if you want to receive a confirmation of a successful message delivery. CONF(YES) can be used only with two-way pagers or telephones with a messaging service. Default is CONF(NO). CONF(NO) can be specified when calling any pager or telephone.</td>
</tr>
<tr>
<td>DEBUG</td>
<td>(optional) Specify DEBUG(YES) to print the contents of all TCP/IP exchanges (and some other debugging information) to the BBI Journal.</td>
</tr>
</tbody>
</table>

**Note**: Message text should be written after key parameters. If the message text must contain an open parentheses symbol, (, separate it with <<. When QAOSNPP2 is called from an IMFEXEC SELECT statement, the total statement length is restricted to 256 characters (including information content). If QAOSNPP2 is called from another EXEC with a CALL statement, the length can be up to 2,500 characters.

### Parameters for MODE(STATUS)

Use the MODE(STATUS) parameter only when an ALERT is created after a message is sent.

The only parameter for MODE(STATUS) is the variable name that contains the list of values for status requests.

### Flow process

The calling EXEC, QAOSNPP1, has four components:

- A tuning section where you can edit the parameters
- A checking section to verify the returned information
- The CALL statement
- A section for result messages processing

QAOSNPP2 contains the main logic and performs the following tasks:

- It extracts and checks parameters.
- It prints the calling format and exit if help is requested.
- It connects to the server/port and performs the TCP/IP message exchange with a server.

- In the CONF(YES) mode, it creates a MainView AutoOPERATOR ALERT that escalates within a minute and calls a follow-up EXEC in STATUS mode. It creates a unique variable name formatted PAGING.xxxxxxxxx (where xxxxxxxx are the last eight numbers of the current time) and puts all necessary information into a long profile variable with this name. The variable name becomes the second parameter of QAOSNPP2.

**Note**

When you specify CONF(NO), no more actions are required.

- When QAOSNPP2 is called in STATUS mode, it sends a status request to the server and replaces the original ALERT with the ALERT escalating up (if the message is still not delivered). Otherwise, it sends a status request to the server and replaces the original ALERT with a clearing ALERT escalating down.

### Variables returned from QAOSNPP2

If the QAOSNPP2 EXEC is invoked with an IMFEXEC SELECT command, the following variables are returned:

- IMFCC: contains the return code from the IMFEXEC SELECT statement.

- IMFRC: contains the condition code returned on the IMFEXEC EXIT statement of QAOSNPP2 when IMFEXEC SELECT WAIT(YES) is specified.

- LOCAL variables APFCC and APFRC, which contain condition and return codes.

- LOCAL variables APFLN1 through APFLN n, which contain additional information.

- LOCAL variable APFNOL, which contains the number of APFLN1-Nn variables.

The following table contains the error messages and the IMFRC, APFRC and APFCC values that are found in LOCAL variables APFLN1 through APFLN n.

<table>
<thead>
<tr>
<th>Error message in APFLN n</th>
<th>IMFRC</th>
<th>APFCC</th>
<th>APFRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF00001I SUCCESS</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PF2001E Message timed out.</td>
<td>8</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>PF2002E <code>&lt;messageFromTheServiceProvider&gt;</code></td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>PF2002E Transmission ID Error</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Error message in APFLNn</td>
<td>IMFRC</td>
<td>APFCC</td>
<td>APFRC</td>
</tr>
<tr>
<td>--------------------------------------------------------------</td>
<td>-------</td>
<td>-------</td>
<td>------------------------</td>
</tr>
<tr>
<td>PF2003E PIN is not recognized by service provider.</td>
<td>8</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>PF3001E QAOPAGE1: SOCKET(SOCKET)PF3001E QAOPAGE1: SOCKET(CONNECT)PF3001E QAOPAGE1: SOCKET(WRITE)PF3001E QAOPAGE1: SOCKET(SELECT)</td>
<td>8</td>
<td>8</td>
<td>TCP/IP return code</td>
</tr>
<tr>
<td>PF4001E QAOPAGE1: User ID is unknown.</td>
<td>8</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>PF4002E QAOPAGE1: Page service provider is unknown.</td>
<td>8</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>PF4003E QAOPAGE1: Unable to initialize SOCKET</td>
<td>8</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>PF5051E QAOPAGE1: Error found in MODE parameter</td>
<td>8</td>
<td>16</td>
<td>51</td>
</tr>
<tr>
<td>PF5052E QAOPAGE1: Error, SERVER or PORT parameter is not provided.</td>
<td>8</td>
<td>16</td>
<td>52</td>
</tr>
<tr>
<td>PF5053E QAOPAGE1: Error in CONF parameter (YES/NO).</td>
<td>8</td>
<td>16</td>
<td>53</td>
</tr>
<tr>
<td>PF5054E QAOPAGE1: Message text is not provided.</td>
<td>8</td>
<td>16</td>
<td>54</td>
</tr>
<tr>
<td>PF5055E QAOPAGE1: PIN number is not provided or not numeric.</td>
<td>8</td>
<td>16</td>
<td>55</td>
</tr>
<tr>
<td>PF5056E QAOPAGE1: Error in TIMEOUT parameter, it is not numeric.</td>
<td>8</td>
<td>16</td>
<td>56</td>
</tr>
<tr>
<td>PF5057E QAOPAGE1: Error in DEBUG parameter (YES/NO)</td>
<td>8</td>
<td>16</td>
<td>57</td>
</tr>
<tr>
<td>PF5061E QAOPAGE1: Error in parameter structure</td>
<td>8</td>
<td>16</td>
<td>61</td>
</tr>
<tr>
<td>PF5062E QAOPAGE1: Unknown parameter</td>
<td>8</td>
<td>16</td>
<td>62</td>
</tr>
<tr>
<td>PF5063E QAOPAGE1: Error during VGETL</td>
<td>8</td>
<td>16</td>
<td>63</td>
</tr>
</tbody>
</table>
The QAOUSS solution allows you to submit a USS command, program, or script from MainView AutoOPERATOR.

Review the comments in the beginning of the EXEC in BBPROC member QAOUSS. These comments will explain what you need to change to make the EXEC work.
Sample solutions

The MainView AutoOPERATOR product provides a set of sample solutions for automation tasks in data centers.

**WARNING**

This set consists of Rules, EXECs, Timer Facilities, and SYSPROG commands. Use the sample solutions to accomplish many of the initial automation tasks quickly and efficiently after you install the MainView AutoOPERATOR product.

With the sample solutions, BMC delivers sample source code that is to be used as examples for solution implementation. BMC does not provide support for any sample solution that has been modified. Customers seeking technical support from BMC must be able to demonstrate that the distributed samples have not been modified.

Distributed sample solutions

All EXECs for sample solutions are distributed in the BBPROC data set allocated to the SYSPROC DD statement in the MainView AutoOPERATOR subsystem.

All Rules for sample solutions are distributed DISABLED in various members of the BBPARM data set.

EXEC naming conventions

The naming convention for the sample solutions EXECs uses the following format:

\[ gffnnnnl \]

where

<table>
<thead>
<tr>
<th>Placeholder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>is the first letter of the group (for example, M for MVS)</td>
</tr>
</tbody>
</table>
### EXEC naming conventions

<table>
<thead>
<tr>
<th>Placeholder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ff</td>
<td>is a 2-character abbreviation for the functional area within the group where</td>
</tr>
<tr>
<td></td>
<td>• AM: represents ACCESS METHODS*</td>
</tr>
<tr>
<td></td>
<td>• ER: represents ERROR RECOVERY*</td>
</tr>
<tr>
<td></td>
<td>• IN: represents INFO SYS/MANAGEMENT</td>
</tr>
<tr>
<td></td>
<td>• J2: represents JES2</td>
</tr>
<tr>
<td></td>
<td>• J3: represents JES3*</td>
</tr>
<tr>
<td></td>
<td>• MN: represents MONITORS</td>
</tr>
<tr>
<td></td>
<td>• PE: represents PERFORMANCE*</td>
</tr>
<tr>
<td></td>
<td>• ST: represents STORAGE SUBSYSTEMS*</td>
</tr>
<tr>
<td></td>
<td>• UT: represents UTILITIES</td>
</tr>
<tr>
<td></td>
<td>• VT: represents VTAM/NCP</td>
</tr>
<tr>
<td>nnn</td>
<td>is a sequence from 001 to 999</td>
</tr>
<tr>
<td>l</td>
<td>is the language of the particular EXEC (for example, X for REXX or C for CLIST)</td>
</tr>
</tbody>
</table>

The following three categories of EXECs do not follow this naming convention:

- **operator-initiated EXECs**

  Operator-initiated EXECs are executed as a result of an operator command.

- **text ID-driven EXECs**

  Text ID-driven EXECs are executed as a result of a WTO or WTOR. The names for Text ID-driven EXECs are the text IDs.

- **SYSPROG Services EXECs**

  SYSPROG Services EXECs are the command interfaces to SYSPROG Services. When a sample solution requires more data to complete its analysis, the SYSPROG Services EXECs can be used to provide concise and pertinent data.

  The sample solutions EXECs are distributed with a documentation box. Refer to “Documentation boxes” on page 22 for an example of the documentation box.
Standards for sample solutions

Some national characters do not translate correctly when you go from EBCDIC format to ASCII format and then back to an EBCDIC format.

The national characters #, !, %, and @ translate differently depending on the country in which the translation takes place. BMC recommends that you avoid using national characters in your naming conventions.

Sample VTAMNCP solution

Much of the work in data centers is performed interactively using online systems and terminal networks.

However, the commands and replies used to interface with the network software can be cumbersome to use.

BMC provides a sample VTAM/NCP solution named VTAM commands which provides a command interface for VTAM.

WARNING

With the sample solutions, BMC delivers sample source code that is to be used only as examples for solution implementation

BMC Customer Support will not provide support for any sample solution which has been modified. Customers seeking technical support from BMC must be prepared to demonstrate that the distributed samples have not been modified.

The following sections describe the sample VTAM Commands solution.

Variables

Activating and inactivating VTAM/NCP resources requires memorizing VTAM command formats.

This sample solution provides command interfaces for enabling and disabling resources and alleviates the need to memorize command formats.

This sample solution does not require any values to be set for sample solution variables.
Invocation

This sample solution is operator-invoked.

To activate a resource

1. Enter the following command on the COMMAND line:

   `%%NETACT P1 P2`

   The following table describes the possible parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>resource name</td>
</tr>
<tr>
<td>P2</td>
<td>scope (optional)</td>
</tr>
</tbody>
</table>

To inactivate a resource

1. Enter the following command on the COMMAND line:

   `%%NETINACT P1 P2`

   The following table describes the possible parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>resource name</td>
</tr>
<tr>
<td>P2</td>
<td>type</td>
</tr>
</tbody>
</table>

To inactivate and then reactivate a resource

1. Enter the following command on the COMMAND line:

   `%%NETBOUNC P1 P2 P3`

   The following table describes the possible parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>resource name</td>
</tr>
<tr>
<td>P2</td>
<td>type of inactivate action (the default value is IMMEDIATE)</td>
</tr>
<tr>
<td>P3</td>
<td>scope for activate</td>
</tr>
</tbody>
</table>
Flow process

NETACT issues a V NET,ACT command for the resource specified.

If specified by the operator, SCOPE is included in the command. NETINACT issues a VARY NET,INACT command for the resource specified. If specified by the operator, TYPE is included in the command.

NETBOUNC issues a VARY NET,INACT command for the resource specified. A type of IMMEDIATE is used in the command, unless the operator specifies FORCE. After waiting 10 seconds for the resource to become inactive, a VARY NET,ACT command is issued. SCOPE is included in the command if specified by the operator.

EXECs

The following list describes the EXECs for the sample VTAM commands solution.

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETACT</td>
<td>activates resources</td>
</tr>
<tr>
<td>NETBOUNC</td>
<td>bounces resources</td>
</tr>
<tr>
<td>NETINACT</td>
<td>inactivates resources</td>
</tr>
</tbody>
</table>

Rules

There are no Rules for the sample VTAM commands solution.

Sample operations monitors solution

The sample operations monitoring solution is designed to assist data center personnel monitor events occurring in the system.

The operations monitors sample helps operators gather detailed information about problem areas.
Variables

This sample does not require any variable values to be set.

Invocation

This sample solution is operator-invoked.

Invoke the sample solution by positioning the cursor under certain fields on the MainView AutoOPERATOR MVS STATUS panel and pressing Enter.

For example, positioning the cursor underneath the TSOCPU field invokes the TSOCPU monitor.

Flow process

The following sections describe the processing flow for the sample operations monitors solution.

CSAABV and CSABLO monitors

The SYSPROG CSSUM command is issued to determine which address spaces have the most extended CSA (CSAABV) or CSA (CSABLO) allocated.

The top five users are reported, using the ALERT application.

TOTSIO monitor

The SYSPROG MDEV command is issued to monitor all DASD devices for 15 seconds.

The ALERT facility reports the five volumes with the highest I/O rates. A follow-up EXEC (TIOTSIO2) is associated with the ALERTs.

TOTSIO2 issues the SYSPROG MDEV command to monitor the device specified in the ALERT for 15 seconds. ALERTs are issued to show which address spaces are using the volume and the percentage of utilization attributed to each address space.
TOTCPU JOBCPU TSOCPU and STCCPU monitors

The SYSPROG CPU command is issued to determine which address spaces are major users of the CPU.

Each monitor reports on a different type of top user:

- TOTCPU: top overall CPU user
- JOB CPU: top batch job CPU user
- TSOCPU: top TSO CPU user
- STCCPU: top Started Task CPU user

Another ALERT is generated in addition to the one identifying the major CPU users. The additional ALERT asks the operator if the address space should be canceled. If the operator confirms the cancel, the address space is canceled.

WATCH monitor

The SYSPROG PROGRESS command is issued to determine current statistics for the job under which the cursor is positioned.

The information is reported by using the ALERT facility.

EXECs

The following list describes the EXECs for this solution:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CSAABV</td>
<td>determines top five users of extended CSA</td>
</tr>
<tr>
<td>$CSABLO</td>
<td>determines top five users of CSA</td>
</tr>
<tr>
<td>$JOB CPU</td>
<td>determines top batch job user of CPU</td>
</tr>
<tr>
<td>$STCCPU</td>
<td>determines top Started Task user of CPU</td>
</tr>
<tr>
<td>$TSOCPU</td>
<td>determines top TSO user of CPU</td>
</tr>
<tr>
<td>$TOTCPU</td>
<td>determines top user of CPU</td>
</tr>
<tr>
<td>$TOTSIO</td>
<td>determines top five busiest DASD devices</td>
</tr>
<tr>
<td>$TOTSIO2</td>
<td>determines which address spaces are accessing a particular device</td>
</tr>
<tr>
<td>$WATCH</td>
<td>provides job statistics</td>
</tr>
<tr>
<td>EXEC name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>MMN002C</td>
<td>issues message asking if the address space should be canceled</td>
</tr>
<tr>
<td>MMN003C</td>
<td>cancels address space</td>
</tr>
</tbody>
</table>

**Rules**

There are no Rules for the sample operations monitors solution.

**Sample CICSPlex solution**

During its operations, CICSPlex issues a number of messages with various actions or severity levels.

Some of these messages require immediate attention.

The sample CICSPlex solution

- transforms these messages (listed in “Invocation” on page 280) into MainView AutoOPERATOR ALERTs
- assigns a color to each message based on the message's ID and severity level of the CICSPlex problem

Rules for BB* type messages are processed first for address spaces other than CICSPlex address spaces that issue these messages. Therefore, ALERTS created by these Rules provide the visibility of problems in other areas (such as a CAS).

If you require that the sample solution applies only to the operations of CICSPlex, you must modify the necessary BB* Rules to include the appropriate logic to determine the job name issuing the message.

**Invocation**

This sample solution is invoked by message-initiated Rules that match the following categories.

MainView AutoOPERATOR ALERTs are created (with throttling) in the CICSPlex ALERT queue for the following messages.
### Table 22: Sample CICSPlex solutions message types

<table>
<thead>
<tr>
<th>Message ID types</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
</table>
| BB*U             | all unrecoverable messages | CICSPlex processing terminates  
A red critical ALERT is created that includes the time, date, job name, message text, and an ALERT key of **UNRECOVERABLE ERROR**. |
| BB*S             | all severe error messages | CICSPlex processing is suspended  
A red critical ALERT is created with time, data, job name, message text, and an ALERT key of **SEVERE ERROR**. |
| BBMZA800W        | CAS(ssid) connect interface disabled | a red critical alert is created with time, date, job name, and message text |
| BBC*W            | all warning messages related to the Communications Manager | possible problems but CICSPlex processing continues  
A yellow major ALERT is created with time, date, job name, message text, and an ALERT key of **COMMUNICATIONS ERROR: INVESTIGATE**. |
| BB*W             | all warning messages | possible errors but CICSPlex processing continues  
A yellow major ALERT is created with time, date, job name, message text, and an ALERT key of **WARNING ERROR: INVESTIGATE**. |
| BBMZA995A        | CAS(ssid) reply U to wait or C to continue termination | WTOR for termination  
A pink major ALERT is created with time, date, job name, message text, and an ALERT key of **WTOR FOR TERMINATION**. |
| BBMZA002A        | CAS(ssid) operator interface is ready | a green informational ALERT is created with time, date, job name, message text, and an ALERT key of **CAS OPERATOR INTERFACE READY**. |
| BBMZA094A        | CAS(ssid) reply Y to confirm cold start or N to terminate | WTOR to confirm cold start as coordinating address space determined that SSID is currently or previously used  
A pink major ALERT is created with time, date, job name, message text, and an ALERT key of **COLD START WTOR OUTSTANDING**. |
| EYU*U            | all unrecoverable messages | CICSPlex processing terminates  
A red critical ALERT is created with time, date, job name, message text and an ALERT key of **UNRECOVERABLE ERROR**. |
| EYU*S            | all severe error messages | CICSPlex processing is suspended  
A red critical ALERT is created with time, data, job name, message text and an ALERT key of **SEVERE ERROR**. |
| EYUC*W           | all warning messages related to the Communications Manager | possible problems but CICSPlex processing continues  
A yellow major ALERT is created with time, date, job name, message text and an ALERT key of **COMMUNICATIONS ERROR: INVESTIGATE**. |
**Message ID types**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EYUXL0205D</td>
<td>parameter errors have occurred: reply GO or CANCEL</td>
<td>WTOR for handling parameter errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A pink critical ALERT is created with time, date, job name, message text, and an ALERT key of WTOR OUTSTANDING.</td>
</tr>
</tbody>
</table>

**EXECs**

There are no EXECs for this sample solution.

**Rules**

One message Rule for each message or message category is listed in the Invocation section.

The Rules are located in BBPARM member RULBCPSM.

**Sample infosystem management solution**

The sample solution Info/System Management helps data center personnel add problem records to the INFO database.

Due to the requirements and site-specific implementations of Info/System Management, BMC does not provide ready-to-use sample solutions. Instead, BMC distributes a sample interface to the INFO database in the BBUSER data set; the member is OSPINFO1. OSPINFO0 was created by using the OSPI application.

Refer to the *MainView AutoOPERATOR Advanced Automation Guide* for more information about the OSPI application.

The member OSPINFO1 was created from the base OSPINFO0. After the initial member was created, the input information was changed into keywords, a documentation box, and comments to explain the flow of the OSPINFO1 EXEC.

All applications require the use of a USERID and password. To eliminate hardcoded personal USERIDs and passwords, BMC recommends that data centers define a common USERID for use in logging on to INFO.
Invocation

When the Rule for IEF450I is triggered, a comparison is made against the job name contained in the text and the user specified job names.

If a match is found, an ALERT is sent to inform the operator of the failure.

The flow of IEF450I continues by calling the OSPINFO1 EXEC to generate an INFO System Management record. Upon return from OSPINFO1, a VGET is issued for the variables SYSPROG and RECNUM.

EXECs

The following list describes the EXECs for the sample Info/System Management solution:

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEF450I</td>
<td>call OSPINFO1 based on job name</td>
</tr>
<tr>
<td>OSPINFO1</td>
<td>write an Info/System Management Record</td>
</tr>
</tbody>
</table>

Rules

The following list describes the Rule for sample Info/System Management solution:

<table>
<thead>
<tr>
<th>Rule ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEF450I</td>
<td>text ID = IEF450I, distributed DISABLED</td>
</tr>
</tbody>
</table>

Sample MVALARM solution

The MVALARM solution enables you to convert alarms sent from MainView Alarm Manager to MainView AutoOPERATOR ALERTs.

The MainView AutoOPERATOR ALERTs can be viewed with the ALERT Management Facility, which makes you aware that an alarm has been generated and sent to MainView AutoOPERATOR for automation.

To ensure that only current, valid alarm information is displayed in the ALERT Management Facility, the ALERT’s priority is upgraded every 10 minutes until it
reaches CRITICAL. At CRITICAL, the ALERT Management Facility deletes the ALERT.

If the condition that created the alarm is resolved before the ALERT reaches CRITICAL, the ALERT Management Facility will delete the ALERT.

