BMC Runtime Component System Configuration and Administration Guide

Supporting

Version 6.1 of MainView Infrastructure
MainView products
Version 10.1 of APPTUNE for DB2®
Version 10.1 of BMC System Performance for DB2
Version 10.1 of Pool Advisor for DB2
Version 10.1 of SQL/Explorer for DB2
Version 10.1 of SQL Performance for DB2

September 2012
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**

**Address**
BMC SOFTWARE INC
2101 CITYWEST BLVD
HOUSTON TX 77042-2827
USA

**Telephone**
1 713 918 8800 or 1 800 841 2031

**Fax**
1 713 918 8000

**Outside United States and Canada**

**Telephone**
+01 713 918 8800

**Fax**
+01 713 918 8000


BMC, BMC Software, and the BMC Software logo are the exclusive properties of BMC Software, Inc., are registered with the U.S. Patent and Trademark Office, and may be registered or pending registration in other countries. All other BMC trademarks, service marks, and logos may be registered or pending registration in the U.S. or in other countries. All other trademarks or registered trademarks are the property of their respective owners.

CICS, DB2, DFSMS, IBM, IMS, MQSeries, MVS, OS/390, RACF, REXX, VTAM, WebSphere, z/OS, and zSeries are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is the registered trademark of Linus Torvalds.

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 UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES.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, 2101 CITYWEST BLVD, HOUSTON TX 77042-2827, USA. Any contract notices should be sent to this address.
Customer support

You can obtain technical support by using the BMC Support Central website or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, see “Before contacting BMC.”

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 issues similar to yours and possible solutions
■ order or download product documentation
■ download products and maintenance
■ report an issue or ask a question
■ subscribe to receive proactive e-mail alerts when new product notices are released
■ 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 issue
■ commands and options that you used
■ messages received (and the time and date that you received them)
  — product error messages
  — messages from the operating system, such as file system full
  — messages from related software
License key and password information

If you have questions about your license key or password, use one of the following methods to get assistance:

- Send an e-mail message to customer_support@bmc.com.
Contents

About this book

Related publications .......................................................... 13
Conventions ........................................................................ 14
Syntax statements .............................................................. 14

Chapter 1 Understanding RTCS

Introduction ................................................................. 17
Continuous system operation ........................................ 17
RTCS components ....................................................... 19
  RTCS kernel .......................................................... 19
  RTCS subsystem address space .................................. 20
  RTCS system registry ............................................... 20
  RTCS memory registry ............................................. 21
  RTCS product registry ............................................. 21
  RTCS Generalized Server address space ...................... 21
Understanding the RTCS system registry ....................... 22
  Examples of configuring the RTCS system registry .......... 22
  About the registry VLDS owner .................................. 24
  When to unallocate the registry VLDS ......................... 24
  Registry instance state transition ............................... 25

Chapter 2 Using RTCS product libraries

Library categories ......................................................... 27
  Distribution libraries ............................................... 28
  Target libraries ...................................................... 29
  Runtime production libraries .................................... 29
RTCS target libraries ..................................................... 31
  RTCS subsystem program library (TOSZRTCS) .......... 31
  RTCS product program library (TOSZLINK) ................ 31
  RTCS hypertext document library (TOSZHTML) .......... 32
  RTCS XML document library (TOSZRXXML) ............... 32
  Sample JCL and parameter library (TOSZCNTL) ........... 33

Chapter 3 Configuring RTCS

Introduction ................................................................. 37
Using OZI Customization ............................................... 38
Modifying an RTCS configuration .................................. 38
Manually configuring RTCS .................................................. 38
Creating POSZRTCS .......................................................... 40
Creating POSZLINK ............................................................. 42
Creating POSZHTML ............................................................ 43
Authorizing the RTCS libraries .............................................. 45
Allocating the RTCS system registry ....................................... 45
Ensuring sufficient common area data spaces ......................... 47
Ensuring sufficient system linkage indexes .............................. 47
Adding RTCS programs to the z/OS PPT ................................. 48
Selecting an SSID for the RTCS subsystem control address space .... 49
Updating the subsystem name table ....................................... 49
Creating and customizing the RTCS Initiator started task PROC .... 50
Creating the RTCS subsystem started task PROC ...................... 51
Creating the RTCS Generalized Server started task PROCs ........... 52
Establishing ESM user IDs for RTCS address spaces .................. 53
Starting RTCS ................................................................. 62
Adding the RTCS start command to SYS1.PARMLIB .................... 62

Chapter 4 RTCS administration .......................... 65

Backing up the RTCS system registry ................................. 65
Recovering the RTCS system registry ................................... 66
Starting RTCS ................................................................. 67
Stopping RTCS ................................................................. 68
RTCS operator commands. .................................................. 70
csbName parameter ....................................................... 70
command parameter ..................................................... 71
operand parameter ....................................................... 77
RTCS Security Manager .................................................... 78
Global security parameters ................................................ 79
Customizing RTCS global security parameters ....................... 81
Global class properties ..................................................... 82
Customizing RTCS global class properties ........................... 83

Appendix A RTCS Registry Maintenance Utility ............... 85

Introduction ................................................................. 85
Registry basics .............................................................. 86
Executing the RMU ........................................................... 86
Required DD statements ................................................... 87
SYSIN .............................................................. 87
SYSPRINT .......................................................... 87
BACKUP ......................................................... 87
RESTORE ...................................................... 88
Control statements ......................................................... 88
SET .............................................................. 88
SELECT ....................................................... 90
DELETE ....................................................... 91
BACKUP ...................................................... 92
RESTORE .................................................... 93
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPORT</td>
<td>94</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>95</td>
</tr>
<tr>
<td>Return codes and messages</td>
<td>96</td>
</tr>
<tr>
<td>Return codes</td>
<td>96</td>
</tr>
<tr>
<td>Messages</td>
<td>97</td>
</tr>
<tr>
<td>Appendix B Sample JCL and z/OS image configuration members</td>
<td>99</td>
</tr>
<tr>
<td>RTCS Initiator started task PROC</td>
<td>99</td>
</tr>
<tr>
<td>STEPLIB DD statement</td>
<td>101</td>
</tr>
<tr>
<td>RTCS subsystem started task PROC</td>
<td>101</td>
</tr>
<tr>
<td>RTCS Generalized Server started task PROC</td>
<td>102</td>
</tr>
<tr>
<td>z/OS Program Properties Table additions</td>
<td>103</td>
</tr>
<tr>
<td>Appendix C RTCS initialization member</td>
<td>105</td>
</tr>
<tr>
<td>Index</td>
<td>117</td>
</tr>
</tbody>
</table>
Figures

Registry instance state transition ................................................................. 26
RTCS product libraries .................................................................................. 28
Sample JCL to create RTCS subsystem program library ................................. 41
Sample JCL to create RTCS product program library ...................................... 43
Sample JCL to create RTCS hypertext document library ............................... 44
Sample JCL to create the RTCS system registry ........................................... 46
RACF example ............................................................................................... 53
CA ACF2 example .......................................................................................... 56
CA Top Secret example .................................................................................. 59
RTCS subsystem messages ............................................................................ 68
Generalized Server messages ......................................................................... 71
Registry Maintenance Utility (RMU) execution JCL ................................. 86
RMU DELETE NODE processing JCL ............................................................ 91
RMU DELETE VALUE processing JCL .......................................................... 92
RMU DELETE TREE FORCE processing JCL .............................................. 92
RMU BACKUP processing JCL ..................................................................... 93
RMU RESTORE processing JCL .................................................................. 93
RMU system registry IMPORT processing JCL ........................................... 95
RTCS initiator started task PROC ................................................................. 99
RTCS subsystem started task PROC ............................................................ 101
RTCS Generalized Server started task PROC ........................................... 102
z/OS Program Properties Table additions .................................................. 103
RTCS initialization member .......................................................................... 105
Tables

Tasks to manually configure RTCS .......................................................... 39
SET statements ....................................................................................... 73
Subcommands ......................................................................................... 74
Operands .................................................................................................. 77
Registry Import Utility return codes ......................................................... 96
RTCS Initiator started task PROC customizations .................................... 100
About this book

This book contains information about configuring and administering the Runtime Component System (RTCS).