**Invocation**

The MVALARM solution uses two Rules in the AAORUL00 Rule Set: ALRMSTRT and ALRMSTOP.

**To enable the Rules**

1. Invoke the Rule Processor application by typing `RULES` from any BBI terminal session screen.

2. Ensure that the AAORUL00 Rule Set is enabled. If not, enable the Rule Set by entering the (E)nable line command next to the Rule Set name.

3. Select the AAORUL00 Rule Set.

4. Enable the ALRMSTRT and ALRMSTOP Rules by completing one of the following actions:
   - enabling the Rule on the Rule Processor Detail Control panel
   - using the Enable line command on the Rule Set Overview panel.
   - issuing the BBI commands:
     - `.T RULE,ENA,ALRMSTRT`
     - `.T RULE,ENA,ALRMSTOP`

Once the Rules are enabled, when MVALARM sends an alarm to MainView AutoOPERATOR, the Rule ALRMSTRT fires and an ALERT is created. The ALERT contains the text of the alarm and is upgraded in priority every 10 minutes.

After the ALERT reaches the CRITICAL priority, it is deleted. If the situation that caused the alarm is resolved before this point, the Rule ALRMSTOP fires and deletes the ALERT.

**To inactivate the solution**

1. Disable the Rules by completing one of the following actions:
- disabling the Rule on the Rule Processor Detail Control panel.
- using the Disable line command on the Ruleset Overview panel.
- issuing the BBI command:
  
  .T RULE,DIS,ALRMSTRT
  .T RULE,DIS,ALRMSTOP
CSM errors

This appendix includes information that you can use to diagnose CSM errors.

Creating a cross reference report

Occasionally, to diagnose the source of a problem in CSM, BMC support personnel might request a copy of your database.

This copy is used to get a better picture of your environment and determine how the objects are related to one another.

To gather the requested information, you can create a CSM database cross-reference report. This reporting facility is built into CSM dialog. When completed, the output is routed to a sequential data set as well as, if requested, a sysout class for printing.

To generate a database cross-reference report, follow these steps:

1 Enter the Xref primary command on the Global Overview panel. The following panel is displayed.

   Figure 75: CSM Diagnostic Cross-Reference panel

   BMC Software ------ CSM Diagnostic Cross-Reference ------ MainView AutoOPERATOR
   COMMAND ===>                                                      TGT --- TB62

   Generate cross-reference report:
   Repository ===> BMCTI.C720808.AOCT.BBIREPS__________________
   Group id      ===> AOCT____

   The report will be written to:
   Data Set Name ===> 'BAOTYB.CSMXREF.CSMXGRP2'___________________
   Browse report ===> Y (Y/N)
   SYSOUT class  ===> _

   To use a different data set name, enter a new name in the Data Set Name field. Remember to enclose the data set name in quotation marks.
To print the cross-reference report printed on any JES printer, specify a valid output class in the **Sysout class for printed report** field.

2 Press **Enter** to generate the report.

3 Specify **Y** in the **Browse report after generation** field to display the report for browsing.

When you are finished browsing the report data set, press **PF3/END**.

4 Write the report data set to tape and send it or the printed report or both to BMC Support.

## CSM environmental information

The following section describes how to use the EXEC CSMDIAG to gather information about the BBI-SS where CSM is running.

### CSM EXECs for diagnostics

In some cases, BMC support personnel might request information that you can obtain only with traces.

The **CSMDEBUG** EXEC provides REXX traces at two levels: EXEC or Global. These traces are produced in BBILOG.

To schedule the EXEC, enter the following command from the BBI Journal

```bash
%CSMDEBUG L( level) A( action) P(I|R) T( trace)
```

These parameters are described in the following table.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Possible values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L( level)</td>
<td>GLOBAL or EXEC</td>
<td>Specify GLOBAL to produce a trace of all the CSM EXECs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specify EXEC to produce a trace of that EXEC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifying L (GLOBAL) might produce a lot of output, especially if you use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>it with the T (TRACE) parameter.</td>
</tr>
<tr>
<td>A( action)</td>
<td>SET or UNSET</td>
<td>Specify SET to activate the trace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specify UNSET to deactivate the trace.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Possible values</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>P(I</td>
<td>R)</td>
<td>I or R</td>
</tr>
<tr>
<td>T( trace)</td>
<td>TRACEDISPLAY QUIET</td>
<td>Specify TRACE to produce a REXX trace Specify DISPLAY to produce a trace of the EXECs that have been executed. Specify QUIET to stop all trace activity at the level specified by the L (LEVEL) parameter.</td>
</tr>
</tbody>
</table>

**CSM EXEC management**

You can use the CSMDEBUG EXEC to detect the EXECs that were executed and to trace the EXECs.

Schedule the CSMDEBUG EXEC from the BBI Journal by entering

```
%CSMDEBUG L(GLOBAL) A(SET) T(DISPLAY)
```

You will receive in BBILOG a message each time an EXEC starts (ENTRY) and a message each time an EXEC stops (EXIT). The messages are time-stamped in seconds and hundredths of a second so that you can determine the exact sequence of the EXEC invocation.

You also can enter the following command to display information from CSMACT; for example:

```
%CSMDEBUG L(CSMACT) A(SET) T(DISPLAY)
```

**CSM EXEC activity**

If a problem is suspected in a specific CSM EXEC (for example, the CSMACT EXEC), you can write a trace to the BBI Journal by scheduling the CSMDEBUG EXEC with the following command:

```
%CSMDEBUG L(CSMACT) A(SET) P(I) T(TRACE)
```

A REXX trace that traces all clauses before execution (as well as shows the intermediate results during evaluation of expressions and substituted names) is written to the BBI Journal. You might be asked to send the results of this trace to BMC for debugging purposes.
### CSM EXEC traces

When the diagnostic data is generated at the global level, the debug facility should be deactivated in the following manner:

\%
CSMDEBUG L(GLOBAL) A(SET) T(QUIET)

This command causes no diagnostic data to be written to the BBI Journal for any CSM EXEC.

When the diagnostic data is generated at the EXEC level (for example, the EXEC named CSMACT), the debug facility should be deactivated in the following manner:

\%
CSMDEBUG L(CSMACT) A(SET) T(QUIET)

This command causes no diagnostic data to be written to the BBI Journal for the EXEC specified (in this example, CSMACT).

Each time CSMDEBUG executes, it issues a BBI display variable command for CSM.DEBUG, which allows confirmation of trace deactivation. If the trace is active, the entry in the profile pool contains the value TRACE along with the REXX trace option. If it is inactive, the entry contains the value QUIET.

### CSM start and stop Rules

There are two objects that you might need to create start and stop Rules for: APPC and TCPIP.

#### Creating CSM start and stop Rules for the APPC object

To create start and stop Rules for the APPC object, complete the following steps.

The CSM-generated start and stop Rules are created for APPC, but do not fire because these Rules have an IMFOJOB variable value equal to APPC as selection criteria and the IMFOJOB variable value for APPC is equal to CONSOLE.
The APPC object might be started and UP, but the actual state is never recognized by CSM as UP.

**To define the APPC object in CSM and create its start and stop Rules**

1. On the Add NEW Object panel for APPC, specify **NO** in the **Generate the Rules** field.

2. Build a start Rule by specifying **ATB007I** on the **Text Id** field of the Selection Criteria panel.

3. On the Variable Dependencies panel, specify **!CSM.ACTIVEFLAG EQ YES**.

4. On the Automation Specification panel, specify **YES** for the **Display at dest.** field, and specify **CSMUP CSMUP APPC** on the **EXEC Name/Parms** field.

5. Ensure that the Rule resides in a Rule Set other than AAORULCM and AAORULBC.

6. Build a stop Rule by specifying **ATB002I** on the **Text Id** field of the Selection Criteria panel.

7. On the Variable Dependencies panel, specify **!CSM.ACTIVEFLAG EQ YES**.

8. On the Automation Specification panel, specify **YES** for the **Display at dest.** field, and specify **CSMDOWN CSMDOWN APPC** on the **EXEC Name/Parms** field.

9. Ensure that the Rule resides in a Rule Set other than AAORULCM and AAORULBC.

Note that, any time you create a Start Rule and Stop Rule, you must not attempt to add them to Rule Sets AAORULCM and AAORULBC. The Rules can be added to any other Rule Set.

**Creating CSM start and stop Rules for the TCPIP object**

To create start and stop Rules for the TCPIP object, complete the following steps.

The CSM-generated start and stop Rules are created for TCPIP, but do not fire because these Rules have an IMFOJOB variable value equal to TCPIP as selection criteria and the IMFOJOB variable value for TCPIP is equal to CONSOLE.

The TCPIP object might be started and UP, but the actual state is never recognized by CSM as UP.
To define the TCPIP object in CSM and create its start and stop Rules

1. On the Add NEW Object panel for APPC, specify NO in the Generate the Rules field.

2. Build a start Rule by specifying EZB6473I on the Text Id field of the Selection Criteria panel.

3. On the Variable Dependencies panel, specify !CSM.ACTIVEFLAG EQ YES.

4. On the Automation Specification panel, specify YES for the Display at dest. field, and specify CSMUP CSMUP TCPIP on the EXEC Name/Parms field.

5. Ensure sure that the Rule resides in a Rule Set other than AAORULCM and AAORULBC.

6. Build a stop Rule by specifying IEF404I on the Text Id field of the Selection Criteria panel.

7. On the Variable Dependencies panel, specify !CSM.ACTIVEFLAG EQ YES.

8. On the Automation Specification panel, specify YES for the Display at dest. field, and specify CSMDOWN CSMDOWN TCPIP on the EXEC Name/Parms field.

9. Ensure that the Rule resides in a Rule Set other than AAORULCM and AAORULBC.

Note that, any time you create a Start Rule and Stop Rule, you must not attempt to add them to Rule Sets AAORULCM and AAORULBC. The Rules can be added to any other Rule Set.

Common questions and answers

This section contains, in a question and answer format, various discussions about the use of CSM. The discussions cover these topics:

- errors discovered during the starting and stopping of objects
- errors discovered during the restarting of objects
- defining special objects
- CSMACT usage
- terminal sessions
- TSO/E or environmental problems
- installation problems
- implementation problems
- other tips and hints

Starting and stopping errors

**Question:** A Normal type object has been *started* and is running. Its STATE variable is set to STARTING and its DESIRED variable is set to UP. Why is the state STARTING and not UP status?

**Answer:** Check the Start Rule for that object in Rule Set AAORULCM. It probably has not fired. CSMUP is the EXEC the Start Rule schedules and the CSMUP EXEC sets the object’s STATE variable to UP. All you need to do is correct the Rule so that it fires when the object starts.

**Question:** I am trying to define CA-TOP SECRET to CSM. The object has been defined and its Rule has been built. It starts at the scheduled time and the object comes UP. The STATE variable is set to STARTING and the Rule has not fired. In addition, an ALERT has been issued indicating a START command timeout. Why?

**Answer:** Unlike most other objects, CSM must always issue a MVS D A,TOP command for the CA-TOP SECRET object. To create an object for CA-TOP SECRET correctly, put the Started Task for CA-TOP SECRET in a member called TOP and name the procedure in this member TOP.

This process ensures that all commands issued for this object are for TOP. The IMFOJOB variable (which is used throughout CSM) is set to TOP and the dynamically generated Rule is for TOP. This process causes the Rule to fire properly, the object’s state will be UP, and the timeout does not occur.

**Question:** An object is stuck with its state as STARTING. I know it is UP and have corrected the problem in either the Rule or the object definition. How do I get the state changed to UP?

**Answer:** From the BBI Journal, schedule the EXEC CSMUP for the object as follows:

```
%CSMUP objectname
```

This EXEC changes the object state variable STATE variable to UP.
**Question**: An object is stuck with its state as STARTING, but I know the object has terminated. In addition, the object terminated before the ASOD variable was created by the START event. How do I get CSM to recognize the object as terminated?

**Answer**: When an object has a Start command specified in its definition, CSMEOM automatically recognizes any end-of-memory events that occur while the object is still in STARTING status.

However, if an object has a Start EXEC specified in its definition, you must make the following modifications to ensure that CSMEOM recognizes end-of-memory events that occur while the object is still in STARTING status.

When you are using your own Start object EXEC, insert following the Start command. For REXX EXECs:

```
"IMFEXEC SELECT EXEC(CSMASID "IMFCASID" "OBJECT") WAIT(YES)"
```

For CLIST EXECs:

```
IMFEXEC SELECT EXEC(CSMASID &IMFCASID &OBJECT) WAIT(YES)
```

**Note**

When passing special characters in the parameter list of IMFEXEC SELECT EXEC, you need to enclose the entire parameter list in single quotation marks, for example:

```
IMFEXEC SELECT EXEC('STRTCHINT TQV4MSTR EarlyUp (CSQY022I TQV4MQM Q MANAGER INITIALIZATION DONE')
```

In this example, the extra right parenthesis might interfere with parsing of the parameter list causing you to receive unexpected results.

Before calling CSMASID, ensure that the object variable is set correctly and that it contains the name of the object that was started. The IMFCASID variable is set by the object’s Start command. CSMASID returns a value of 8 in IMFRC, if it was not successful, and the messages accompanying the return code describe the error.

---

**Restart Only objects**

**Question**: How does Restart Only Control = Y work?

**Answer**: When an object is defined to CSM with the Restart Only Control field set to Y (yes), CSM does not start it automatically during IPL. The object is started and stopped externally by other means (console command, scheduler, and so on).
After it has been started the first time, the **Restart Only Control= Y** setting means CSM will control restarts for this object only if the object terminates unscheduled.

### Problems with special objects

This section describes common problems with special objects.

**Question**: I am trying to stop VTAM with CSM. I use message **IST020I** as my stop message ID but the Rule never fires and the object’s STATE variable is set to STOPPING. Why?

**Answer**: The Rule never fires because MVS and VTAM do not supply a job name for message IST020I. Since job name is part of the selection criteria for the Rule, the Rule will never fire.

The best way around this problem (other than using $HASP395 or IEF404I as the Stop text ID) is to replace the Stop command in the object definition with a short EXEC. The EXEC needs to do two things: first: it must issue the Stop command for the object with an IMFEXEC CMD statement; second, it must issue an IMFEXEC MSG statement to send the following message to the BBI Journal:

```
ACM902I objname
```

where *objname* is the object to be stopped.

Make sure an invalid Stop text ID (one that causes the Rule to never fire) is coded on the object definition panel. The ACM902I message in the Journal causes CSMDOWN to fire, using the correct object name, which ensures that the CSM variables are set properly.

**Question**: Why doesn’t CSM recognize the start and stop of the APPC object?

**Answer**: If the Start and Stop Rules have an IMFOJOB variable value equal to APPC as the selection criteria, and if the IMFOJOB variable value for APPC is equal to CONSOLE, the Rules do not fire.

To define APPC to CSM, define the APPC object to have a start and stop EXEC, and do not use a command to start and stop the APPC object. You can code the start EXEC as follows:

```rexx
/* REXX */
ADDRESS IMFEXEC
   "IMFEXEC CMD XXXX" /* XXXX IS THE START COMMAND */
   "IMFEXEC MSG 'ACM901I YYYY'" /* YYYY IS THE OBJECT NAME */
   "IMFEXEC EXIT CODE(0)"
EXIT
```

Appendix A  CSM errors 295
You can code the stop EXEC as follows:

```rexx
/* REXX */
ADDRESS IMFEXEC
  "IMFEXEC CMD XXXX"  /* XXXX IS THE STOP COMMAND */
  "IMFEXEC MSG 'ACM902I YYYY'"  /* YYYY IS THE OBJECT NAME */
  "IMFEXEC EXIT CODE(0)"
EXIT
```

The text ID ACM901I will cause the CSMUP EXEC to fire for object YYYY and text ID ACM902I causes the CSMDOWN EXEC to fire. Rules for these text IDs are in the CSM Rule Set AAORULBC.

**Question**: What is the best way to define DB2 as an object to CSM. Should all five DB2 address spaces (MSTR, DBM1, IRLM, DIST and SPAS) be defined separately to the CSM database?

**Answer**: When defining DB2 as an object to CSM, define only the dsnMSTR address space. When the dsnMSTR address space is started (and successful initialization is confirmed), dsnDBM1 and dsnMSTR (where IRLM can be any jobname) are started automatically. Whenever dsnMSTR is stopped (by way of the STOP DB2 command) dsnDBM1 and dsnIRLM also are automatically stopped.

If any of the five DB2 address spaces terminates abnormally, the resolution procedure requires that all five DB2 address spaces are taken down with the STOP DB2 command, and restarted with the START DB2 command.

Therefore, dsnMSTR should be defined as a normal, client/server object. Because dsnMSTR can provide database services for either CICS or IMS, dsnMSTR can be a server of either CICS or IMS in your system.

The start command for dsnMSTR is

```
#xDSN START DB2
```

where x is the subsystem character for DB2; for example

```
#xDB2T START DB2
```

The stop command for dsnMSTR is

```
#xDSN DB2 MODE(QUIESCE)
```

Select the Start and Stop text IDs for dsnMSTR by using the same guidelines as you would for any other object to be defined to CSM.

For example, the message DSN9022I is issued to indicate that the dsnMSTR object has successfully started and ended. dsnMSTR successfully starts but it will not be stopped by CSM unless you create special Rules to start and stop dsnMSTR objects.
You are advised to define a Rule in a Rule Set other than AAORULBC, AAORULBD, AAORULBG, or AAORULBE that triggers when the DSN9022I message is issued by dsnMSTR.

With this Rule in place, DB2 is recycled properly (and CSM variables are updated properly) in the event that any DB2 component fails.

### Defining the dsnMSTR object

This section describes how to define the dsnMSTR object in CSM.

#### To define dsnMSTR as a CSM object and create its start and stop Rules

1. On the Add New Object panel for the dsnMSTR object, specify **NO** in the *Generate the Rules* field.

2. Build a start Rule on the Selection Criteria panel by specifying DSN9022I on the *Text ID* field.

3. Fill in the Variable Dependencies panel as shown in the following figure where $x$ is the BBI-SS PAS command character and $DSN$ is the DB2 system name:

<table>
<thead>
<tr>
<th>Variable-name</th>
<th>Op</th>
<th>Variable-Value</th>
<th>OR/AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSM.ACTIVEFLAG</td>
<td>EQ</td>
<td>YES</td>
<td>AND</td>
</tr>
<tr>
<td>WORD2</td>
<td>EQ</td>
<td>$x$DSN</td>
<td>AND</td>
</tr>
<tr>
<td>WORD3</td>
<td>EQ</td>
<td>DSNYASCP</td>
<td>AND</td>
</tr>
<tr>
<td>WORD4</td>
<td>EQ</td>
<td>*START</td>
<td></td>
</tr>
</tbody>
</table>

4. On the Automation Specification panel, specify **YES** for *Display at Dest* field, and specify **CSMUP CSMUP dsnMSTR** on the *EXEC Name/Parm* field.

5. Verify that the Rule resides in a Rule Set other that AAORULBC, AAORULBD, AAORULBE, or AAORULBG.

6. Build a stop Rule on the Selection Criteria panel by specifying DSN9022I in the *Text ID* field.
7 Fill in the Variable Dependencies panel as shown in the following figure where \( x \) is the BBI-SS PAS command character and \( DSN \) is the DB2 system name:

```
BMC Software ------- Variable Dependencies--------- MAINVIEW AutoOPERATOR
COMMAND ===>                                                 TGT --- A064
Rule-set === RULDB200 Rule-id === NEWRULE1

<table>
<thead>
<tr>
<th>Variable-name</th>
<th>Op</th>
<th>Variable-Value</th>
<th>OR/AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSM.ACTIVEFLAG</td>
<td>EQ</td>
<td>YES</td>
<td>AND</td>
</tr>
<tr>
<td>WORD2</td>
<td>EQ</td>
<td>xDSN</td>
<td>AND</td>
</tr>
<tr>
<td>WORD3</td>
<td>EQ</td>
<td>DSNYASCP</td>
<td>AND</td>
</tr>
<tr>
<td>WORD4</td>
<td>EQ</td>
<td>*STOP</td>
<td></td>
</tr>
</tbody>
</table>
```

8 On the Automation Specification panel, specify YES for Display at Dest field, and specify `CSMDOWN CSMDOWN dsnMSTR` on the EXEC Name/Parm field.

**CSMACT usage**

This section describes common problems with the CSMACT EXEC.

**Question:** How do I return control of an object’s schedule to CSM?

**Answer:** The RESET command can be used to return control of a parent and its dependents to CSM control.

When using the CSM panel, use the `rese(T)` line command and be sure to specify a scope of C for children. From the console or EXCP/VTAM sessions, schedule the CSMACT EXEC as follows:

```
CSMACT object RESET DEP
```

where `object` is the name of the CSM controlled object.