This book is written for IBM z/OS® technical professionals and managers who are involved in the installation, deployment, and management of RTCS, as well as the configuration and management of z/OS systems to support those products.

To use this book, you should be familiar with the following items:

- configuration of your IBM z/OS images, DASD volumes, ICF catalog structure, data set naming conventions, and TCP/IP communication environment
- MVS job control language (JCL), MVS operator commands, and the installed external security manager (ESM)

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).

Related publications

From the BMC Support Central website (http://www.bmc.com/support), you can use either of the following methods to access related publications that support your product or solution:
Conventions

This book 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

- The symbol => connects items in a menu sequence. For example, Actions => Create Test instructs you to choose the Create Test command from the Actions menu.

Syntax statements

The following example shows a sample syntax statement:

```
COMMAND KEYWORD1 [KEYWORD2 | KEYWORD3] KEYWORD4={YES | NO} fileName...
```
The following table explains conventions for syntax statements and provides examples:

<table>
<thead>
<tr>
<th>Item</th>
<th>Example</th>
</tr>
</thead>
</table>
| Items in italic type represent variables that you must replace with a name or value. If a variable is represented by two or more words, initial capitals distinguish the second and subsequent words. | alias  
databaseDirectory  
serverHostName |
| Brackets indicate a group of optional items. Do not type the brackets when you enter the option. A comma means that you can choose one or more of the listed options. You must use a comma to separate the options if you choose more than one option. | [tableName, columnName, field]  
[-full, -incremental, -level] (UNIX) |
| Braces indicate that at least one of the enclosed items is required. Do not type the braces when you enter the item. | {DBDDName | tableName}  
UNLOAD device={disk | tape, fileName | deviceName}  
{-a | -c} (UNIX) |
| A vertical bar means that you can choose only one of the listed items. In the example, you would choose either commit or cancel. | {commit | cancel}  
{-commit | -cancel} (UNIX) |
| An ellipsis indicates that you can repeat the previous item or items as many times as necessary. | columnName . . . |
Understanding RTCS

This chapter provides conceptual information about the RTCS subsystem, the RTCS kernel, the RTCS system registry, the RTCS Generalized Server, and their support for BMC products.

Introduction

The Runtime Component System (RTCS) is an infrastructure that uses the latest z/OS system facilities to take advantage of modern enterprise servers. The RTCS infrastructure forms the basis for products that are designed to simplify product installation and configuration.

The RTCS infrastructure is established in a target z/OS image by a dedicated z/OS subsystem control address space. After being initialized, the RTCS subsystem control address space only executes code that is necessary to respond to operator commands and RTCS system registry operations.

All RTCS administrative functions and all products execute in either a dedicated product address space (PAS) or in an RTCS Generalized Server address space that is started by RTCS for the exclusive use by one or more products.

Continuous system operation

Continuous system operation refers to software and system availability without interruption or downtime.

After the RTCS subsystem has been started, products can be started and run. If any product address spaces (PASs) exist, any attempt to stop or cancel the RTCS subsystem address space while these PASs are still executing will be suppressed. As a consequence, RTCS subsystem and kernel software updates cannot easily be performed by shutting down and restarting RTCS.
The RTCS subsystem provides a facility for updating software and configuration information that enables the RTCS kernel and any loaded dynamic link library (DLL) packages to be refreshed dynamically without disruption to ongoing operation. In addition, changes that are made to the RTCS system registry are immediately effective, and the RTCS Registry Import Utility is provided to effect the changes.

SMP/E-based maintenance of program object elements in the RTCS subsystem program library cannot interfere with ongoing RTCS kernel or subsystem functions because the RTCS subsystem address space does not retain an allocation for the target or production library after the RTCS kernel is initialized. RTCS dynamically deallocates the data set, which permits the data set to be updated or reallocated. When the library is needed again (for example, to refresh the RTCS kernel after application of maintenance), the library is dynamically allocated.

**NOTE**

This process applies regardless of whether the SMP/E TOSZRTCS target library is being used directly or a production copy (POSZRTCS) of that library has been created for use in production on a particular z/OS image or shared among all members of the sysplex.

The RTCS kernel and DLL refresh facility dynamically allocate the program libraries with the original DSNAMEs. If these libraries are not the actual SMP/E target libraries, updated elements (or the entire library) must be copied to the original library that is being used in production. These DSNAMEs can be changed dynamically by an RTCS operator SET command (for more information, see “SET field=value” on page 73). Because these libraries are either dynamically deallocated or can be freed by stopping any product Generalized Server address spaces, updating the production libraries for this purpose is proper, and conventional program library utilities can safely be used to perform this function. In general, it is not necessary to shut down Generalized Server address spaces prior to performing maintenance using SMP/E or copying SMP/E target libraries to z/OS image production libraries.

**NOTE**

The kernel or DLL elements that you intend to refresh for maintenance or upgrade must have a date and time stamp that is more recent than the code that is being refreshed. To prevent regression to a back-level version of the code, the RTCS kernel and DLL refresh facility checks to ensure that maintenance has actually been applied or that the affected contents of the library have a more recent date.
**RTCS components**

RTCS services are established and maintained on each target z/OS image by a dedicated z/OS subsystem control address space. This address space is intended to run under the Master Subsystem (MSTR) and remain active for the life of the IPL, but it can be started under the primary JES and shut down when desired, provided that there are no dependent address spaces still executing.

The RTCS infrastructure comprises the following components:

- RTCS kernel
- RTCS subsystem address space
- RTCS system registry
- RTCS memory registry
- RTCS product registry
- RTCS Generalized Server address spaces

**RTCS kernel**

RTCS runs as a z/OS subsystem. During initialization, the RTCS subsystem loads the RTCS kernel into above-the-line common storage. All code that is directly used by products is physically located in the kernel; references to code in the RTCS address space are not allowed. After the RTCS subsystem has been successfully initialized and the RTCS kernel has been loaded, RTCS services are available to applications and products that have been written to use the services.

Only one instance of RTCS can execute on each z/OS image. After the RTCS kernel is initialized, the RTCS subsystem control address space remains in the system as the owner of certain z/OS resources. Other than responding to operator commands and remote requests for local system registry data, plus hardening any updates made to the system registry (which are infrequent), it is essentially idle, executing very little code.

Products interact only with the kernel and not directly with the RTCS subsystem address space. Only operator commands that are directed specifically to RTCS (such as MODIFY) and RTCS system registry activity will cause any instructions to be executed in the RTCS subsystem control address space. Operator commands are normally only required when you want to start RTCS Generalized Server (product) address spaces or to control RTCS system registry operations.
RTCS subsystem address space

The main responsibilities of the RTCS subsystem address space are to initialize the kernel and to provide the point of ownership for certain z/OS resources. z/OS exits and cross-memory resources that are established by the RTCS subsystem are used by functions and code in other permanent z/OS system address spaces, which never terminate. Because these RTCS functions, z/OS resources, and the RTCS kernel code are normally expected to be available for the life of the IPL, the RTCS subsystem address space should not be shut down.

Because the RTCS subsystem address space should not be stopped, starting it under any JES as a started task is not recommended. Instead, it should be started under the Master Subsystem by using SUB=MSTR on the START command.

Because the RTCS subsystem address space should not be shut down after it is started, only a limited number of data sets are allocated to it. The RTCS system registry VSAM linear data set (LDS) is the only data set that normally remains allocated.

The RTCS subsystem program library is a separate, dedicated SMP/E target library that is not shared by any other product. It is dynamically allocated by the RTCS subsystem address space and is deallocated after the RTCS kernel is initialized. The RTCS subsystem program library is required to be a partitioned data set extended (PDSE) program library.

RTCS system registry

The RTCS system registry is a repository for configuration parameters, product definitions, and other typical PARMLIB-type data. The RTCS system registry can be seen as read/write auxiliary storage for the RTCS subsystem and all products that use the registry. Data in the RTCS system registry is accessed as if it were in virtual storage, but it is actually stored on DASD in a VSAM LDS. The RTCS system registry is managed and accessed much like an z/OS page data set, except that it is organized in a tree-like structure, similar to the Microsoft Windows registry. This structure speeds up the process of locating data in the registry because complex, self-defining structures can be stored as easily as a single byte or a string of characters.