**Question:** I have tried to stop the MVS object when trying to bring the system down and have received this ALERT message:

```
ACM705A Please verify DOWN request for MVS
```

I know this means that a dependent of MVS has Verify Force Down set to YES, but is there a way around this ALERT?
**Answer:** If you shut down the MVS object using the CSMACT EXEC using the following format:

```
CSMACT MVS MAEARLY DEP X
```

the Verify Force Down option for the MVS dependent is ignored for this execution of CSMACT. ALERT ACM705A is not issued and all of the dependents of MVS will (assuming they have been properly defined) come down.

**Question:** I am trying to execute CSMACT by scheduling it through a Rule. The Rule fires and CSMACT is scheduled. The problem I have is that CSMACT appears to have not affected the object specified.

The following parameters were entered:

```
CSMACT object UPEARLY NOREQ
```

**Answer:** CSMACT expects the first parameter passed to it to be the EXEC name. When scheduled by a Rule, CSMACT does not get the EXEC name as the first parameter because of the difference in EXEC scheduling between the Rules Processor and other EXEC submission vehicles. See the *MainView AutoOPERATOR Advanced Automation Guide* for more information.

To make CSMACT work properly when scheduled through the automation action panel, enter its parameters:

```
CSMACT CSMACT object UPEARLY NOREQ
```

**Question:** Explain how the `REQ|NOREQ` and `DEP|NODEP` parameters should be used when CSMACT is scheduled manually.

**Answer:** Use `REQ|NOREQ` whenever an UP command needs to be processed (for example, UPEARLY, UP, and so on).

Specifying `REQ` causes the parents of the object named in the CSMACT parameter list to be checked and, if necessary, brought up before the object named is started. `NOREQ` causes the check not to be performed.

Use `DEP|NODEP` whenever a DOWN command is to be processed (for example, DOWNEARLY, DOWN, and so on). Specifying `DEP` causes the children of the object named in the CSMACT parameter list to be checked and, if necessary, stopped before the object named is stopped. `NODEP` causes the check not to be performed.

Either `DEP|NODEP` or `REQ|NOREQ` can be used with a RESET or RESETC command. If `DEP` is used with a RESET|RESETC command, the named object, along with its children, is affected. If `REQ` is used with a RESET|RESETC command, the named object, along with its parents, is affected.
If neither DEP|NODEP nor REQ|NOREQ is specified in the CSMACT parameter list, only the named object is affected.

**Question:** I am trying to move an object and all of its dependents from one CSM group to another. How can I do that using CSMACT?

**Answer:** Do this process the same way that you would use CSMACT to move a single object from one CSM group to another. Schedule the CSMACT EXEC in the following way:

```
CSMACT object MOVE target_group DEP
```

CSM determines the names of the dependents of the object named in the CSMACT parameters. CSM then finds the first dependent that does not have an object depending upon it. This is the bottom of the dependency structure and CSM stops that object. When the object is down, CSM stops the parent of that object (and so on) until reaching the object named in the parameters.

---

**BBI terminal sessions issues**

**Question:** Why are the CSM dialog panels not displayed when I select it? No error message is generated.

Why?

**Answer:** Check the BBITSP00 member in BBPROF. If `ISPF=NO` is specified, the terminal session (even though ISPF is engaged in the TSO user address space) runs in TSO mode.

The terminal session must be in ISPF mode for the CSM dialogs to function. Set `ISPF=YES` in the BBITSP00 member.

---

**TSOE or environmental problems**

**Question:** When I attempt to initialize CSM, CSMBUILD issues message:

```
```

What do I do?

**Answer:** If the ALLOCATE command processor program is moved from SYS1.CMDLIB to SYS1.LPALIB, this error occurs. To correct the problem, either move the ALLOCATE module back to SYS1.CMDLIB or add a STEPLIB to the BBI-SS for SYS1.LPALIB.
CSM installation issues

**Question:** I am converting to CSM version 3.1 from CSM version 1.1.

When I start CSM version 3.1 for the first time, a panel is displayed that indicates that the format of the CSM database will be changed.

The BBPARM data set and the target subsystem specified are from a previous release. Where is this information coming from?

**Answer:** EXEC CSMSPF01 is taking the target ID and the BBPARM data set name from variables in the ISPF profile pool by way of an IMFEXEC VGET.

To make sure CSM uses the correct target ID and BBPARM data set when the database is saved, reply GO to the conversion panel. When CSM displays the object list, issue the PROFILE primary command. At this point, the target subsystem and BBPARM data set used by CSM can be changed.

Other tips and questions

**Question:** Instead of simply restarting a CSM managed object following an abend, is there some way to have a recovery procedure executed prior to restarting the object?

**Answer:** CSM does not distinguish between a normal and abnormal termination for an object. However, it is possible to implement a recovery process with an EXEC that performs the start in conjunction with additional Rules or EXECs.

One approach is to create a Rule for message IEF450I to capture the abend/termination status and store it in a variable in the shared pool called USER.object.TERMTYPE.

The start EXEC for the object would use this variable to determine whether or not the recovery procedure is needed. Following an abend, the start EXEC initiates the appropriate recovery process, after which the failed object is started. But following a normal termination for the object, the start EXEC would simply start the object.

**Question:** I cannot access a terminal session and I need to perform a SYNC. Can I perform this from a console?

**Answer:** Yes. To issue a SYNC, two EXECs must run one after the other. First, execute CSMSTAT. After CSMSTAT is complete, execute CSMSCAN. Neither EXEC requires input parameters.

**Question:** How can I deactivate CSM?
**Answer:** Schedule EXEC CSMDEL. It will set the CSM.ACTIVEFLAG variable to NO, which will prevent CSM processing from taking place.

**Question:** I used to see CSM messages being written to the BBI Journal (for example, ACM001I, ACM003I, ACM104I, ACM986I, and ACM987I). Now, I do not see them anymore. Why?

**Answer:** Rules in the Rule Set AAORULBC intercept ACM messages written to the journal. If the AUDIT parameter in BBIPARM member BBISSP00 is set to YES, the messages are written to the BBI Journal. If the parameter is set to NO, the messages are not written to the Journal (the only exception to this Rule is message ACM488I).

**Question:** When is the Start Command limit reset?

**Answer:**
- at midnight each day
- by invoking the CSMACT EXEC with command RESETC for the object
- by successfully starting the object and its actual state being set to UP
- at IPL

**Question:** How do I start or stop an object from an operator’s console and still keep the object under CSM control?

**Answer:** Build two Rules in a Rule Set other than AAORULBC or AAORULCM. For one, set the selection criteria to

```
text ID = P
Event type = CMD
Variables = &IMFOJOB/ NE &QSSNAME/ AND !CSM.WORD2.STATE NE ''
```

Set the action to

```
Reject Cmd. = Y
EXEC = CSMACT . &WORD2 DOWNEARLY NODEP
```

For the other, set the selection criteria to

```
text ID = S
Event type = CMD
```

---

302  *MainView AutoOPERATOR Solutions Guide*
Variables = &IMFOJOB/ NE &QSSNAME/ AND !CSM.WORD2.STATE NE ''

Set the action to

Reject Cmd. = Y

EXEC = CSMACT . &WORD2 UPEARLY NOREQ

These Rules allow the object to be either started or stopped from the console with the normal Start or Stop commands, but still allows CSM to function properly.

If DB2 is an object in the CSM database, the Rule should be modified to not fire for START DB2 commands. DB2 requires that the START command for the internal jobs (MSTR, DBM1) are issued internally and not issued by CSM.

To modify the Rule, change

VARIABLES=&IMFOJOB/ NE &QSSNAME/ AND !CSM.WORD2.STATE NE"

to

VARIABLES=&IMFOJOB/ NE &QSSNAME/ AND !CSM.WORD2.STATE NE" AND WORD2 NE DB2

Maximize CSM implementation

**Question:** I regularly shut down an object (for example, CICS) at a scheduled period each day.

Batch maintenance processing for that object takes place during that shutdown period (for example, file backups or database reorganizations). I want to have the object automatically start back up when the maintenance processing is complete. What’s the best way to do this?

**Answer:** Add a step to the end of the batch maintenance job to run IMFSUBEX. Use this additional step to schedule CSMACT for the object.

See the *MainView AutoOPERATOR Advanced Automation Guide* for information on IMFSUBEX usage.

Schedule the CSMACT EXEC by entering

**CSMACT object_name UPEARLY NOREQ**

Make sure you specify UPEARLY and not UP.
**Question:** I need to bypass MainView AutoOPERATOR and CSM control of the system startup on occasion. How can I accomplish this?

**Answer:** CSM startup is triggered by message PM0010I. When that message is issued, it schedules the EXEC ACMM000. ACMM000 calls CSMINIT and so on.

To bypass AO/CSM control of the system, you could write a small EXEC that based on the reply to a WTOR, schedules or does not schedule ACMM000 with an IMFEXEC SELECT EXEC statement; for example:

```rexx
/* REXX */

"IMFEXEC WTOR 'DO YOU WANT CSM TO START THE SYSTEM?  REPLY "Y" TO', "START.' WAIT(60) REPLY(REP)"
IF IMFCC = 8 THEN DO
  "IMFEXEC WTO 'TIME HAS BEEN EXCEEDED.  CSM WILL NOT START SYSTEM.'"
  "IMFEXEC EXIT CODE(0)"
  EXIT
END

IF REP = 'Y' THEN DO
  "IMFEXEC SELECT EXEC (ACMM000)"
END
ELSE DO
  "IMFEXEC WTO 'CSM WILL NOT START SYSTEM.'"
END

"IMFEXEC EXIT CODE(0)"
EXIT
```

After writing this EXEC (or something similar), change Rule ID ACMRC005 in Rule Set AAORULBC to schedule this new EXEC instead of ACMM000.

Remember, anytime a PTF performs maintenance against AAORULBC, Rule ID ACMRC005 must be changed to call the EXEC, instead of ACMM000.
CSM panels

This section contains the CSM panels and descriptions for the line commands, primary commands and fields.

CSM Global Overview panel

The following shows an example of the Global Overview panel for CSM.

**Figure 76: Global Overview panel**

```
BMC Software ------------ Global Overview ------------ MainView AutoOPERATOR
COMMAND ===>                                                   SCROLL ===> CSR
TGT ---> KMZ1

Primary Commands: EXCeption CANcel UNsort CONVERT SAVERULE SHUTSYS
CALENDAR CMDSHOW Xref BUILD

LC CMDS -------- (S)tart b(O)unce rese(T) depe(N)dents (A)dd
                (EX)ec (E)dit (L)ients (D)elete
                (M)ove (R)epeat (B)rowse (S)ync
                (E)nable (D)isable

LC Object Group System Actual Desired Ind Type
------- ---- --- ---- ---- ---- ---- ---- ---- ---- ----
         ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___
___ MVS
___ AAOCSM01 KMZ1 SJSD UP UP NORM
___ AAOCSM01 KMZ5 INACTIVE INACTIVE NORM
___ AAOCSM02 KMZ1 SJSD INACTIVE INACTIVE TRAN
___ AAOCSM02 KMZ2 INACTIVE INACTIVE TRAN
___ AAOCSM02 KMZ5 INACTIVE INACTIVE TRAN
___ AAOCSM08 KMZ1 SJSD INACTIVE INACTIVE NORM
___ AAOCSM08 KMZ2 INACTIVE INACTIVE NORM
___ AAOCSM08 KMZ5 INACTIVE INACTIVE NORM

To view additional information about each object, scroll left by pressing **PF10/PF22**.

**Figure 77: Global Overview panel: scrolled LEFT**

```
Appendix B  CSM panels 305
```
The following table describes the fields for the CSM Global Overview panel.

### Table 23: Field descriptions: CSM Global Overview panel

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>contains the name of the object as specified in the object definition. Objects are listed in alphabetical order.</td>
</tr>
<tr>
<td>Group</td>
<td>is the name identifying a group of objects managed by Continuous State Manager. This name must be unique among the other members of the CSM-Plex to ensure proper object status reporting on the Global Overview display.</td>
</tr>
<tr>
<td>System</td>
<td>is the OS/390 SYSNAME where the group is enabled. If the group is disabled, this column is blank.</td>
</tr>
<tr>
<td>Actual</td>
<td>indicates the object’s current status in the system. This state should always match the desired state. When the actual and desired state do not match, CSM attempts to make them match.</td>
</tr>
<tr>
<td>Desired</td>
<td>indicates the object’s desired state which is governed by its schedule. This state should always match the actual state. When the actual and desired state do not match, CSM attempts to make them match.</td>
</tr>
</tbody>
</table>
| Ind     | indicates object relationships. Possible values are as follows: 
L indicates the object has clients. 
N indicates the object has children (dependents). 
X indicates the object has an outstanding ALERT associated with it. To view the ALERT and its contents, you must use the ALERT Management Facility. 
E indicates that the outstanding ALERT associated with the object has a follow-up EXEC. You can schedule the follow-up EXEC by entering the e(X)ec line commands next to the object in the LC column. |
| Type    | shows the type of object: possible values are NORM (normal object) and TRAN (transient object). |
| Description | shows a description of the object as specified in the object definition. |
| Id      | shows the TSO ID of the last person who modified the object. |
The following table describes the primary commands for the CSM Global Overview panel.

Table 24: Primary commands: CSM Global Overview panel

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>**Exception</td>
<td>EXC**</td>
</tr>
<tr>
<td><strong>Group</strong></td>
<td>displays the Object Groups application, where you can manage CSM Groups</td>
</tr>
<tr>
<td>**Sort</td>
<td>SORT**</td>
</tr>
<tr>
<td>**Save</td>
<td>SAVE**</td>
</tr>
<tr>
<td>**Add</td>
<td>ADD**</td>
</tr>
<tr>
<td>**Calendar</td>
<td>CAL**</td>
</tr>
<tr>
<td>**Cancel</td>
<td>CAN**</td>
</tr>
<tr>
<td>**Unsort</td>
<td>UN**</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>CONVERT</td>
<td>converts CSM database from MainView AutoOPERATOR release 4.1 or 5.1 into a new group with objects in the CSM repository</td>
</tr>
</tbody>
</table>
| Build | saves your parameter changes in the CSM database and automatically issues a command that tells CSM to reload the control parameters and make the changes effective immediately  
This command requires the GROUPNAME parameter. When you issue this command, a confirmation panel appears, requesting verification.  
BUILD makes changes effective immediately. |
| Locate | finds a specified object on the CSM display  
**Format:** `L object name` |
| CMDSHOW | enables you to control the display of the Primary and Line commands  
Issuing the command causes the application to be displayed without the Primary and Line commands.  
Issuing CMDSHOW again reverses the change and re-displays the Primary and Line commands. |
| Xref | invokes the CSM Diagnostic Cross-Reference facility to define a cross-reference of the CSM database  
The cross-reference report is written to a specified data set from which the report can be browsed or printed. To obtain a printed copy of the report:  
1. Change the default value for Browse from `Y` to `N`.  
2. Specify the SYSOUT class; for example:  
3. `Data Set Name==>'ABC.CSMXREF.RS31'`  
4. `Browse report after generation (Y/N)? N`  
5. `SYSOUT class for printed report? R`  
6. Press **END** to generate or print the cross-reference report. |
| SAVERULE | generates the CSM Ruleset that is defined for the specified group  
The command requires the group name as a parameter. |
| SHUTSYS | shuts down all the objects in a group with a single command  
The command requires that a group name is specified and can have an object as an optional parameter. The default for object is OS/390; therefore, if an object is not specified, all objects from the group are stopped. If an object other than OS/390 is specified, the shutdown occurs for this object and all of its children (dependents).  
**Format:** `SHUTSYS groupid object`  
**Example:**  
`SHUTSYS PROD - CSM shuts down all objects in group PROD`  
`SHUTSYS PROD TSO - CSM shuts down TSO and all of its children` |
The (EXC)ception and (MON)itor primary commands toggle the display between the CSM Global Overview panel and the CSM Exceptions panel. On the CSM Global Overview panel, the (EXC)ception command is shown. On the CSM Exceptions panel, the (MON)itor command is shown.

The following table describes the line commands for the CSM Global Overview panel.

Table 25: Line commands: CSM global overview panel

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>S</td>
</tr>
</tbody>
</table>

Starts an object

CSM uses this command or the EXEC that is specified for this object. Before starting this object, CSM prompts you with three verification fields:

Please confirm START for object

? N (Y/N)

START all PARENTS of object

? Y (Y/N)

START now, but STOP as scheduled:

? N (Y/N)

If an object must start without starting its parents, specify N in the second verification field.

If TSO/E and ISPF are not active and an object must be started, use the CSM EXEC, CSMACT.

If the Start or Stop command needs to be issued in response to an outstanding WTOR, you can use the EXEC CSMWTOR supplied by BMC. The CSMWTOR EXEC works only in a JES2 environment. This EXEC has two parameters: object name and reply. The syntax is as follows:

%CSMWTOR object_name reply_text

CSMWTOR can be defined in the Command Specifications panel as the Start or Stop command.

This EXEC locates the outstanding reply from the object and responds to it with the reply text that you specified.

This command takes effect immediately.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounce</td>
<td>causes CSM to issue a stop command for the selected object and then restart the object immediately if the object’s schedule indicates that the object should be active. All other control parameters for the object remain in effect. The b(0)unce command lets you recycle a task without modifying the normal control parameters for the object. The b(0)unce command affects only the selected object. It does not affect the object’s parents or children. This command takes effect immediately.</td>
</tr>
</tbody>
</table>
| Reset    | causes CSM to restore the object to its defined schedule. You might use this command if you have used the (S)tart or sto(P) line commands to start and stop objects. Before resetting this object, CSM prompts for the following verifications:  
Please confirm RESET for object  
? N (Y/N)  
RESET command count for object  
? N (Y/N)  
Scope of RESET of :pv.object:epv.  
? S (Single/Parent/Children)  
The third verification field provides you with three options:  
S returns object to its defined schedule.  
P returns object and its parents to their defined schedules.  
C returns object and its children to their defined schedules.  
The rese(T) line command clears the status of an object in a manual state and returns it to CSM control.  
To reset all objects to their normal schedules, reset OS/390 and specify C as the scope.  
Rese(T) takes effect immediately. |
| Dependents | displays the objects that are defined as being dependents (children) of the selected object. In a parent-child relationship, the parent object must be active for the child object to be active. Therefore, if a parent object is taken down, all dependent objects are taken down first.  
If the selected object has no dependents defined, the Dependents of object panel contains no data. |
<p>| Add      | adds objects to the CSM database. The added object automatically assumes the schedule of the object on which the (A)dd line command was performed and assumes the selected object as a required object. This command invokes the Object Detail Control panel. (A)dd takes effect after you issue the BUILD command. |</p>
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>**Stop</td>
<td>P**</td>
</tr>
<tr>
<td>**Exec</td>
<td>EX**</td>
</tr>
<tr>
<td>**Edit</td>
<td>E**</td>
</tr>
<tr>
<td>**Clients</td>
<td>L**</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Delete | eliminates the object from CSM control  
   The selected object must be managed through some other means after this command is issued.  
   If you attempt to (DE)lete an object that has children, CSM does not allow deletion of the object. Do one of the following actions to delete a parent object:  
   - Delete the child objects and then delete the parent object.  
   - Redefine the parent-child dependency relationship to be a client-server relationship.  
   - Define the child objects to another parent.  
   (DE)lete modifications take effect after you issue the BUILD command. If you delete an object by mistake, issue the CANCEL primary command. |
| Cancel | causes CSM to issue the (CAN)cel command for the selected object  
   If the definition does not have a (CAN)cel command, the sto(P) command is issued. This command does not affect the object’s parents or children.  
   Before canceling the object, CSM prompts you for verification:  
   Please confirm Cancel for object  
   Cancel?? N (Y/N)  
   This command takes effect immediately. |
| Move | enables you to move CSM objects from the local BBI-SS PAS to a remote BBI-SS PAS where CSM is installed |
| Repeat | enables you to duplicate an existing object’s definition to define a new object with the same information  
   Use this command when you want to define objects with similar attributes. The only limitation is that no two objects can have the same name. |
| Browse | enables you to browse the control parameters for the selected object  
   The control parameters cannot be modified with this line command. The (B)rowse command displays the Object Detail Control panel. |
| Synchronize | checks for discrepancies between the current state and the physical state of an object  
   For example, an object could change state without issuing the expected start/stop messages, so its current and physical states would become unsynchronized.  
   When you issue this command, CSM checks every defined object to see if it is physically present on the system. After checking for the presence of an object, CSM updates the object’s current state variable.  
   This command does not change the status of objects that are in a manual state. |
| ENable | enables CSM's control over the object  
   CSM controls starting or stopping this object.  
   Issuing the ENable line command enables the object immediately. |
### Command Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| DISable | disables CSM’s control over the object  
CSM does not start or stop this object. CSM ignores this object if it is part of the dependency chain of another object that CSM is starting or stopping.  
The object will remains disabled until the object is enabled. |

---

## Defining and modifying object panels

To define a new object to CSM, enter the A(dd) line command in the LC field next to the object that will be the new object’s parent, or enter the ADD primary command on the Global Overview panel.

The Object Detail Control panel is displayed and it links together all the panels that you use to define an object to CSM. The panels are as follows:

- Object Detail Control
- Command Specification
- Conditional Commands
- Event Specification 1
- Event Specification 2
- User Notification
- Object Groups
- Object Requisites
- Client/Server Relationships
- Object Scheduling

You can use the following shortcut commands on the COMMAND line from any of the panels in the table to display another panel.

### Table 26: Shortcut commands: Object Detail Control panel

<table>
<thead>
<tr>
<th>Command</th>
<th>Displays this panel</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMD</td>
<td>Command Specifications</td>
</tr>
<tr>
<td>EV1</td>
<td>Event Specifications 1</td>
</tr>
</tbody>
</table>
The following sections list each of these panels and their field, primary and line command descriptions.

### Object Detail Control panel

The following is an example of the Object Detail Control panel.

**Figure 78: Object Detail Control panel description**

<table>
<thead>
<tr>
<th>BMC Software</th>
<th>Object Detail Control</th>
<th>Add</th>
<th>MainView AutoOPERATOR</th>
<th>COMMAND</th>
<th>TGT</th>
<th>KMZ1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following options are displayed in sequence, or may</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>be selected by entering the three-character code</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMD - Command Specifications</td>
<td>EV1 - Event Specifications 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EV2 - Event Specifications 2</td>
<td>NOT - User Notification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRP - Object Groups</td>
<td>REQ - Object Prerequisites</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLS - Client/Server Relationships</td>
<td>SCH - Object Scheduling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object Name</td>
<td>===&gt; AA0CSM01</td>
<td>Member of Group</td>
<td>===&gt; KMZ1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object Type</td>
<td>===&gt; NORM</td>
<td>Type of object ( ? for list)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify Force Down</td>
<td>===&gt; N (Y/N)</td>
<td>Restart Only Control</td>
<td>===&gt; N (Y/N)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Information:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Author</td>
<td>===&gt; BA0KMZ1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>===&gt; Child of MVS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last Modified by BA0KMZ2 on 01/30/2001 at 15:43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enter END command to process and return, ENTER to continue, or CANcel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following table describes the fields for the Object Detail Control panel.

**Table 27: Field descriptions: Object Detail Control panel**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Name</td>
<td>enter the name of the object</td>
</tr>
<tr>
<td></td>
<td>You can specify a name up to 8-alphanumeric characters representing any valid job name.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Member of Group       | is the name identifying a group of objects managed by Continuous State Manager  
This name must be unique among the other members of the CSM-Plex to ensure proper object status reporting on the Global Overview display.  
**Format:** 8-character, alphanumeric  
**Default:** None |
| Object Type           | is the type of the object, either Normal or Transient  
A Normal object has a schedule based on the time-of-day and the day-of-week. A Transient object is started once after the IPL of the system and then its state is not managed by CSM.  
You can enter a question mark (?) in this field to see a list of valid object types.  
**Format:** NORM or TRAN. Default is NORM |
| Verify Force Down     | prevents accidental stopping of the object by requiring command verification  
**Format:** Y or N  
Y: CSM automatically asks for the stop request to be verified. You must reply to ALERT ACM705A before CSM will stop the object.  
N: CSM stops the object without confirmation. |
| Restart Only Control  | specifies whether or not CSM waits for the scheduling product to start the object  
Most sites have a scheduling package that manages work based on a time schedule. If a scheduling package is already managing the schedule of an object and you do not want to move control of that object to CSM, you can specify the CSM manages that object only for restart control with this parameter.  
When **YES** is specified, CSM waits for the scheduling product to start the object. If the scheduling product does not start the object at the desired time, CSM issues ALERT messages, indicating that the object is scheduled to be started but it has not started.  
Once started by the scheduling product, CSM manages the availability of the object by automatically restarting the object if it ends out of schedule.  
Even though CSM is not responsible for starting the object, you must still specify the complete schedule for the object for this parameter to perform correctly.  
**Format:** Y or N  
Y: CSM manages only restart functions.  
N: CSM starts, stops, and restarts the object.  
Objects that are defined as Restart Only can be manually started or stopped one of the following ways:  
- By using the MVS START or STOP command  
- By issuing the UP or DOWN command with the CSMACT EXEC  
- By changing the object’s definition with the object definitions panels. |
| Author                | enter the TSO ID of the person who modified the object  

---

Appendix B  CSM panels 315
Command Specifications panel

The first panel displayed from the Object Detail Control panel is the Command Specifications panel.

Figure 79: Command Specification panel description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>enter text to describe this object</td>
</tr>
<tr>
<td></td>
<td>The description should be meaningful to the viewer. You can provide the normal</td>
</tr>
<tr>
<td></td>
<td>system in which the object runs, what the object is, or some indication of</td>
</tr>
<tr>
<td></td>
<td>the dependencies for the object.</td>
</tr>
<tr>
<td></td>
<td><strong>Format</strong>: 1 to 40 alphanumeric characters of free-form text. The text</td>
</tr>
<tr>
<td></td>
<td>should not contain quote marks.</td>
</tr>
<tr>
<td></td>
<td><strong>Example</strong>: IRLM for IMSPROD1 on SYSB</td>
</tr>
<tr>
<td></td>
<td>The description is not used for any object management function.</td>
</tr>
</tbody>
</table>

The following table describes the fields for the Command Specifications panel.
Table 28: Command Specification panel fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Start Command/EXEC** | Specify the start command or EXEC to start this object. If you specify a command, enter the same command used to start the object from the OS/390 console. Prefix the Start command with the appropriate command recognition character. For example, use a pound sign (#) as a prefix for OS/390 commands: 
#S JES2 
If the command has any embedded quotation marks, you must use two single quotation marks on each side of the command specified in the Start Command/EXEC field; for example: 
#S ACF,PARM="data"
If you use an EXEC instead of a command, do not prefix the EXEC name with any command characters. Parameters can be passed to the EXEC; for example, to start IMS with a warm start, specify the EXEC as follows: 
STARTIMS PROD1 WARM
When the EXEC STARTIMS is scheduled, it will be passed the two parameters PROD1 and WARM.
The EXEC name is limited to eight characters and must be a valid PDS member name. The EXEC must reside in a library in the SYSPROC concatenation of the MainView AutoOPERATOR subsystem address space. |
| **User Post Start EXEC** | Specify the name of an optional, user-specified EXEC that CSM will schedule when CSM receives an object’s start message. When CSM receives the object’s start message, the parameters CSM passes to the User Post Start EXEC include
- Post Start EXEC name
- Object name
- Object status (normal, forced, or abnormal up/down)
- Start message text ID
- Remaining words in start message
The User Post Start EXEC must reside in a library in the SYSPROC concatenation of the MainView AutoOPERATOR subsystem address space. |

Note: A Start Command/EXEC is not required for transient objects.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Stop Command/EXEC**  | Specify the stop command or EXEC to stop this object normally. If you specify a command, enter the same command used to end the object from the OS/390 console. Prefix the Stop command with the appropriate command recognition character. For example, use a pound sign (#) as a prefix for OS/390 commands: **#F CICSPAY,CEMT P SHUT**  
If the command has any embedded quotation marks, you must use two single quotation marks on each side of the command specified in the Stop Command/EXEC field.  
If you use an EXEC instead of a command, do not prefix the EXEC name with any command characters. Parameters can be passed to the EXEC in the same way as with a startup EXEC.  
The EXEC name is limited to eight characters and must be a valid PDS member name. The Stop EXEC must reside in a library in the SYSPROC concatenation of the MainView AutoOPERATOR subsystem address space. The Cancel command/EXEC is automatically issued if the Stop command/EXEC times out. See below. |
| **User Post Stop EXEC** | Specify the name of an optional, user-written EXEC that CSM schedules when CSM receives an object’s stop message. The User Post Stop EXEC can be used to call batch work from the active scheduling product, update or define incident tracking system tickets, or trigger other events that should occur after the object has ended. The parameters passed to the User Post Stop EXEC are the same as the one you can use for the User Post Start EXEC. The User Post Stop EXEC must reside in a library in the SYSPROC concatenation of the MainView AutoOPERATOR subsystem address space. |
| **Cancel Command/EXEC** | Specify the command or EXEC to cancel this object. If you specify a command, enter the same command used to cancel the object from the OS/390 console. Prefix the Cancel command with the appropriate command recognition character. For example, use a pound sign (#) as a prefix for MVS commands: **#Z NET,CANCEL**  
If the command has any embedded quotation marks, you must use two single quotation marks on each side of the command specified in the **Cancel Command/EXEC** field.  
If you use an EXEC instead of a command, do not prefix the EXEC name with any command characters. Parameters can be passed to the EXEC.  
The Cancel EXEC must reside in a library in the SYSPROC concatenation of the MainView AutoOPERATOR subsystem address space. The Cancel command/EXEC is automatically issued if the Stop command/EXEC times out. |

**Note:** A Stop Command/EXEC is not required for transient objects.
### Field | Description
--- | ---
Recovery Command/EXEC | Specify the command or EXEC name of an optional, user-specified EXEC that CSM schedules when CSM receives an object’s abnormal termination message. If you specify a command and the command has any embedded quotation marks, you must use two single quotation marks on each side of the command specified in the Recovery Command/EXEC field. When CSM receives the object’s abnormal termination message, the parameters CSM passes to the Recovery Command/EXEC include
- Recovery EXEC name
- Object name
- Object status (which is ABENDDOWN)
- Abnormal termination message text ID
- Remaining words in the message
The Recovery EXEC must reside in a library in the SYSPROC concatenation of the MainView AutoOPERATOR subsystem address space.

Start Command Time Out | Enter a number from 1 to 1440 (in minutes) that CSM waits to verify the Start commands. Rather than just changing the object’s state when a Start command is issued, CSM waits for the confirmation message to indicate that the Start command was processed properly. The Start Command Time Out parameter indicates to CSM how long it should wait for this confirmation message before posting an ALERT to the ALERT Management Facility to report the failed command. **Format:** mmmm, where m=minutes **Range:** 1 through 1440 minutes

Stop Command Time Out | Enter a number from 1 to 1440 (in minutes) that CSM waits to verify the Stop commands. Rather than just changing the object’s state when a Stop command is issued, CSM waits for the confirmation message to indicate that the Stop command was processed properly. The Stop Command Time Out parameter indicates to CSM how long it should wait for this confirmation message before posting an ALERT to the ALERT Management Facility to report the failed command. **Format:** mmmm, where m=minutes **Range:** 1 through 1440 minutes If you define a Cancel command/EXEC for an object, CSM automatically issues this command when the Stop command/EXEC reaches the time out limit set in this field.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start Command Limit</strong></td>
<td>Specify how many times during a 24-hour period that CSM will automatically issue the start commands for an object.</td>
</tr>
<tr>
<td></td>
<td>For example, you can specify that CSM will automatically attempt restarts for three abends of an object. If the restart fails, an ALERT is issued.</td>
</tr>
<tr>
<td></td>
<td>The count of retries is reset when the object stops normally, at midnight every night, and when you issue the line command Reset.</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td>■ If you specify 0, CSM does not ever attempt to start the object.</td>
</tr>
<tr>
<td></td>
<td>■ If you specify 1, CSM attempts to start the object once with no retries.</td>
</tr>
<tr>
<td></td>
<td>■ If you specify 4, CSM attempts to start the object once and then tries three more times if the object terminates unexpectedly.</td>
</tr>
<tr>
<td></td>
<td>If restarting fails at any attempt, an ALERT is issued.</td>
</tr>
<tr>
<td></td>
<td>After unsuccessfully restarting the object the specified number of times, CSM generates an ALERT to the ALERT Management Facility that the number of retries for the specified object has been reached. An EXEC associated with the ALERT will start the object. No further restarts can be attempted until the start command counter is reset.</td>
</tr>
<tr>
<td></td>
<td><strong>Format:</strong> xxxx for 4 positions</td>
</tr>
<tr>
<td></td>
<td><strong>Range:</strong> 0 to 9999</td>
</tr>
<tr>
<td><strong>Start Limit Reset</strong></td>
<td>Prevent an abnormally terminating object from endlessly restarting and terminating CSM; holds a Start Command Limit for the object.</td>
</tr>
<tr>
<td></td>
<td>When the object is started an internal command count for the object is compared to the Start Command Limit value. If the value is exceeded, the object is not restarted.</td>
</tr>
<tr>
<td></td>
<td><strong>Format:</strong> Y or N</td>
</tr>
<tr>
<td></td>
<td>Y: If the object is in an UP state or is a transient, its command count will be set to 1. Otherwise the command count will be set to 0. The count will be set each day at midnight.</td>
</tr>
<tr>
<td></td>
<td>N: Do not reset the command count for this object at midnight.</td>
</tr>
<tr>
<td><strong>Conditional Start</strong></td>
<td>Specify additional object Start commands.</td>
</tr>
<tr>
<td></td>
<td>Use conditional commands when an object requires more than one command in order to be started or a user wants to issue an additional command if the object does not start within a prescribed period of time.</td>
</tr>
<tr>
<td></td>
<td><strong>Format:</strong> Y or N</td>
</tr>
<tr>
<td></td>
<td>Y: Allow user to specify Conditional Start commands.</td>
</tr>
<tr>
<td></td>
<td>N: Do not allow user to specify Conditional Start commands.</td>
</tr>
</tbody>
</table>
# Conditional Command Specifications panel

If you entered **Y** for either the **Conditional Start** or the **Conditional Stop** fields on the Command Specifications panel, the Conditional Commands Specification panel is displayed.

The following figure shows the Conditional Command Specifications commands.

**Figure 80: Conditional Command Specifications panel: start**

![Conditional Command Specifications panel](image)

The following shows an example of the Stop commands panel.

**Figure 81: Conditional Command Specifications panel: stop**

![Conditional Command Specifications panel](image)

## Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional Stop</td>
<td>Specify additional object Stop commands. Use conditional commands when an object requires more than one command in order to be stopped or a user wants to issue an additional command if the object does not stop within a prescribed period of time.</td>
</tr>
</tbody>
</table>

**Format:** **Y** or **N**

**Y:** Allow user to specify Conditional Stop commands.

**N:** Do not allow user to specify Conditional Stop commands
The following describes the fields for the Conditional Commands Specifications panels.

Table 29: Conditional Commands Specifications panel fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>After ____ minutes</td>
<td>Specify a number of elapsed minutes before the Conditional Commands should be used.</td>
</tr>
</tbody>
</table>
| Start Command/EXEC     | Specify the name of the Conditional Command / EXEC to be submitted for execution if the object start event is not detected and the Conditional Command Wait Time value is exhausted. For example, you can enter a command

#S IMSCNTL1

If this command is an MVS command, the command must be preceded by the MVS command recognition character (#). Always use the pound sign for MVS commands. Maximum number of characters for commands submitted to MVS is 126.

You can also enter an EXEC name: EXECNAME, which is the name of the EXEC used to perform a series of commands for the start up of this task. The EXEC must be in a library included in the SYSPROC concatenation of the MainView AutoOPERATOR subsystem address space. |
| Stop Command/EXEC      | Specify the name of the Conditional Command / EXEC to be submitted for execution if the object stop event is not detected and the Conditional Command Wait Time value is exhausted. For example, you can enter a command

#P IMSCNTL1

If this command is an MVS command, the command must be preceded by the MVS command recognition character (#). Always use the pound sign for MVS commands. Maximum number of characters for commands submitted to MVS is 126.

You can also enter an EXEC name, which is the name of the EXEC used to perform a series of commands for the shut down of this task. The EXEC must be in a library included in the SYSPROC concatenation of the MainView AutoOPERATOR subsystem address space. |

Event Specifications 1 panel

The following shows an example of the Event Specifications 1 panel.

Figure 82: Event Specifications 1 panel example
The following table describes the fields for the Event Specifications 1 panel.

**Table 30: Event Specifications 1 panel fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generate</strong></td>
<td>Specify whether CSM automatically generates Start and Stop Rules for CSM objects.  &lt;br&gt;<strong>Yes:</strong> CSM automatically generates Start and Stop Rules for the Text IDs specified on the Event Specifications 2 panel  &lt;br&gt;<strong>No:</strong> CSM does not automatically generate Start and Stop Rules for the Text IDs specified on the Event Specifications 2 panel.  &lt;br&gt;The Start and Stop Rules (and optionally, the Rules for an abnormal termination event) must be user-defined. These Rules must be stored in a different BBPARM member than AAORULCM. Only the CSM-generated Rules are stored in BBPARM member AAORULCM.  &lt;br&gt;The reverse is also true: specifying YES on the Event Specifications 2 panel cause Rules to be created for the Text IDs specified on the Event Specifications 1 panel. Specifying NO on the Event Specifications 2 panel means no Rules are generated for the Text IDs specified on the Event Specifications 1 panel.</td>
</tr>
<tr>
<td><strong>Start Events: Text ID</strong></td>
<td>Specify the ID of the message that indicates the object has been started.  &lt;br&gt;By default, this field contains the value specified on the default control parameters panel. The text ID is used to dynamically build a Rule in the Rule Set allocated for CSM.  &lt;br&gt;When this message is issued by an object, the Rules Processor schedules EXEC CSMUP to inform CSM of the object’s state.  &lt;br&gt;<strong>Format:</strong> 1 to 16 characters  &lt;br&gt;<strong>Example:</strong> $HASP373 or IEF403I</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Specify the event type of the event that signals CSM that an object has initialized.  &lt;br&gt;A Rule is generated using the Start Text-ID and this event type.  &lt;br&gt;Must be one of the following event types: MSG, CMD, ALRT, JRNL  &lt;br&gt;For MSG events, if the Job Type / Name fields are left blank, the resulting Rule is generated with a Job Type of STC and a Job name equal to the object name.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Job Type   | Specify the job type of the text ID that signals to CSM that an object has either started or stopped.  
The Job Type field, along with the event type, Job Name and Text-ID, is used to construct a Rule for this object.  
Must be one of the following job types: STC, JOB, TSO                                                                                                       |
| Job Name   | Specify the job name of the event that signals CSM that an object has abnormally terminated.  
The Job Name field, along with the other fields on the Event Specifications #2 panel are used to construct event Rules for this object.  
Must be a valid job name. Rules Processor wildcard characters are permitted.                                                                                     |
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Stop Events Text-ID** | Specify the ID of the message that indicates the object has been stopped. By default, this field contains the value specified on the default control parameters panel. The text ID is used to dynamically build a Rule in the Rule Set allocated for CSM. When this message is issued by an object, the Rules Processor schedules EXEC CSMDOWN to inform CSM of the object’s state. **Format:** 1 to 16 characters **Example:** $HASP395 or IEF404I Normally, CSM monitors object status changes using the object’s Start or Stop text IDs. However, there are situations where monitoring these messages is not sufficient because an object might terminate without producing the expected messages. For example, the expected stop messages might not be produced if the OS/390 FORCE or SYSPROG EXEC commands are used to terminate an object. The End-of-Memory EXEC CSMEOM enables CSM to monitor an object stopping regardless of how the object stops. Because MainView AutoOPERATOR allows only one End-of-Memory EXEC in each MainView AutoOPERATOR subsystem, the CSM End-of-Memory EXEC is designed to be called from the EXEC, IMFEOM. Two parameters are passed to the CSM End-of-Memory EXEC:  
  - The address space ID of the terminated address space  
  - A character string which can have one of two values: NORMAL: Indicates normal address space termination ABNORMAL: Indicates address space was terminated by passing it to RTM This might happen when using the SYSPROG EXIT command or the OS/390 FORCE command. This is not an indication that the address space abended with a system or user abend code. If the installation already uses an IMFEOM EXEC, you can add a call to the CSMEOM EXEC by adding the following to the IMFEOM EXEC: **IMFEXEC SELECT(CSMEOM IMFOASID EOMSTAT)** If you have not used the IMFEOM EXEC before, use BBSAMP sample member SAMPEOM to implement the End-of-Memory EXEC. To use this EXEC, copy SAMPEOM to prefix.UBBPROC, and make sure to rename it to IMFEOM during the copy process. |

### Event Specifications 2 panel

The following shows an example of the Event Specifications 2 panel.

**Figure 83: Event Specifications 2 panel example**
The following table describes the fields for the Event Specifications 2 panel.

### Table 31: Event Specifications 2 panel fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abnormal Termination events</strong></td>
<td>Specify the text ID of the message that indicates the object has terminated abnormally (abended); this is also known as an abnormal termination event. If you specify an abnormal termination text ID, when an abnormal termination event occurs, the Rules Processor will schedule the CSMABEND EXEC, which notifies CSM that the object has abended. If a recovery command or EXEC is defined for the object (refer to the description of Recovery Command/EXEC), the command or EXEC is issued and CSM also issues the ALERT ACM766A to notify you that the abnormal termination event has occurred. <strong>Format:</strong> 1 to 16 characters <strong>Example:</strong> IEF450I</td>
</tr>
<tr>
<td><strong>Text ID</strong></td>
<td>Specify the job name of the event that signals CSM that an object has abnormally terminated. The Job Name field, along with the other fields on the Event Specifications #2 panel are used to construct event Rules for this object. Must be a valid job name. Rules Processor wildcard characters are permitted.</td>
</tr>
<tr>
<td><strong>Job Name</strong></td>
<td>Specify the job type of the text ID that signals to CSM that an object has either started or stopped. The Job Type field, along with the event type, Job Name and Text ID, is used to construct a Rule for this object. Must be one of the following job types: STC, JOB, TSO</td>
</tr>
<tr>
<td><strong>Job Type</strong></td>
<td>Specify optional text. You might want CSM to handle object abnormal termination processing for particular abend codes displayed in the message identified by the Abend Text-ID field. To do this, you can use the optional Text String field. The 1- to 16 character value specified in this field is compared (using the OP field) with the actual value of the WORDx variable generated from the text ID. Abnormal termination handling does not take place unless the compare is successful.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Word</strong></td>
<td>Specify the 2-digit number of the message text. The Abend Text-ID field identifies the event which signifies abnormal termination for an object. The Rules Processor breaks the message down into a series of blank delimited values. Each value is stored in a WORDnn variable. The Abend String Position field identifies which WORDnn variable (i.e. WORD7, WORD12, WORD2, etc.) contains the value compare against the value in the Abend Search String to determine if abnormal termination processing is to take place. Valid values are numeric, up to 4 positions.</td>
</tr>
<tr>
<td><strong>Op</strong></td>
<td>Specify one of the following operands: EQ (equal), LT (less than), GT (greater than), NE (not equal). The values specified in the Abend Search String and the Abend String Position fields identifies the two values to be compared to determine if abnormal termination processing is to take place for an object. The operand specified in the Op field indicates what kind of comparison occurs between the two values.</td>
</tr>
<tr>
<td><strong>Event Type</strong></td>
<td>Specify the event type of the event that signals CSM that an object has initialized. A Rule is generated by using the Abend Text-ID and this event type. Must be one of the following event types: MSG, CMD, ALRT, JRNL</td>
</tr>
</tbody>
</table>

**User Notification panel**

The following shows an example of the User Notification panel.

**Figure 84: User Notification panel example**

```
BMC Software --------- User Notification for AAOD71X - Edit --------- AutoOPERATOR
COMMAND ===> TGT --- D71A
TSO SEND ===> Users
Pager Notification (ELAN) not supported.
Object Alert Queue ===> MAIN Alert Publish Mode ===> ADD
Enter END command to process and return to Detail Control, ENTER to continue
```

The following table describes the fields for the User Notification panel.
Table 32: User Notification panel fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSO SendUser ID</td>
<td>CSM automatically sends a TSO message if an exceptional condition for the managed object exists. The TSO Send User ID parameter should contain the user ID of the responsible party for this object. The message sent to this TSO user ID is the same as the ALERT message generated for the exception condition. If the designated TSO user is not logged on to the system when the message is sent, the message will be dropped (The LOGON attribute is not used).</td>
</tr>
<tr>
<td>Format:  userid01</td>
<td>Range: Any valid 1- to 7-character TSO user ID</td>
</tr>
<tr>
<td>Object Alert Queue</td>
<td>Specify the name of the ALERT queue in the ALERT Management Facility that will receive ALERTs for this object from CSM.</td>
</tr>
<tr>
<td>Format:  1 to 8 alphanumeric characters</td>
<td>Example: MAIN or CSMOS/390</td>
</tr>
<tr>
<td>Alert Publish Mode</td>
<td>Specify whether the ALERT is published</td>
</tr>
</tbody>
</table>

Object Groups panel

The following shows an example of the Object Groups panel.

Figure 85: Object Groups panel example

BMC Software --------- Object Groups object --------- MainView AutoOPERATOR
COMMAND ===> SCROLL ===> CSR
Primary Commands: Locate CMDSHOW CANCEL
LC CMDs ------- (R)emove (L)ist (M)ember M(o)veable
Enter END command to process and return to Detail Control, ENTER to continue
LC Group Status Objects Description
_ NS61 Member 6 Converted on: 01/30/2001 15:45
_ NS62             0
****************************************************************************** Bottom of data ******************************************************************************

The following table describes the fields for the Object Groups panel.

Table 33: Object Groups panel fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>shows the name identifying a group of objects managed by CSM This name must be unique among the other members of the CSM-Plex to ensure proper object status reporting on the Global Overview display.</td>
</tr>
<tr>
<td>Status</td>
<td>shows the status of the object</td>
</tr>
</tbody>
</table>
Field | Description
---|---
Objects | shows the number of objects within the group
Description | contains a description about the group

The following table describes the primary commands for the CSM Object Groups panel.

**Table 34: Primary commands: Object Groups panel**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locate</td>
<td>enables you to locate a specific CSM Group in the Object Groups display. This command works similar to Locate command in ISPF display.</td>
</tr>
<tr>
<td>CMDSHOW</td>
<td>enables you to control the display of the Primary and Line commands. Issuing the command causes the application to be displayed without the Primary and Line commands. Issuing CMDSHOW again reverses the change and re-displays the Primary and Line commands.</td>
</tr>
<tr>
<td>CANCEL</td>
<td>enables you to cancel your changes</td>
</tr>
</tbody>
</table>

The following table describes the primary commands for the CSM Object Groups panel.

**Table 35: Line commands: Object Groups panel**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove</td>
<td>removes the association of this object with the selected group. Each CSM object must be a member of at least one CSM group. Therefore, you cannot remove an object from a group if it belongs only to that group.</td>
</tr>
<tr>
<td>List</td>
<td>displays all objects, associated with the selected group</td>
</tr>
<tr>
<td>Member</td>
<td>makes the object a member of the selected group. When object is a member of a group, CSM will start the object according to the schedule on the BBI-SS PAS where the group is active.</td>
</tr>
<tr>
<td>Moveable</td>
<td>adds the object as a moveable to the selected group. When object is Moveable to a group CSM will not start the object when this group is active until CSM Move is issued to move the object to that group.</td>
</tr>
</tbody>
</table>

**Object Requisites panel**

The following shows an example of the Object Requisites panel.
The following table describes the fields for the Object Requisites panel.

**Table 36: Object Requisites panel fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Parent Object | shows the name of the parent for an object  
If that object is a member of more than one group, each group will be listed on this panel. |
| Group       | shows the name identifying a group of objects managed by Continuous State Manager  
This name must be unique among the other members of the CSM-Plex to ensure proper object status reporting on the Global Overview display. |
| Description | shows a description about the object                                          |

The following table describes the primary commands for the CSM Object Requisites panel.

**Table 37: Primary commands: Object Requisites panel**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| ADD       | Enter the ADD command to add a new parent for an object; the Add Object Requisites panel is displayed.  
On this panel, enter the object name of the parent and the group to which the object belongs. |
| END       | Enter END to return to the Object Detail Control panel.                                                                                                                                                     |
| CMDSHOW   | Enter this command to control the display of the Primary and Line commands.  
Issuing the command causes the application to be displayed without the Primary and Line commands.  
Issuing CMDSHOW again reverses the change and re-displays the Primary and Line commands. |

The following table describes the primary commands for the CSM Object Requisites panel.
Table 38: Line commands: Object Requisites panel

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| Edit | displays the Edit Object Requisites panel  
On this panel, enter the object name of the parent and the group the object belongs to. |
| Delete | removes a server object from the required objects list for a selected object |

Client Server Relationships panel

The following shows an example of the Client/Server Relationships panel.

**Figure 87: Client/Server Relationships panel example**

BMC Software --- Client/Server Relationships for object -- MainView AutoOPERATOR  
COMMAND ===> SCROLL ==> CSR  
TGT --- NS61  
Primary Commands: ADD END CMDSHOW  
LC CMDS ------- (E)dit server, (D)elete server  
Enter END command to process and return to Detail Control, ENTER to continue  
LC Server Group Description  
Connect Command  
Disconnect Command  
******************************* Bottom of data ********************************

The following table describes the fields for the Client/Server Relationships panel.

Table 39: Client/Server relationships panel fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Server | shows the name of the server for an object  
If that object is a member of more than one group, each group will be listed on this panel.  
This is the same name that is started from the MVS console or submitted as a batch job. The server name must be an object under the control of CSM. This field is modifiable. |
| Group | is the name identifying a group of objects managed by CSM  
This name must be unique among the other members of the CSM-Plex to ensure proper object status reporting on the Global Overview display. |
| Description | shows a description about the object |
### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Connect Command** | shows the command or EXEC routine to connect the client object to the server object  
This is usually, though not restricted to, an MVS Modify command or a VTAM Vary command.  
Prefix this command with the appropriate command recognition character, or CSM reads the input as an EXEC. Use a pound sign (#) for MVS commands.  
Sample Modify command:  
#F IMSPROD1,START  
Sample VTAM Vary command:  
#V NET,ACT,ID=IMSPROD1  
If an EXEC name is used, it must reside in the library concatenation of the SYSPROC DD name of the MainView AutoOPERATOR subsystem address space. |
| **Disconnect**      | shows the command or EXEC to disconnect the client object from the server object before the server object is ended  
This is usually, though not restricted to, an MVS Modify command or a VTAM Vary command.  
Prefix the Disconnect command with the appropriate command recognition character, or CSM reads the input as an EXEC. Use a pound sign (#) for MVS commands.  
Sample Modify command:  
#F IMSPROD1,STOP  
Sample VTAM Vary command:  
#V NET,INACT,ID=IMSPROD1  
If an EXEC name is used, it must reside in the library concatenation of the SYSPROC DD name of the MainView AutoOPERATOR subsystem address space. |

The following table describes the primary commands for the CSM Client/Server Relationships panel.

**Table 40: Primary commands: Client/Server Relationships panel**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| ADD     | Enter ADD to add a new server for an object. Displays the Add Client/Server Relationships where you can add the  
■ Server name of the object  
■ Group Name of the object  
■ Connect command for the object  
■ Disconnect command for the object |
| END     | Enter END to return to the Object Detail Control panel |
Command | Description
--- | ---
CMDSHOW | Use this command to control the display of the Primary and Line commands. Issuing the command causes the application to be displayed without the Primary and Line commands. Issuing CMDSHOW again reverses the change and re-displays the Primary and Line commands.

The following table describes the primary commands for the CSM Client/Server Relationships panel.

**Table 41: Line commands: Client/Server Relationships panel**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>displays the Edit Client/Server Relationships panel. On this panel, edit the</td>
</tr>
<tr>
<td>Delete</td>
<td>removes a server object from the required objects list for a selected object</td>
</tr>
<tr>
<td>Server name of the object</td>
<td></td>
</tr>
<tr>
<td>Group Name of the object</td>
<td></td>
</tr>
<tr>
<td>Connect command for the object</td>
<td></td>
</tr>
<tr>
<td>Disconnect command for the object</td>
<td></td>
</tr>
</tbody>
</table>

**Object Scheduling panel**

The following shows an example of the Client/Server Relationships panel.

**Figure 88: Object Scheduling panel example**

| BMC Software ------ Object Scheduling for object ------ MainView AutoOPERATOR |
| COMMAND ===> | SCROLL ===> | CSR |
| TGT ---- NS61 |
| Primary Commands: ADD CANcel END CMDSHOW |
| LC CMD$ -------- (E)dit Time, (D)elete Time |
| Enter END to process/return to Detail Control, ENTER to continue, or CANcel |
| LC Object Day Down From Down To |
| Group HH:MM HH:MM |

The following table describes the fields for the Object Scheduling panel.
Table 42: Object Scheduling panel fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object/Group</td>
<td>shows the name for the group for which this schedule is defined. The object must be associated with this group.</td>
</tr>
<tr>
<td>Day</td>
<td>shows the name of the day when the object is scheduled to be DOWN.</td>
</tr>
<tr>
<td>Down From</td>
<td>shows the time the object is scheduled to come down; CSM uses the 24-hour clock for managing down times. If a down time is to span midnight, two</td>
</tr>
<tr>
<td></td>
<td>down time parameters must be specified. For example, if an IMS Control Region object is scheduled to be DOWN from 8:00 P.M. on Sunday night to 2:00 A.M. on</td>
</tr>
<tr>
<td></td>
<td>Monday morning, the scheduled down time for that object would be specified as:</td>
</tr>
<tr>
<td></td>
<td>SUNDAY 20:00 24:00</td>
</tr>
<tr>
<td></td>
<td>MONDAY 00:00 02:00</td>
</tr>
<tr>
<td></td>
<td><strong>Format</strong>: hh:mm, where hh=hours and mm=minutes</td>
</tr>
<tr>
<td></td>
<td><strong>Range</strong>: 00:00 to 24:00. The 00:00 is the beginning of the day; the 24:00 is the end of the day.</td>
</tr>
<tr>
<td>Down To</td>
<td>shows the time the object starts after a scheduled down time; CSM uses the 24-hour clock for managing down times. <strong>Format</strong>: hh:mm, where hh=</td>
</tr>
<tr>
<td></td>
<td>hours and mm=minutes</td>
</tr>
<tr>
<td></td>
<td><strong>Range</strong>: 00:00 to 24:00. The 00:00 is the beginning of the day; the 24:00 is the end of the day.</td>
</tr>
</tbody>
</table>

The following table describes the primary commands for the CSM Object Scheduling panel.
## Table 43: Primary commands: Object Scheduling panel

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| ADD     | displays the Add Object Scheduling panel where you can add  
  - name of the group for which this schedule is defined; the object must be associated with this group  
  - days of the week the object is supposed to be down  
  To schedule an object to be down at a specific time on a specific day, enter the day the down time is to occur (Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday). For ease of use, days have also been grouped by keywords:  
  - Weekday: includes Monday, Tuesday, Wednesday, Thursday and Friday  
  - Weekend: includes Saturday and Sunday  
  - Daily: all the days of the week  
  All days can be abbreviated to the first two letters except for Weekday (WKD), Weekend (WKN) and Daily (DLY).  
  - Down From and Down To times (in \textit{hh:mm} format) |
| END     | enter **END** to return to the Object Detail Control panel |
| CMDSHOW | enables you to control the display of the Primary and Line commands  
  Issuing the command causes the application to be displayed without the Primary and Line commands.  
  Issuing CMDSHOW again reverses the change and re-displays the Primary and Line commands. |

The following table describes the primary commands for the CSM Object Scheduling panel.
Table 44: Line commands: Object Scheduling panel

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| Edit | displays the Edit Object Scheduling panel On this panel, edit  
■ name for the group for which this schedule is defined. The object must be associated with this group.  
■ days of the week the object is supposed to be down  
To schedule an object to be down at a specific time on a specific day of the week, enter the day the down time is to occur (Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday)  
For ease of use, days have also been grouped by keywords:  
■ Weekday: includes Monday, Tuesday, Wednesday, Thursday and Friday  
■ Weekend: includes Saturday and Sunday  
■ Daily: all the days of the week  
All days can be abbreviated to the first two letters except for Weekday (WKD), Weekend (WKN) and Daily (DLY).  
■ Down From and Down To times (in hh:mm format) |
| Delete | use to delete a scheduled DOWN time |

Global Calendar Override panel

CSM enables the creation of calendar overrides for special situations in which objects can be brought DOWN outside the normal scheduled down time.

Overrides also can be scheduled to extend the time an object is available.

The Global Calendar Override application does not check for conflicting schedules with dependents of the object for which the override is scheduled.

When an override is specified for an object, that object and all of its dependents are taken down on the specified date and time. This eliminates the need to specify the special down times for each object affected by the temporary schedule change.

The Global Calendar Override functions under the control of a specific calendar date. The override is only for the date specified in this panel; therefore, the affected
objects do not have to be restored to their normal schedules after the override takes place.

**Figure 89: Global Calendar Override panel example**

| BMC Software --------- Global Calendar Override --------- | MainView AutoOPERATOR |
| TGT --- NS61 |
| Primary Commands: ADD CANcel END CMDSHOW |
| DATE --- 01/31/20 |
| TIME --- 17:25 |
| LC CMDS --- (I)nsert Time, (D)elete Time |
| Press END to process and return, CANcel to exit without changes. |
| LC Object | Group / SYSNAME | Calendar Date | Down | Reason for Down Time |
| mm/dd/yy | HH:MM | HH:MM | Contact Name |

Press END to process and return, CANcel to exit without changes.

The following table describes the fields for the Global Calendar Override panel.

**Table 45: Global Calendar Override panel**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>object name to which this special schedule override applies</td>
</tr>
<tr>
<td></td>
<td>Because of the parent-child dependencies that CSM manages, CSM automatically</td>
</tr>
<tr>
<td></td>
<td>overrides the schedules of any objects that are defined as dependents of</td>
</tr>
<tr>
<td></td>
<td>the selected object. CSM also automatically disconnects any client objects</td>
</tr>
<tr>
<td></td>
<td>defined to the selected object before ending the object.</td>
</tr>
<tr>
<td></td>
<td>When the object is returned to operation, the dependent objects are</td>
</tr>
<tr>
<td></td>
<td>restarted and the client objects are connected to the server object</td>
</tr>
<tr>
<td></td>
<td>automatically.</td>
</tr>
<tr>
<td></td>
<td><strong>Format:</strong> objname</td>
</tr>
<tr>
<td>Group/SYSNAME</td>
<td>group and OS/390 SYSNAME where this override is supposed to occur</td>
</tr>
<tr>
<td>Calendar Date</td>
<td>specific date for which the special schedule applies</td>
</tr>
<tr>
<td></td>
<td>This is the only date the override is in effect. On all other days and</td>
</tr>
<tr>
<td></td>
<td>dates, the normal schedules for the objects are in effect. If a special</td>
</tr>
<tr>
<td></td>
<td>schedule is specified for an object, it is used in addition to the normal</td>
</tr>
<tr>
<td></td>
<td>schedule for the object.</td>
</tr>
<tr>
<td></td>
<td><strong>Format:</strong> 8 characters mm/dd/yy, where mm=month, dd=day, yy=year</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **Down From** | time CSM begins the process of stopping the selected object and its dependents  
If it is necessary to have the system completely down by a specific time, you must allow time for the normal shutdown process to occur. If a down time is to span midnight of one day into another day, two down time parameters must be specified.  
For example, if an IMS Control Region object is scheduled to be DOWN from 8:00 P.M. on Sunday night to 2:00 A.M. on Monday morning, the scheduled down time for that object would be specified as:  
SUNDAY 22:00 24:00  
MONDAY 00:00 02:00  
**Format:** hh:mm, where hh=hours and mm=minutes  
**Range:** 00:00 to 24:00  
The 00:00 is the beginning of the day; the 24:00 is the end of the day. |
| **Down To** | time CSM attempts to start the object, except in the case of OS/390  
If OS/390 is not active, then MainView AutoOPERATOR and CSM are not active. However, when MainView AutoOPERATOR and CSM are initialized, CSM automatically starts all the objects that should be active at that time.  
**Format:** hh:mm, where hh=hours and mm=minutes  
**Range:** 00:00 to 24:00  
The 00:00 is the beginning of the day; the 24:00 is the end of the day. |
| **Reason for Down Time** | allows to enter documentation for the reason for the special down time for the affected objects  
**Format:** 1 to 21 positions of free-form text |
| **Contact** | group or individual responsible for scheduling the special down time for the affected objects  
**Format:** 1 to 16 positions of free-form text |

The following table describes the primary commands for the Global Calendar Override panel.

**Table 46: Global Calendar Override primary commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| ADD     | Add scheduled down times for the selected objects.  
Type ADD on the **COMMAND** line of the display for an additional scheduled down time for this object. If the added schedule conflicts with any required objects, a message indicating the condition is displayed.  
ADD takes effect after you issue the BUILD command. |
| CANcel  | Return to the screen without processing changes.  
Type CANcel on the **COMMAND** line of the display to cancel any changes and return to the screen. |
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>Enter END to return to the Object Detail Control panel.</td>
</tr>
<tr>
<td>CMDSHOW</td>
<td>Control the display of the Primary and Line commands. Issuing the command causes the application to be displayed without the Primary and Line commands. Issuing CMDSHOW again reverses the change and re-displays the Primary and Line commands.</td>
</tr>
</tbody>
</table>

The following table describes the line commands for the Global Calendar Override panel.

**Table 47: Global Calendar Override line commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert Time</td>
<td>I  Add an additional object and special schedule to the Global Calendar Override panel. To insert a new schedule for an object, type the line command in the line command (LC) column and press Enter. Use the tab key to move the cursor to the name field and fill in the blanks in the display. Upon completing each field, press the tab key to position the cursor in the next field. (I)nsert modifications take effect after you issue the BUILD command.</td>
</tr>
<tr>
<td>Delete Time</td>
<td>D  Remove the special schedule for an object. To use this line command, use the tab key to position the cursor in the line command (LC) column for the desired object and type D. Press Enter to remove the special schedule for the selected object. If you delete a schedule by mistake, issue the CANCEL primary command. (D)elete modifications take effect after you issue the BUILD command.</td>
</tr>
</tbody>
</table>

**Object Groups panel 2**

Every object must belong to a group.

Therefore, when you bring CSM up for the first time, the first task you must complete is to define a group before you can begin defining objects. A group name can be one- to eight-characters long, but it is recommended that you name the first group the SSID of the subsystem that you bring CSM up on.

The following shows an example of the Object Groups panel.

**Figure 90: Object Groups panel example**
The following table describes the fields for the Object Groups panel.

**Table 48: Object Groups panel fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>is the name identifying a group of objects managed by Continuous State Manager. This name must be unique among the other members of the CSM-Plex to ensure proper object status reporting on the Global Overview display.</td>
</tr>
<tr>
<td>System</td>
<td>is the OS/390 SYSNAME where the group is enabled. If a group is disabled, this field will be blank.</td>
</tr>
<tr>
<td>SSID</td>
<td>is the subsystem ID (SSID) where the group is enabled. If a group is disabled, this field will be blank.</td>
</tr>
<tr>
<td>Objects</td>
<td>lists the objects in the group.</td>
</tr>
<tr>
<td>Description</td>
<td>contains a description about the group</td>
</tr>
</tbody>
</table>

The following table describes the primary commands for the CSM Object Groups panel.

**Table 49: Primary commands: Object Groups panel**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>enables you to add a new Group to the current CSM repository.</td>
</tr>
<tr>
<td></td>
<td>CSM displays the Group Detail Control panel where you can specify all Group definitions.</td>
</tr>
<tr>
<td></td>
<td>The ADD Primary command can be entered by itself, or with a Group name; for example: <strong>ADD GROUP123</strong>.</td>
</tr>
<tr>
<td></td>
<td>To activate the newly defined Group you need to issue Enable line command from the Object Groups display and specify on which CSM partner (PAS) the Group should be active.</td>
</tr>
<tr>
<td>Locate</td>
<td>enables you to locate a specific CSM Group in the Object Groups display.</td>
</tr>
<tr>
<td></td>
<td>This command works similar to Locate command in ISPF display.</td>
</tr>
<tr>
<td>CMDSHOW</td>
<td>enables you to control the display of the Primary and Line commands.</td>
</tr>
<tr>
<td></td>
<td>Issuing the command causes the application to be displayed without the Primary and Line commands.</td>
</tr>
<tr>
<td></td>
<td>Issuing CMDSHOW again reverses the change and re-displays the Primary and Line commands.</td>
</tr>
</tbody>
</table>
**Command** | **Description**
--- | ---
BUILD | enables you to issue Build against CSM Group
Build saves all changes made online and activates them immediately.
The BUILD command requires parameter of Group Name.

The following table describes the primary commands for the CSM Object Groups panel.

**Table 50: Line commands: Object Groups panel**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>enables you to save all changes made online and activates them immediately against a CSM Group</td>
</tr>
</tbody>
</table>
| Synchronize | tells CSM that there could be a discrepancy between the current state and the physical state of an object
For example, an object could change state without issuing the expected start/stop messages, so its current and physical states would become unsynchronized.
When you issue this command, CSM checks every defined object to see if it is physically present on the system. After checking for the presence of an object, CSM updates the object’s current state variable.
This command does not change the status of objects that are in a manual state. |
| Offload | offloads captured event statistics information to the statistics data set named in the group’s definition |
| Edit | displays the Group Detail Control panel where you can edit the attributes of the group |
| Enable | enables a group |
| Disable | disables a group |
| Add | add new groups to be controlled by CSM |
| Delete | eliminates the group from the control of CSM |
| List | display all objects, associated with the selected group |
| Repeat | enables you to duplicate an existing group’s definition to define a new group with the same information
Use this when you want to define groups with similar attributes. The only limitation is that no two groups can have the same name. |
| Browse | enables you to browse the control parameters for the selected group
The control parameters cannot be modified with this line command. The (B)rowse command displays the Object Groups Detail Control panel. |
Group Detail Control

The following shows an example of the Group Detail Control panel.

**Figure 91: Group Detail Control example**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group name</strong></td>
<td>is the name identifying a group of objects managed by Continuous State Manager</td>
</tr>
<tr>
<td></td>
<td>This name must be unique among the other members of the CSM-Plex to ensure</td>
</tr>
<tr>
<td></td>
<td>proper object status reporting on the Global Overview display.</td>
</tr>
<tr>
<td><strong>Command Prefix</strong></td>
<td>is the 1- to 16-character identifier used by the CSM Command Line Interface to route commands to the appropriate CSM object group for processing</td>
</tr>
<tr>
<td></td>
<td>The prefix must be unique among other members of the local CSM repository.</td>
</tr>
<tr>
<td></td>
<td>Commands issued to the interface can enable users to monitor and manage objects in the group without the need for a terminal session display, knowledge of specialized EXECs and parameters or displaying groups of MainView AutoOPERATOR variables.</td>
</tr>
<tr>
<td><strong>Rule Dataset</strong></td>
<td>is the Rules that indicate the Start, Stop and Recovery events for the objects managed by this group are stored in a parameter data set that is allocated to the MainView AutoOPERATOR BBI-PAS</td>
</tr>
<tr>
<td></td>
<td>In order for the CSM application to know where to write the Rules to be generated when either the BUILD or the SAVERULE command is issued for this group you must specify the Ruleset data set name in this field.</td>
</tr>
<tr>
<td></td>
<td>For proper operation, the data set specified in this entry must be in the BBIPARM DD concatenation of the MainView AutoOPERATOR BBI-PAS.</td>
</tr>
<tr>
<td><strong>Ruleset ID</strong></td>
<td>is the 2-character suffix of the Ruleset ID containing the Start, Stop and Recovery events for the objects managed by this group</td>
</tr>
<tr>
<td></td>
<td>The subsystem component of CSM will append this suffix onto the string AAORUL and checks to see if this Ruleset is active when CSM is managing this group.</td>
</tr>
<tr>
<td></td>
<td>The Ruleset identified by this suffix must be present in one of the data sets in the BBIPARM concatenation in MainView AutoOPERATOR BBI-PAS.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Rule Prefix</td>
<td>is the 4-character string makes up the first four characters of the Rules generated by CSM for this group (see Ruleset ID tutorial)</td>
</tr>
<tr>
<td></td>
<td>As each Rule is generated by the CSM application the prefix will have the Rule Suffix appended to it and written into the <strong>Rule ID</strong> field.</td>
</tr>
<tr>
<td>Suffix</td>
<td>is the 4-digit number that makes up the last four characters of the Rules generated by CSM for this group</td>
</tr>
<tr>
<td></td>
<td>As each Rule is generated by the CSM application the suffix is incremented by one, appended to the Rule Prefix and written into the <strong>Rule ID</strong> field.</td>
</tr>
<tr>
<td>Dataset Name</td>
<td>is the name of a preallocated data set must be supplied in the <strong>Statistics Dataset Name</strong> field when you specify that CSM will capture statistics on object Start, Stop and Recovery events.</td>
</tr>
<tr>
<td></td>
<td>When CSM captures event statistics information the data is written to the CSMSTATISTICS array in the local subsystem. EXEC ACMM800 off loads the data from the array daily at midnight to the data set named in the <strong>Statistics Dataset Name</strong> field.</td>
</tr>
<tr>
<td></td>
<td>Note: The CSM object statistics data set should be a fixed block sequential data set with the maximum record length of 80 bytes.</td>
</tr>
<tr>
<td>Disposition</td>
<td>is used in conjunction with the <strong>Dataset Name</strong> field</td>
</tr>
<tr>
<td></td>
<td>Determines if the data is to be offloaded to the beginning of the data set (SHR) or after the data previously off-loaded (MOD) to the data set.</td>
</tr>
<tr>
<td>Author</td>
<td>is a display-only field showing the TSO ID of the person who last modified the group</td>
</tr>
<tr>
<td>Description</td>
<td>is a brief description (30 characters or less) of the group</td>
</tr>
</tbody>
</table>
Sample EXECs

This appendix describes the JESDOWN sample EXEC that is distributed with MainView AutoOPERATOR solutions.

JESDOWN sample EXEC

The JESDOWN EXEC is located in BBPROC.

It is invoked by the operator to shutdown JES2. When invoked, a WTOR message is displayed. The operator must respond with Y to continue with the shutdown or N to cancel processing. If Y is entered, the following steps are taken:

1. A $PI command is issued to drain all of the initiators.
2. A $P command is issued to stop all new JES requests and reset current activities.
3. Commands are issued to display and then to stop all remotes.
4. Commands are issued to display and then to stop all lines.
5. Commands are issued to display and then to stop all active batch jobs.
6. Commands are issued to display and then to stop all active printers.
7. A IMFEXEC WAIT command is issued.
8. A WRITELOG command is issued.
9. A $PJES2 command is issued next.

10. If the operator gets a response that the system is not dormant, a WTOR message is displayed giving the operator the option to force a shutdown.

11. If the operator replies with a Y to force the shutdown, a $PJES2, ABEND command is issued.
12 If JES2 ends successfully, a WTO message "JES2 shutdown completed successfully" is displayed. If JES2 does not end, a WTO message "JES2 is still draining, please initiate manual procedures" is displayed.

The JESWAIT PROFILE variable can be customized. It is located in the MSU002C EXEC.
Sample REXX EXECs

The BMC sample REXX EXECs use some of the features of the REXX Language.

For a complete discussion of the constructs and the syntax of the REXX language, refer to the IBM manuals for the version of REXX installed in your environment.

The following sample EXECs are the basis for the comparison and discussion of the REXX procedural language. Although further customization and changes are required to fully utilize these EXECs in an production environment, they are intended to illustrate some of the features of the REXX language.

These sample EXECs for REXX are distributed in the BBPROC data set.

REXX compound variable initialization EXECs

Two sample EXECs, MSU006X and MSU006X2, are distributed to show initialization of REXX compound variables.

In CLISTs, compound variables are defined with ampersands (&) and the REXX equivalent of ampersands is the STEM variable. The REXX compound variables are defined with STEM where the STEM variable is of the form VAR.n and n is an integer.

The following instructions reference values for SYS1 and SYS2 within a CLIST:

```
IMFEXEC VGET (SYSN) PROFILE
SET &N = 1 /* INITIALIZE THE INDEX VALUE */
DO WHILE &N LE &SYSN /* SET UP THE LOOP */
  IMFEXEC VGET (SYS&N) PROFILE
  SET &SS = &&SYS&N /* DERIVE THE VALUE */
```

The following set of instructions refer to the same values in a REXX EXEC:

```
ADDRESS IMFEXEC /* SET IMFEXEC AS THE EXTERNAL ENVIRON */
IMFEXEC VGET (SYSN) PROFILE
DO N=1 TO SYSN BY 1 UNTIL N>SYSN
  IMFEXEC VGET ('SYS'*N) PROFILE
  SS = VALUE('SYS'*N) /* DERIVE THE VALUE */
```
Notice that with CLISTs, the value of a compound variable must be derived using the `&&` expression, whereas REXX uses the built-in function `VALUE`.

The apostrophes around `SYS` define it as a literal to REXX and distinguish the literal from the variable `N`. If a STEM variable was defined at initialization, this example could have used the following instruction to get the values for `SYS.N`:

```clist
IMFEXEC VGET (SYS.N) PROFILE
```

With this instruction, there are no apostrophes to distinguish `SYS` as a literal because `'SYS.' is a STEM and REXX defaults it to a literal.

Note that there is no `SET` command in REXX EXECs. Any expression with an equal sign (=) implies an ASSIGNMENT command that functions the same as a CLIST `SET` command. MSU006X initializes the `SYSN` variables for reference by both CLIST and REXX EXECs. MSU006X is a REXX EXEC because the STEM variable, `SYS.1`, is not a valid expression within a CLIST.

MSU006X illustrates the assignment of values to both `SYS1` and `SYS.1` variables but you must code the values within the EXEC. MSU006X2 illustrates the assignment of values to the STEM variables `SYS.n` by deriving the values of `SYS&n` within the context of MainView AutoOPERATOR initialization.

**Figure 92: REXX EXEC - MSU006X**

```clist
/*-----------------------------*/
SYS1="NONE"    /* TARGET SYSTEM(NAME#1) FOR CLIST REFS */
SYS2="NONE"    /* TARGET SYSTEM(NAME#2) FOR CLIST REFS */
SYS.1="NONE"   /* TARGET SYSTEM(NAME#1) FOR REXX REFS */
SYS.2="NONE"   /* TARGET SYSTEM(NAME#2) FOR REXX REFS */
/*-----------------------------*/
```

At MainView AutoOPERATOR initialization, the values for `SYS&n` were specified to match the number of target systems within your environment. To build the REXX STEM variables, you can either change the whole start up EXEC, MSU002C or use MSU006X2, which dynamically assigns the values.

**EXECs for REXX compound variable initialization**

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSU006X</td>
<td>assigns values for SYS1, SYS2, SYS.1 and SYS.2 statically</td>
</tr>
<tr>
<td>MSU006X2</td>
<td>assigns values for SYS.1 and SYS.2 dynamically by deriving the values for SYS1 and SYS2</td>
</tr>
</tbody>
</table>
REXX DASD reserve analysis EXEC

The sample REXX DASD Reserve Analysis EXEC is modeled after the CLIST version of the same EXEC.

This discussion concentrates on the use of REXX functions and instructions.

For the EXEC to function properly in your environment, you should establish values for the following variables.

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSN</td>
<td>Number of MVS systems in the shared DASD configuration</td>
</tr>
<tr>
<td>SYS1-SYSn</td>
<td>Names (for example, SMFID) of each system in a shared DASD configuration</td>
</tr>
</tbody>
</table>

This EXEC is initiated by a Rule for the following IOS message:

IOS071I ucb, chpid, jobname, start pending.

When REXX EXECs are started based on a TEXT ID, MainView AutoOPERATOR does not parse the message text prior to scheduling the specified EXEC. The REXX EXEC must do the parsing using the REXX instruction, PARSE.

The following instruction parses the IOS071I message in the distributed EXEC, IOS071IX.

PARSE ARG IOS071IX P1 ',', P2 ',', P3 ',', P4 P5 .

The PARSE instruction has other keyword operands besides ARG.

You should consult the TSO Extensions Version 2 Reference for a full discussion of the PARSE instruction. In this EXEC, the ARG keyword indicates that parsing should be done on the passed parameter strings.

The parameter string contains commas and blanks; in REXX EXECs, the EXEC must specify the parsing pattern. This example has commas in the pattern to subdivide the message. The following values would be placed in the variables:

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOS071IX</td>
<td>IOS071I</td>
</tr>
<tr>
<td>P1</td>
<td>ucb</td>
</tr>
<tr>
<td>P2</td>
<td>chpid</td>
</tr>
<tr>
<td>P3</td>
<td>jobname</td>
</tr>
<tr>
<td>P4</td>
<td>start</td>
</tr>
</tbody>
</table>
Because this discussion does not go beyond the P5 variable, the EXEC has a period (.) as the last variable.

IOS071IX uses the ADDRESS command to inform REXX that non-REXX language instructions should be passed on to the external environment. After a VGET is completed to derive the value for SYSN (the number of shared systems), IOS071IX goes into the following loop:

```plaintext
DO N=1 TO SYSN BY 1 UNTIL N>SYSN      /* LOOK AT ALL SYSTEMS */
  IMFEXEC VGET ('SYS'N) PROFILE      /* START GETTING SS IDS*/
  TARG = VALUE('SYS'N)               /* SET TARGET SYS NAME */
  IF TARG /= IMFORGSS THEN DO        /* DON'T PROCESS THIS */
    ADDRESS IMFEXEC
    IMFEXEC SELECT,                /* COMMAS ARE USED FOR */
    /* CONTINUATION IN REXX*/
    "EXEC(MPE001X" IMFORGSS P1 P3 ") TARGET("TARG")"
  END
END
```

Because REXX does not permit the use of ampersands, this example has apostrophes to distinguish a literal from a variable when the VGET was issued for the SS IDs. Remember that SYSN is the total number of subsystems in a shared DASD complex and 'SYS'N is the subsystem ID. The built-in function VALUE is used to assign TARG; if this value function was not used, the following IF clause would fail every time:

```plaintext
IF 'SYS'N = IMFORGSS THEN DO
```

The next instruction could have been used successfully in the IF construction:

```plaintext
IF IMFORGSS = VALUE('SYS'N) THEN DO
```

Because follow-up EXEC MPE001X needs to be scheduled on the TARGET system, it is easier to use the ASSIGNMENT clause to set TARG. Note that when MPE001X is scheduled, double quotation marks (" ") are used to distinguish the literals from the variables which are then passed to the IMFEXEC environment. If double quotation marks were not used, REXX would have interpreted EXEC(MPE001X) as a built-in function.

MPE001X is the REXX counterpart to MPE001C. The MPE001X REXX EXEC uses the REXX PARSE and ADDRESS commands along with the VALUE built-in function. MPE001X issues a VGET for IMFRC after the call to the RRES utility and then MPE001X VGETs the number of lines passed back from RRES utility. For this discussion, the following command was added:

```plaintext
IF DATATYPE(RRESROL1)1 = 'NUM' THEN DO   /* IS IT NUMERIC?   */
  IMFEXEC MSG "'RRESROL1 NOT NUMERIC, RRESROL1="RRESROL1"
  SIGNAL MPEXIT                       /* EXIT             */
END
```
The REXX built-in function, `DATATYPE`, ensures that the number of lines is numeric. The `DATATYPE` function also can verify other characteristics. The `IMFEXEC MSG` facility requires the text of the message to be enclosed in single quotation marks, but within REXX, double and single quotation marks are interchangeable.

The samples use double quotation marks to tell REXX that the contents are literals so that REXX passes on the single quotation marks to the external environment, IMFEXEC. The `SIGNAL` command of REXX can be used to branch or trap certain conditions. REXX does not have a `GOTO` command so loops or branches must be controlled by `SIGNAL`, `LEAVE`, or `CALL` commands.

If the UCB in the IOS071I message has an outstanding reserve, MPE001X schedules the MPE002X EXEC. MPE002X issues an ALERT back to the system requesting the information about outstanding reserves. MPE002X is functionally equivalent to MPE002C.

BMC Software distributes two REXX samples, MPE002X and MPE002X2. Both use the REXX `PARSE` and `ADDRESS` instructions and the difference between MPE002X and MPE002X2 is how they build text for ALERTs. MPE002X uses single and double quotation marks and MPE002X2 uses string concatenations.

An ALERT KEY is built with the following instruction:

```
ALERTKEY=INSERT(SYSID,UCB)
```

Using the REXX `INSERT` function, a unique ALERT KEY is built. This instruction inserts the value of SYSID before the value of UCB, which is placed subsequently in a variable called ALERTKEY.

MPE002X issues the ALERT with the following instruction:

```
*IMFEXEC ALERT* ALERTKEY, 
'.MPE002X JOB "JOBNAME" ON SYSTEM "SYSID" USING 
UCB" HELD BY SYSTEM "SS"." "COLOR(WHITE)"
```

Commas indicate continuation in REXX.

There are seven pairs of double quotation marks and one pair of single quotation marks in the above instruction. The same ALERT is issued with the following set of instructions in MPE002X2.

```
WORK1=""                        /* DEFINE LEADING/TRAILING QUOTE */
WORK2='.MPE002X2 JOB '           /* FIRST PART OF THE MESSAGE */
/* CONCATENATE WORK1, WORK2 AND MORE OF THE TEXT */
WORK3=WORK1||WORK2||JOBNAME||' ON SYSTEM "SYSID" USING 'UCB
/* BUILD REST OF THE TEXT WITH THE DELIMITING QUOTE AT END */
WORK4='.', HELD BY SYSTEM '"||SS'||WORK1
TEXT=INSERT(WORK3,WORK4)         /* INSERT WORK3 INFRONT OF WORK4 */
*IMFEXEC ALERT* ALERTKEY TEXT "COLOR(WHITE)"
```

Appendix D  Sample REXX EXECs 351
MPE002X and MPE002X2 are designed to demonstrate the use of quotation marks and string concatenations in REXX. The choice is really stylistic; both results are the same.

EXECs for REXX DASD reserve analysis

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOS071IX</td>
<td>schedules remote EXECs threshold counters</td>
</tr>
<tr>
<td>MPE001X</td>
<td>examines currently held reserves (runs on remote systems)</td>
</tr>
<tr>
<td>MPE002X</td>
<td>issues ALERT back to requesting system</td>
</tr>
<tr>
<td>MPE002X2</td>
<td>issues ALERT back to requesting system</td>
</tr>
</tbody>
</table>

REXX QUEUE PULL and VPUT VGET usage

Two sample EXECs, MUT002X and MUT003X, are distributed to show the use of REXX QUEUE/PULL and IMFEXEC’s VPUT/VGET instructions.

As with all REXX EXECs, the PARSE and ADDRESS commands are used. After these commands, MUT002X calls MUT003X to issue the MVS commands, \texttt{D R,L}. When entered, MUT003X establishes the IMFEXEC as the external environment with the ADDRESS instruction.

The following command displays the outstanding replies:

\texttt{"IMFEXEC CMD 'D R,L' RESPONSE(IEE112I) WAIT(20)"}

Upon return, the response lines are in the local pool. The variable IMFNOL contains the number of lines returned. The variable NUMLINES is assigned the values of IMFNOL and VPUT to the local pool.

MUT003X places information into the MainView AutoOPERATOR local pool and the REXX EXTENDED DATA QUEUE with the following set of instructions:

\begin{verbatim}
   DO N=1 TO IMFNOL BY 1 UNTIL N>IMFNOL
     "IMFEXEC VDCL REXXL"N" LIST(TEXT B C D E F)"
     "IMFEXEC VGET LINE"N" INTO(REXXL"N") LOCAL"
     "IMFEXEC VPUT LINE."N" FROM(REXXL"N") LOCAL"
   QUEUE 'LINE.'N TEXT B C D E F
END
\end{verbatim}

Although \texttt{LINE.'N} looks like a REXX STEM, a specific queued line cannot be referenced by the PULL command; therefore the pseudo stem, \texttt{LINE.'N}, becomes a part of the data that queued.
The `QUEUE` command builds the extended data queue. After the loop is completed, MUT003X issues the `RETURN` command and processing is continued in MUT002X.

Upon return from MUT003X, MUT002X uses the REXX built-in function `QUEUED` to establish the number of lines queued and displays this information before setting up the loop to `PULL` from the extended data queue. The following sequence of instructions is executed:

```plaintext
CNT=QUEUED()
IMFEXEC MSG '.MUT002X: THE NUMBER OF QUEUED LINES IS:' CNT
DO N=1 TO CNT BY 1 UNTIL N>CNT
   PULL LINE TEXT B C D E F
   IMFEXEC MSG '.MUT002X: PULL VARIABLES ARE:' LINE TEXT B C D E F
END
```

Notice that `LINE` is considered a part of the data that is queued and not a REXX `STEM`. REXX discards the data once it has been pulled from the extended data queue. If you want to erase the data queue, build a `PULL` loop without specifying an argument template. The `PULL` instruction also can be specified as `PARSE PULL`.

MUT002X continues execution by getting NUMLINES from the LOCAL POOL and setting up a similar loop to issue `VDCL` and `VGET` instructions to derive the values that were stored in the LOCAL POOL by MUT003X.

**EXECs for REXX QUEUE/PULL and VPUT/VGET usage**

<table>
<thead>
<tr>
<th>EXEC name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUT002X</td>
<td>invokes MUT003X to gather data then VGET the data from the LOCAL POOL and PULL the data from the REX DATA QUEUE</td>
</tr>
<tr>
<td>MUT003X</td>
<td>issues the MVS command <code>D R,L</code> and places the data from the response in the LOCAL POOL and the REXX DATA QUEUE</td>
</tr>
</tbody>
</table>
MVS solution variables

To ensure that the MVS solutions function properly in your environment, you must establish values for several variables.

Variables

The following table lists all MVS solution variables, their default values, and the solutions each one affects.

Edit member MSU002C in the BBPROC data set, distributed in MainView AutoOPERATOR, to set values for the appropriate variables. A Rule for message PM0010I is required to schedule MSU002C upon PAS initialization.

Table 52: MVS solution variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSMM1MIG</td>
<td>ML1 migration in process flag for DASD/Space monitoring</td>
<td>n</td>
<td>dump data set monitoring</td>
</tr>
<tr>
<td>DUMPCLER</td>
<td>task to clear dump data sets</td>
<td>none</td>
<td>dump data set monitoring</td>
</tr>
<tr>
<td>DUMPMMF</td>
<td>minimum number of dump data sets to keep clear</td>
<td>1</td>
<td>dump data set monitoring</td>
</tr>
<tr>
<td>DUMPND5</td>
<td>total number of dump data sets</td>
<td>4</td>
<td>dump data set monitoring</td>
</tr>
<tr>
<td>LBGCLAS</td>
<td>class priority list from high to low</td>
<td>MDRA153</td>
<td>load balancing</td>
</tr>
<tr>
<td>LBGCPUH</td>
<td>CPU utilization to decrease workload</td>
<td>90</td>
<td>load balancing</td>
</tr>
<tr>
<td>LBGCPUL</td>
<td>CPU utilization to increase workload</td>
<td>50</td>
<td>load balancing</td>
</tr>
<tr>
<td>LGBIBEG</td>
<td>beginning initiator number to manage</td>
<td>None</td>
<td>load balancing</td>
</tr>
<tr>
<td>LBGIDEC</td>
<td>number of initiators to stop</td>
<td>1</td>
<td>load balancing</td>
</tr>
<tr>
<td>LBGIEIND</td>
<td>ending initiator number to manage</td>
<td>none</td>
<td>load balancing</td>
</tr>
<tr>
<td>LBGIINC</td>
<td>number of initiators to start</td>
<td>2</td>
<td>load balancing</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Default</td>
<td>Solutions</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------</td>
<td>---------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>LBGPAGE</td>
<td>page rate to decrease workload</td>
<td>xxx</td>
<td>load balancing</td>
</tr>
<tr>
<td>LOGREC</td>
<td>task name to clear SYS1.LOGREC</td>
<td>None</td>
<td>logrec data set monitoring</td>
</tr>
<tr>
<td>MAXACSA</td>
<td>maximum extended CSA utilization</td>
<td>80</td>
<td>performance management</td>
</tr>
<tr>
<td>MAXBCSA</td>
<td>maximum CSA utilization below the limit</td>
<td>80</td>
<td>performance management</td>
</tr>
<tr>
<td>MAXCCPU</td>
<td>maximum complex CPU utilization</td>
<td>95</td>
<td>performance management</td>
</tr>
<tr>
<td>MAXDEVU</td>
<td>maximum device utilization</td>
<td>80</td>
<td>performance management</td>
</tr>
<tr>
<td>MAXTCPU</td>
<td>maximum TSO CPU utilization</td>
<td>50</td>
<td>performance management</td>
</tr>
<tr>
<td>SMFALT</td>
<td>suffix of alternate SMF parameters</td>
<td>none</td>
<td>SMF data set monitoring</td>
</tr>
<tr>
<td>SMFCLEAR</td>
<td>task to dump SMF data sets</td>
<td>none</td>
<td>SMF data set monitoring</td>
</tr>
<tr>
<td>SYSJES</td>
<td>version, release, and modification level of JES</td>
<td>SP3.1.3</td>
<td>solutions that depend upon release-specific messages</td>
</tr>
<tr>
<td>SYSMVS</td>
<td>version, release, and modification level of MVS</td>
<td>SP4.2.2</td>
<td>solutions that depend upon release-specific messages</td>
</tr>
<tr>
<td>SYSN</td>
<td>number of MVS systems in shared DASD complex</td>
<td>2</td>
<td>DASD reserve analysis shared DASD control</td>
</tr>
<tr>
<td>SYSPROG</td>
<td>TSO USERID of primary system programmer to receive warning messages</td>
<td>none</td>
<td>WTO buffer monitoring logrec data set monitoring</td>
</tr>
<tr>
<td>SYS1 - SYSN</td>
<td>system IDs of systems in a shared DASD configuration</td>
<td>none</td>
<td>DASD reserve analysis</td>
</tr>
</tbody>
</table>
CICS solution variables

To ensure that the CICS solutions function properly in your environment, you must establish values for several variables.

Edit member CSUINIT in the BBPROC data set, distributed with MainView AutoOPERATOR, to set values for the appropriate variables.

If the Rule PM0010I (located in Rule Set AAORULM1) is enabled at MainView AutoOPERATOR initialization, the CSUINIT EXEC is scheduled automatically and the variables for the CICS solution are set.

If you choose not to enable PM0010I in Rule Set AAORULM1, you can create a different JRNLR-initiated Rule that fires from the PM0010I initialization message or, you can create an ATSTART time-initiated Rule that fires during MainView AutoOPERATOR initialization. Refer to Chapter 7 of the MainView AutoOPERATOR Basic Automation Guide, Volume 1: Using Rules for more information about how ATSTART Rules work.

Variables

The CICS solution variables also can be set by an operator.

To reset the CICS solution variables, enter %CSUINIT.

Table 53: CICS solution variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCALPT</td>
<td>SSID of MainView AutoOPERATOR subsystem to receive all ALERTs</td>
<td>none</td>
<td>■ error recovery</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ performance management</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Default</td>
<td>Solutions</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------</td>
<td>---------</td>
<td>------------------------------------------------</td>
</tr>
</tbody>
</table>
| ONLCNT01 - ONLCNT0n | name of the controlling subsystem                     | none    | ■ error recovery  
■ file degradation analysis  
■ VSAM control split monitor  
■ transaction response time monitor |
| ONLALT01 - ONLALT0n | color of ALERTs messages for target system n           | RED     | ■ error recovery  
■ file degradation analysis  
■ VSAM control split monitor  
■ transaction response time monitor |
| ONLTYP01 - ONLTYP0n | type of target system n or BBI-SSn                    | None    | ■ error recovery  
■ file degradation analysis  
■ VSAM control split monitor  
■ transaction response time monitor |
| ONLALM01 - ONLALM0n | alarm indicator for ALERT in target system n           | Y       | ■ error recovery  
■ file degradation analysis  
■ VSAM control split monitor  
■ transaction response time monitor |
| ONLSSN     | number of online subsystems                           | 4       | ■ error recovery  
■ file degradation analysis  
■ VSAM control split monitor  
■ transaction response time monitor |
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONLSS01 - ONLSS0n</td>
<td>names of the target subsystems</td>
<td>none</td>
<td>■ error recovery</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ file degradation analysis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ VSAM control split monitor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ transaction response time monitor</td>
</tr>
</tbody>
</table>
DB2 solution variables and parameters

To ensure that the DB2 solutions function properly in your environment, you must establish values for several variables and determine threshold parameters.

Data sets containing variables

DB2 solution variables are found in the following data sets:

- `hilevel.UBBPROC(DSU001C)`
- `hilevel.UBBPROC(DSU101C)`
- `hilevel.UBBPARM(BBIISP00)`
- `hilevel.UBBPARM(BBISSP00)`

where

<table>
<thead>
<tr>
<th>Member name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSU001C</td>
<td>initializes DB2 variables required for all DB2 solutions</td>
</tr>
<tr>
<td>DSU101C</td>
<td>initializes variables per DB2 subsystem</td>
</tr>
<tr>
<td></td>
<td>You can have as many as nine DB2 subsystems. Variables for these additional</td>
</tr>
<tr>
<td></td>
<td>subsystems are set by copying member DSU101C and renaming the member</td>
</tr>
<tr>
<td></td>
<td>DSU10nC90 where n is the DB2 subsystem (1-9) for which you are setting</td>
</tr>
<tr>
<td></td>
<td>variables.</td>
</tr>
<tr>
<td>BBIISP00</td>
<td>contains a startup list to which you add BLK members DPE002B and DPE030B</td>
</tr>
<tr>
<td></td>
<td>during initial customization</td>
</tr>
<tr>
<td>BBISSP00</td>
<td>contains Rules parameters that you update during initial customization</td>
</tr>
</tbody>
</table>
Data sets containing threshold parameters

DB2 threshold parameters are found in these data sets:

- `hilevel.UBBPARM(DPE002B)`
- `hilevel.UBBPARM(DPE030B)`
- `hilevel.UBBPARM(DMRBEX00)`

where

<table>
<thead>
<tr>
<th>Member name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPE002B</td>
<td>DPE002B and DPE030B contain parameters that specify the names of monitors you add during initial customization of the DB2 solutions</td>
</tr>
<tr>
<td>DPE030B</td>
<td>See description for DPE002B</td>
</tr>
<tr>
<td>DMRBEX00</td>
<td>This member contains parameters for setting threshold levels in the DB2RNWY solution during initial customization of the DB2 solutions</td>
</tr>
</tbody>
</table>

Variables and parameters

Variables and threshold parameters for the DB2 solutions are organized in tables according to the solutions that each one affects.

You must set the following variables listed in the “All DB2 solutions” on page 362 section:

- DB2
- D_DB2NUM

These variables are part of the initial environment setup and you must set both of them before you use any of the DB2 solutions.

The rest of the variables found in the following five tables are optional. Defaults are provided for them.

All DB2 solutions

This section contains the variables required for all DB2 solutions.
Table 54: Variables for all DB2 solutions in hilevel.UBBPROC

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
<th>Default</th>
<th>Data set member</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_DB2NUM</td>
<td>number of DB2s monitored by this subsystem</td>
<td>1</td>
<td>DSU001C</td>
</tr>
<tr>
<td></td>
<td>Note: Setting this variable is mandatory.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D_ALERT_IDS</td>
<td>USERIDs to be notified of DB2 Alerts</td>
<td>Null</td>
<td>DSU001C</td>
</tr>
<tr>
<td>D_ALERT_Q</td>
<td>queue name to use for Alerts</td>
<td>DB2</td>
<td>DSU001C</td>
</tr>
<tr>
<td>D_ALERT_SEND</td>
<td>indicates (Y</td>
<td>N) that SEND command is to be issued</td>
<td>N</td>
</tr>
<tr>
<td>D_ALERT_SEN_DOPT</td>
<td>SEND option: either LOGON or NOW</td>
<td>NOW</td>
<td>DSU001C</td>
</tr>
<tr>
<td>DB2A</td>
<td>name you supply to identify each DB2 subsystem</td>
<td>DB2D</td>
<td>DSU101C</td>
</tr>
<tr>
<td>D_ALERT_TGT</td>
<td>target for MainView Alerts</td>
<td>&amp;IMFORGSS</td>
<td>DSU001C</td>
</tr>
<tr>
<td>MSTR</td>
<td>DB2 MSTR address space name associated with this DB2 (for example, DB2AMSTR, DSNBMSTR, and so on)</td>
<td>&amp;DB2.MSTR</td>
<td>DSU101C</td>
</tr>
<tr>
<td>DBM1</td>
<td>DB2 DBM1 address space name associated with this DB2 (for example, DB2ADBM1, DSNBDBM1, and so on)</td>
<td>&amp;DB2.DBM1</td>
<td>DSU101C</td>
</tr>
<tr>
<td>DIST</td>
<td>DB2 DIST address space name associated with this DB2 if applicable at your site (for example DB2ADIST, DSNBDIST, and so on)</td>
<td>&amp;DB2.DIST</td>
<td>DSU101C</td>
</tr>
<tr>
<td>IRLM</td>
<td>IRLM PROC name associated with this DB2 subsystems (for example DB2PROC, IRLMPROC, and so on)</td>
<td>&amp;DB2.PROC</td>
<td>DSU101C</td>
</tr>
<tr>
<td>IMST</td>
<td>optional IMS target name associated with this DB2 subsystem (null if not applicable)</td>
<td>Null</td>
<td>DSU101C</td>
</tr>
<tr>
<td>SPAS</td>
<td>DB2 SPAS address space name associated with this DB2, if applicable at your site (for example DB2ASPAS, DSNBSPAS, and so on)</td>
<td>&amp;DB2.SPAS</td>
<td>DSU101C</td>
</tr>
</tbody>
</table>

DB2RNWY threshold parameters

This section contains the DB2RNWY threshold parameters.

Table 55: DB2RNWY threshold parameters in hilevel.UBBPARM

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Connection</th>
<th>Threshold parameter</th>
<th>Data set member</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG=DZ0610W</td>
<td>IMS</td>
<td>GPRATE=200 CYCLES=3</td>
<td>DMRBEX00</td>
</tr>
<tr>
<td>Keyword</td>
<td>Connection</td>
<td>Threshold parameter</td>
<td>Data set member</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>----------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>MSG=DZ0620W</td>
<td>CICS</td>
<td>GPRATE=200 CYCLES=3</td>
<td>DMRBEX00</td>
</tr>
<tr>
<td>MSG=DZ0630W</td>
<td>TSO</td>
<td>GPRATE=200 CYCLES=3</td>
<td>DMRBEX00</td>
</tr>
<tr>
<td>MSG=DZ0640W</td>
<td>BATCH</td>
<td>GPRATE=500 CYCLES=3</td>
<td>DMRBEX00</td>
</tr>
<tr>
<td>MSG=DZ0650W</td>
<td>CAF</td>
<td>GPRATE=500 CYCLES=3</td>
<td>DMRBEX00</td>
</tr>
<tr>
<td>MSG=DZ0660W</td>
<td>UTILITY</td>
<td>GPRATE=1000 CYCLES=3</td>
<td>DMRBEX00</td>
</tr>
</tbody>
</table>

**DB2TFUL variables**

This section describes the DB2TFUL variables.

Table 56: DB2TFUL variables in *hilevel.UBBPROC*

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
<th>Default</th>
<th>Data Set member</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_UTIL</td>
<td>default maximum database/tablespace (DB/TS) utilization for all DB/TSs</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td></td>
<td>associated with this subsystem; zero means no default threshold</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This value can be overridden for specific DB/TSs within each DB2 subsystem.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX_EXTS</td>
<td>default maximum number of extents for all DB/TSs associatd with this DB2</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td></td>
<td>subsystem</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This value can be overridden for specific DB/TSs within each DB2 subsystem.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUM</td>
<td>total number of overrides for this DB2 subsystem</td>
<td>4</td>
<td>DSU101C</td>
</tr>
<tr>
<td>NAME n</td>
<td>n th DBTS name to be overridden for percent space utilization and</td>
<td>NA</td>
<td>DSU101C</td>
</tr>
<tr>
<td></td>
<td>extent monitoring for this DB2 subsystem</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generic DBTS names may be used.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME1</td>
<td>first DBTS name</td>
<td>DSN*</td>
<td>DSU101C</td>
</tr>
<tr>
<td>NAME2</td>
<td>second DBTS name</td>
<td>DSNDB06_</td>
<td>DSU101C</td>
</tr>
<tr>
<td></td>
<td>DSNAPH01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME3</td>
<td>third DBTS name</td>
<td>DSNDB07_</td>
<td>DSU101C</td>
</tr>
<tr>
<td></td>
<td>DSN4K01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME4</td>
<td>fourth DBTS name</td>
<td>DSNDB06_</td>
<td>DSU101C</td>
</tr>
<tr>
<td></td>
<td>SYS*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX_UTILn</td>
<td>utilization threshold for all DB/TSs that satisfy the NAMEn qualification</td>
<td>NA</td>
<td>DSU101C</td>
</tr>
<tr>
<td></td>
<td>Zero means ignore percent utilization.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### DB2THRD variables

This section describes the DB2THRD variables.

**Table 57: DB2TFUL variables in hilevel.UBBPROC**

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
<th>Default</th>
<th>Data Set member</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_UTIL1</td>
<td>utilization threshold for NAME1</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_UTIL2</td>
<td>utilization threshold for NAME2</td>
<td>90</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_UTIL3</td>
<td>utilization threshold for NAME3</td>
<td>50</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_UTIL4</td>
<td>utilization threshold for NAME4</td>
<td>50</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_EXTSn</td>
<td>extent threshold for all DB/TS's that satisfy the NAMEn qualification. Zero means ignore percent utilization.</td>
<td>NA</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_EXTS1</td>
<td>extent override for NAME1</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_EXTS2</td>
<td>extent override for NAME2</td>
<td>1</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_EXTS3</td>
<td>extent override for NAME3</td>
<td>20</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_EXTS4</td>
<td>extent override for NAME4</td>
<td>1</td>
<td>DSU101C</td>
</tr>
</tbody>
</table>

### DB2RESP monitor thresholds

This section describes the DB2RESP monitor thresholds.

**Table 58: Monitor thresholds for DB2RESP in hilevel.UBBPARAM**

<table>
<thead>
<tr>
<th>Monitor</th>
<th>Connection</th>
<th>Description</th>
<th>Threshold</th>
<th>Data set member</th>
</tr>
</thead>
<tbody>
<tr>
<td>@ELAP</td>
<td>IMS</td>
<td>average elapsed time</td>
<td>5 seconds</td>
<td>DPE002B</td>
</tr>
<tr>
<td>@ELPD</td>
<td>IMS</td>
<td>average elapsed time in DB2</td>
<td>3 seconds</td>
<td>DPE002B</td>
</tr>
<tr>
<td>@CPU</td>
<td>IMS</td>
<td>average CPU time</td>
<td>0.5 seconds</td>
<td>DPE002B</td>
</tr>
</tbody>
</table>
### DB2RESP variables

This section describes the DB2RESP variables.

**Table 59: DB2RESP variables in *hilevel.UBBPROC***

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
<th>Default</th>
<th>Data set member</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_TSOELAP</td>
<td>maximum elapsed time per TSO user</td>
<td>00:01:00 (1 minute)</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_TSOSTMT</td>
<td>maximum number of SQL statements issued per TSO user</td>
<td>500</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_TSOLOCK</td>
<td>maximum number of LOCKS held per TSO user</td>
<td>500</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_BATELAP</td>
<td>maximum elapsed time per BATCH transaction</td>
<td>01:00:00 (1 hour)</td>
<td>DSU101C</td>
</tr>
<tr>
<td>Variable name</td>
<td>Description</td>
<td>Default</td>
<td>Data set member</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>MAX_BATSTMT</td>
<td>maximum number of SQL statements issued per BATCH transaction</td>
<td>100,000</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_BATLOCK</td>
<td>maximum number of LOCKs held per BATCH transaction</td>
<td>2,000</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_UTLELAP</td>
<td>maximum elapsed time per UTILITY transaction</td>
<td>00:30:00 (30 minutes)</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_UTLSTMT</td>
<td>maximum number of SQL statements issued per UTILITY transaction</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_UTLLOCK</td>
<td>maximum number of LOCKs held per UTILITY transaction</td>
<td>50</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_CAFELAP</td>
<td>maximum elapsed time per CAF transaction</td>
<td>00:30:00 (30 minutes)</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_CAFSTMT</td>
<td>maximum number of SQL statements issued per CAF transaction</td>
<td>100,000</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_CAFLOCK</td>
<td>maximum number of LOCKs held per CAF transaction</td>
<td>2,000</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_TPELAP</td>
<td>maximum elapsed time per CICS or IMS transaction</td>
<td>00:00:05 (5 seconds)</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_TPSTMT</td>
<td>maximum number of SQL statements issued per CICS or IMS transaction</td>
<td>100</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_TPLOCK</td>
<td>maximum number of LOCKs held per CICS or IMS transaction</td>
<td>100</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_EDMFAILS</td>
<td>maximum number of EDM pool failures per minute</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_EDMLOADS</td>
<td>maximum percent of EDM pool requests needing loads</td>
<td>20%</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAXBFR_UNAVAIL</td>
<td>maximum number of buffer unavailable conditions per minute</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>Variable name</td>
<td>Description</td>
<td>Default</td>
<td>Data set member</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td>MAXALLOC_DELAY</td>
<td>maximum number of allocation delay conditions per minute</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_RIDUT</td>
<td>maximum RID pool percent utilization</td>
<td>80%</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_BFREXP</td>
<td>maximum number of buffer pool expansions per minute</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_BFRDMC</td>
<td>maximum number of Data Manager Critical conditions per minute</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_BFRIW</td>
<td>maximum number of Immediate Write conditions per minute</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MIN_GETRIO</td>
<td>minimum number of GETPAGES per read I/O in the INTVL</td>
<td>2.0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MIN_SWSPWS</td>
<td>minimum number of system page updates per system pages written in the INTVL</td>
<td>1.0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MIN_PWSWIO</td>
<td>minimum number of system pages written per write I/O in the INTVL</td>
<td>5.0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>RESP_MIN</td>
<td>number of minutes between monitor warning messages before triggering the EXEC again if necessary</td>
<td>1</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_SUSPCT</td>
<td>percent suspended threads (total active threads divided by total suspended threads)</td>
<td>25%</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_THDUTL</td>
<td>thread utilization (total active threads divided by maximum threads allowed)</td>
<td>90%</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_##SUSP</td>
<td>maximum suspensions per active thread per minute (susceptions per minute divided by number of active threads)</td>
<td>1</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_##DEAD</td>
<td>maximum deadlocks per minute in the interval</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>Variable name</td>
<td>Description</td>
<td>Default</td>
<td>Data set member</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td>MAX_#TIMEOUT</td>
<td>maximum timeouts per minute in the interval</td>
<td>0</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_PAGEPCT</td>
<td>maximum DB2 paging as a percent of total system (DB2 paging divided by total system paging)</td>
<td>10%</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_EDMUT</td>
<td>maximum EDM pool utilization</td>
<td>80%</td>
<td>DSU101C</td>
</tr>
<tr>
<td>MAX_OPENPCT</td>
<td>maximum open data set utilization</td>
<td>90%</td>
<td>DSU101C</td>
</tr>
</tbody>
</table>
Index

@DMPCLRO 174
$CSAABV 279
$CSABLO 279
$HASP050 163–165
$HASP093 163–165
$HASP095 163–165
$HASP355 164, 165
$HASP646 163–165
$JOBCPU 279
$STCCPU 279
$TOTCPU 192, 279
$TOTSIO 279
$TOTSIO2 279
$TSOCPU 192, 279
$WATCH 279

A

AAORUL00 273
abend of CICS 198
access view data
  QAOVBAPI EXEC 131
AEW messages 191
AlarmPoint solution 159, 162
ALERT application 191
ALERT profile, modifying 223
AUTOR field
  where displayed 22
AutoOPERATOR
  sample CICSPlex solution 280
  sample info/system management solution 282, 283
  sample monitoring solution 277

B

batch jobs 210

BBISSP00 175
BBPARM data set 273
BBPROC data set 273
BBUSER data set 282

C

CAT2291D 169
CER001C 196
CICS 197
CICS abnormal termination solution
  description 197
  EXECs 198
  invoking 197
  processing flow 198
  Rules 198
  variables 197
CICS performance management solutions 207
CICS solutions
  error recovery solutions 195, 197
  performance management solutions 207
CICS temporary storage suspensions solution
  EXECs 201
  invoking 201
  processing flow 201
  Rules 202
  variables 200
CICS, abend 198
client-server relationship, CSM 32
CMDCON parameter 175
CODE field 22
COMPLETE, CSM object state 117
conflicts, CSM schedules 99
connecting CSM objects 37
Continuous State Manager (CSM)
AATWAIT program
description 59
Abnormal Termination Events 39, 78
actual states 33
associating objects with a group 81, 149, 153
benefits 28
client-server relationship 32
command specification 73
conditional command specification 76
connecting 37
creating a group 69
cross-system management
description 58
enabling 102
defining objects 68
defining parents 83
defining servers 83
diagnosing errors 287
disconnecting 37
exceptional schedules, example 51
grouping object 30, 85
IPL, example 43
masking panels 64
messages 36
moving objects 82
normal object 30
objects 30
overview 27, 52
parent-child relationships 31
parents
adding 87
deleting 89
planning 53
commands 56
object relationships 54
schedules 55
recovering objects 39
routine schedules, example 50
schedules 34
conflicts 99
creating 94
modifying 95, 97
overriding 100
scheduling objects 39
serializing the shared repository
description 58
servers
adding 90
deleting 92
shutdown, example 46
specifying start/stop events 77
specifying user notification 79
start an object 103
Start Command Limits 39
starting commands 36
starting CSM 38
stopping commands 36
transient object 30, 85
Control-M solution
invoking 183
Rules 184
variables 183
conventions, documentation 19
CPU usage 188
CPU, sample monitoring solution 279
cross-system management, enabling 102
CSAABV monitor, sample monitoring solution 278
CSABLO monitor, sample monitoring solution 278
CSM Global Overview panel
description 62

CSM panel description
Client/Server Relationships 331
Command Specifications 316
Conditional Command Specifications 321
CSM Global Overview 305
Event Specifications 1 322
Event Specifications 2 325
Global Calendar Override 336
Group Detail Control 342
Object Detail Control 314
Object Groups 328, 339
Object Requisites 330
Object Scheduling 333
User Notification 327
CSMACT EXEC
commands 121
examples 120
modify objects 119
parameters 119
CSMINFO EXEC 118
CSSUM command, sample monitoring solution 278
customization
EXECs 223
initial 223

D

D_ALERT_IDS variable 232
D_ALERT_Q variable 232
Index

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

DB2 solutions
- DB2 global operations solutions 227
- DB2 performance management solutions 245
- resource contention analysis solutions 239
- DB2 subsystems, determining the number of 223
- DB2 table space filling up solution (DB2TFUL)
DESC field, documentation box 22
desired states 32
determining CSM schedules 94
DFH0508 196
DFH0606 197, 198
DFH0901 199, 200
DFH1311 203, 204
DFH3437I 205
disconnecting CSM objects 37
DISP field, documentation box 22
DMRALRT solution
EXECs 236
invoking 235
processing flow 236
Rules 236
variables 235
DMRUTIL solution
EXECs 237
invoking 237
Rules 238
variables 237
documentation box
description 22
documentation information 18
DOWNFORCE, CSM object state 115
DOWNLOGICAL, CSM object state 115
DOWNLOGICALE, CSM object state 115
DPE002C 250
DPE005C 250
DPE007C 250
DPE020C 247
DPE021C 247
DPE022C 247
DPE023C 247
DPE024C 247
DPE025C 247
DPE026C 247
DPE027C 247
DPE030C 241
DPE031C 241
DPE040C 243
DPE041C 243
DSU001C 234
DSU10nC 234
dump data sets monitoring solution
EXECs 174
invoking 173
processing flow 174
Rules 175
variables 173
DUMPCLER variable 173
DUMPMF variable 173
DUMPNDSS variable 173

E
E-mail solution 251, 252, 255
electronic documentation 17
enabling, cross-system management 102
ERB102I 181, 182
error recovery solutions 195, 197
exception monitoring solution
EXECs 193
invoking 191
processing flow 192
Rules 193
variables 191
EXEC Management application
documentation box 22
JESDOWN 345
EXECs 186

MainView AutoOPERATOR Solutions Guide
@DMPCLRO 174
$CSAABV 279
$CSABLO 279
$HASP050 164
$HASP093 164
$HASP095 164
$HASP355 164
$HASP646 164
$JOBCPU 279
$STCCPU 279
$STOTCPU 192, 279
$TOTSIO 279
$TOTSIO2 279
$TSOCPU 192, 279
$WATCH 279
BBPROC 273
CER001C 196
DFH0508 196
DFH0606 198
DFH131I 203
DPE002C 250
DPE005C 250
DPE007C 250
DPE020C 247
DPE021C 247
DPE022C 247
DPE023C 247
DPE024C 247
DPE025C 247
DPE026C 247
DPE027C 247
DPE030C 241
DPE031C 241
DPE040C 243
DPE041C 243
DSU001C 234
DSU010C 223
DSU10nC 234
ERB102I 182
IEA404A 176
IEA405E 176
IEA911E 174
IEA994A 174
IEA994E 174
IEC701D 171
IEE249I 176
IEE361I 178
IEE362A 178
IEE364I 178
IEE366I 178
IEE391I 178
IEE392I 178
IEE393I 178
IEE949I 178
IEE974I 178
IEF238D 171
IEF433D 171
IEF450I 283
IKT010D 222
IKT012D 222
IOS071I 190
JESDOW 345
LBSTART 188
MJSU001C 186
MJSU002C 186
MJSU003C 186
MJSU004C 186
MJSU005C 186
MJSU006C 186
MJSU007C 186
MJSU008C 186
MMN001C 178
MMN002C 179
MMN003C 279
MMN004C 178
MPE001C 190
MPE002C 190
MPE003C 188
MPE004C 188
MPE005C 188
MPE006C 188
MSU002C 186
MSU005C 186
national characters 275
NETACT 277
NETBOUNC 277
NETINACT 277
OSPINFO1 283
PM00101 186
PWSCPU01 193
PWSCPU02 193
PWSCSA01 193
PWSDV01 193
SCHEDCOM 184
TIOTSIO2 278
TLMS 169
failed objects, recovering 39
file allocation - deallocation solution
   EXECs 212
   invoking 211
   JCL 211
   processing flow 212
   Rules 212
   variables 211
file degradation analysis solution
   EXECs 210
   invoking 208
   processing flow 209
   Rules 210
   variables 208
FOCALPT variable 195
FT094W 215
FTP exit
   automating response with a Rule 128
   FTPPOSTPR 127
   messages 128
FUNC field 22

GROUP field 22
grouping objects
defining 85
description 30

Help
   online 17

IEA404A 176, 177
IEA405E 176, 177
IEA911E 173–175
IEA994A 173–175
IEA994E 173–175
IEC701D 170, 171
IEE249I 176
IEE3611 177–179
IEE362A 177–179
IEE364I 177–179
IEE366I 177–179
IEE3911 177–179
IEE392I 177, 178
IEE392II 179
IEE393I 177–179
IEE949I 178, 179
IEE974I 178, 179
IEF238D 170, 171
IEF433D 170, 171
IEF450L, info/system management solution 283
IFB040I 180, 181
IFB060E 180, 181
IFB070I 180, 181
IKT001D 221, 222
IKT010D 221, 222
IKT012D 221, 222
Index Term 140, 146
info/system management sample solution
   EXECs 283
   Rules 283
initial customization steps 223
initiators
   ACTIVE 188
   DRAINED 188
   HALTED 188
   inactive 188
Interval data solution
   variables 155
IOS071I 190

JES2 monitoring solution
   EXECs 164
   invoking 163
   processing flow 164
   Rules 165
   variables 163
JES2 solution
   See JES2 monitoring solution 163
job scheduling solution 185, 186
   See also CONTROL-M solution 183
JOBCPU monitor, sample monitoring solution 279
   jobs
   scheduling 185
K

K Q command 175

L

LBGCLAS variable 187, 188
LBGCPUL variable 187
LBGIBEG variable 187
LBGIDEC variable 187
LBGIEND variable 187, 188
LBGIINC variable 187
LBGPAGE variable 187
LBSTART 188
load balancing solution
EXECs 188
invoking 188
Rules 189
variables 187
LOGREC data set monitoring solution 179
LOGREC variable 179
LPAR data solution
variables 150

M

MainView Alarm Manager solution
ALRMSTOP 284
ALRMSTRT 284
enabling 283
MainView for CICS, MONITOR command 216
MainView for DB2
major messages 235
MainView for DB2 major messages
EXECs 236
invoking 235
processing flow 236
Rules 236
variables 235
MainView for DB2 utilities
EXECs 237
invoking 237
Rules 238
variables 237
MAX_EXTS variable 239
MAX_EXTSn variable 239

MAX_UTIL variable 239
MAX_UTILn variable 239
MAXACSA variable 239
MAXBCSA variable 191
MAXCCPU variable 191
MAXDEVU variable 191
MAXTCPU variable 191
MDEV command, sample monitoring solution 278
MJSU001C 186
MJSU002C 186
MJSU003C 186
MJSU004C 186
MJSU005C 186
MJSU006C 186
MJSU007C 186
MJSU008C 186
MMN001C 178
MMN002C, sample monitoring solution 279
MMN003C, sample monitoring solution 279
MMN004C 178
MONITOR command 216
monitoring solutions 173, 182
See also dump data sets monitoring solution,
LOGREC data set monitoring solution,
RMF monitoring solution, SMF data set
monitoring molution, WTO buffers
monitoring solution 173
MPE001C 190
MPE002C 190
MPE003C 188
MPE004C 188
MPE005C 188
MPE006C 188
MSU002C 186
MSU005C 186
MV solution samples
CICSPlex sample solution 280
info/system management sample solution 282, 283
VTAM/NCP sample solution 275
MVALARM solution
disabling 284
enabling 283
MVS performance management solutions
DASD reserve analysis 189
MVS solutions
JES2 solution 163, 165
See also JES2 monitoring solution 163
job scheduling solution 185, 186
See also CONTROL-M solution 183
monitoring solutions 173, 182
See also dump data sets monitoring solution, LOGREC data set monitoring solution, RMF monitoring solution, SMF data set monitoring solution, WTO buffers monitoring solution 173
performance management solutions 187
storage subsystems solutions 167
See also shared DASD control solution, storage reply solution, TLMsolution 167
VTAM/NCP solution 149, 153, 221

EXECs 279
invoking 278
processing flow 278
operator-initiated EXECs 273
OSPI application 282
OSPINFO1, info/system management solution 283
overriding CSM schedules 100, 336

page threshold counters 188
Paging sample 265
EXECs 265, 271
QAOSNPP1 265
QAOSNPP2 265
parent-child relationship, CSM 31
parents, CSM
adding 87
deleting 89
performance management solutions
CICS 207
MVS 187
PM00101 186
PROGRESS command, sample monitoring solution 279
publications, related 18
PWSCPU01 193
PWSCPU02 193
PWSCSA01 192, 193
PWSDEV01 192, 193

QAOSMTP1 252, 255
QAOSMTP1 sample EXEC 252, 255
QAOSMTP2 251
QAOSNMP sample EXEC 257
QAOSNMP1 sample EXEC 258
QAOVBAPI EXEC
access view data 131
example 132
parameters 132
return code 138
variables 136

N
NAMEn 239
naming convention
exceptions to 273
national characters 275
national characters 275
NETACT 277
NETBOUNC 277
NETINACT 277
normal objects, CSM
adding parents 83
adding servers 83
description 30
NUM 239

O
object relationships, CSM 31
objects, CSM
description 36
starting 103
ONLALMN variable 195
ONLALTN variable 195
ONLCNTN variable 195
online Help 17
ONLSYSN variable 195
ONLTYPEN variable 195
operations monitors sample solution
R

RECNUM variable, info/system management solution 283
related publications 18
repeat, CSM, line commands 86
REPLIES command 169
RESOLVE commands
   REPLIES 169
resources, inactivating and reactivating 276
response time 207
RMF monitoring solution 181, 182
Rule Processor application
   BBPARM 273
   IEF450I 283

Rules

$HASP050 165
$HASP093 165
$HASP095 165
$HASP355 165
$HASP646 165
activate 223
DFH0901 200
DFH1311 204
ERB102I 182
IEA404A 177
IEA405E 177
IEA911E 175
IEA994A 175
IEA994E 175
IEC701D 171
IEE361I 179
IEE362A 179
IEE364I 179
IEE366I 179
IEE391I 179
IEE392I 179
IEE393I 179
IEE949I 179
IEE974I 179
IEF238D 171
IFB040I 181
IFB060E 181
IFB070I 181
IKT001D 222
IKT010D 222
IKT012D 222
IOS071I 190
PWSCPU02 193
PWSCSA01 193
PWSDEV01 193
update 223

S

sample monitoring solution 277
SCHEDCOM 184
schedules
   See also CONTROL-M solution 183
schedules, CSM
conflicts 99
creating 94
description 34
determining 94
exceptional 35
modifying 95, 97
overriding 100
routine 34
scheduling Control-M solution
EXECs 184
invoking 183
variables 183
server, CSM
adding 90
deleting 92
service level objectives 216
shared DASD control solution
description 167
EXECs 168
invoking 168
processing flow 168
Rules 168
variables 167
shared device 177, 189
SMF data set monitoring solution
EXECs 178, 181
invoking 177, 180
processing flow 178, 180
Rules 179, 181
variables 177
SMFALT variable 177
SMFCLEAR variable 177
SNMP solution 257
EXECs 258, 262
QAOSNMP sample EXEC 257
QAOSNMP1 258
QAOSNMP1 sample EXEC 258
QAOSNMP3 262
QAOSNMP4 262
starting CSM objects 103
state, CSM
actual 33
desired 32
STCCPU monitor, sample monitoring solution 279
storage reply solution
EXECs 171
invoking 170
processing flow 170
Rules 171
variables 170
storage subsystems solutions
See shared DASD control solution, storage
reply solution, TLMS solution 167
storage violation solution
EXECs 196
invoking 196
processing flow 196
Rules 196
variables 195
SYS1.LOGREC data set 179
SYS1.MANx 177
SYSN variable 167, 189
SYSPROG AEW messages 191
SYSPROG commands
CPU 279
CSSUM 278
MDEV 278
PROGRESS 279
SYSPROG services EXECs 279
SYSPROG variable 163, 283
system dump data sets 173

T

tape
See storage subsystems solutions 167
TCAS reply solution
EXECs 222
invoking 221
processing flow 222
Rules 222
variables 221
temporary storage data set extension failure solution
EXECs 203
invoking 203
processing flow 203
Rules 204
variables 203
terminal errors solution
EXECs 205
invoking 205
processing flow 205
Rules 205
variables 204
text ID-driven EXECs 273
threshold level 223
TIOTSIO2, sample monitoring solution 278
TLMS solution
EXECs 169
invoking 169
processing flow 169
Rules 170
variables 169
TOTCPU monitor
EXECs 192
sample monitoring solution 279
TOTSIO monitor, sample monitoring solution 278
transaction response time monitor solution
EXECs 216
invoking 215
JCL 215
processing flow 216
Rules 216
variables 215
transient object, CSM 30
TSOCPU monitor
EXECs 192
sample monitoring solution 279

U
UBBPROC data set 273
UPEARLY, CSM object state 115
UPFORCE, CSM object state 115
UPLOGICAL, CSM object state 115

V
variables
CICS solution 357
DB2 solution 361, 365
MVS solution 355
view data

W
WATCH monitor, sample monitoring solution 279
WTO buffer shortages 175
WTO buffers monitoring solution
EXECs 176
invoking 176
processing flow 176
Rules 177
variables 175
WTORS
IKT001D 222
IKT010D 222
TCAS 221

QAOVBAPI EXEC 131
VSAM control area split monitor solution
EXECs 214
invoking 214
JCL 213
processing flow 214
Rules 214
variables 213
VSAM subtask abnormal termination solution
EXECs 199
invoking 199
processing flow 199
Rules 200
variables 198
VTAM commands sample solution
EXECs 277
inactivating a resource 276
invoking 276, 280
processing flow 277
reactivating a resource 276
Rules 277