The RTCS system registry can optionally be shared with additional RTCS subsystems on other z/OS images in the same sysplex. In the shared mode, one of the RTCS address spaces becomes the local registry owner and manages all access to the RTCS system registry. The other RTCS address spaces become remote registry accessors and make requests to put data into the registry and to retrieve data from the registry to the local owner through an XCF communications pipeline.

For more information about the RTCS system registry, see “Understanding the RTCS system registry” on page 22.
RTCS memory registry

The RTCS memory registry is not backed by a VSAM VLDS and is intended to retain information temporarily as long as the RTCS subsystem is available. All content in the RTCS memory registry is lost when RTCS is shut down or the z/OS system image fails. The RTCS memory registry cannot be shared. There is one for each RTCS subsystem on each z/OS system image.

RTCS product registry

The RTCS product registry is a repository for product-specific information. It is defined by the products and is optionally backed by a VSAM LDS cluster.

NOTE

This VSAM LDS cluster must be allocated for use by the RTCS subsystem to back the primary RTCS registry partition using the z/OS Data-In-Virtual (DIV) service. Because it is a VSAM cluster and will be dynamically allocated by the RTCS subsystem address space using only its DSNAME, this library must be cataloged.

All processing activity uses the same access techniques as the RTCS system registry, and like the RTCS system registry is organized in a tree-like structure.

The RTCS product registry can optionally be shared with another copy of the product address space either on the same system or on a different z/OS image in the same sysplex. In the shared mode, one of the product address spaces becomes the local registry owner and manages all access to the RTCS product registry. The other product address spaces become remote registry accessors and make requests to put data into the registry and to retrieve data from the registry to the local owner through an XCF communications pipeline.

RTCS Generalized Server address space

Code for products does not execute in the RTCS subsystem address space. Instead, products execute either in their own address space or are started by RTCS in an RTCS Generalized Server address space.

A single RTCS Generalized Server address space can run multiple products, but all of the products in the address space must execute in the same protection key (keys 0 through 9 are supported). Multiple General Server address spaces might be started if products that execute in different protection keys need to be running at the same time.
Understanding the RTCS system registry

The RTCS system registry is backed by a VSAM LDS, whose DSNAME is specified by the SREGVLDS option in the RTCS initialization member of Logical PARMLIB. The z/OS Data-In-Virtual (DIV) service writes all changes that are made to the contents of the registry to the LDS.

Each z/OS image requires some form of access to the data in the registry VSAM LDS. A single VSAM LDS can be shared among all members of the sysplex or a dedicated VSAM LDS can be used for each individual z/OS system image. BMC recommends that you share a single VSAM LDS among all members of a sysplex. The MainView CAS places certain configuration information in the registry in order to make it available to all CASs in the sysplex. If you do not share a single VSAM LDS for the registry, you will not be able to take advantage of this CAS function.

A dedicated VSAM LDS can be configured for each RTCS subsystem or a single VSAM LDS can be shared by all members of the sysplex. If sharing a VSAM LDS, only one member actually allocates the registry VLDS with DISP=OLD, but the data in the registry is made available to other (remote) members of the sysplex using the XCF communications facility of z/OS.

Examples of configuring the RTCS system registry

The following examples illustrate different configurations for the RTCS system registry. The examples contain only the statements in the RTCS initialization member (OSZINIxx) that configure the sharing of the RTCS system registry. An example of an entire RTCS initialization member is in Appendix C, “RTCS initialization member.”

Example of separating production and test systems

The following statements would be used in OSZINIxx for the production systems to indicate that the production systems will

- communicate only with other production systems using XCF group RTCSPROD
- share registry hilevel.PRODUCTION.REGISTER

| SREGVLDS=hilevel.PRODUCTION.REGISTER |
| REGISTRY-XCF-GROUP=RTCSPROD |
| ELIGIBLE-OWNER |
| NOPRIVATE-REGISTER |
| SHARED-REGISTER |
Examples of configuring the RTCS system registry

The following statements would be used in OSZINIx for the test systems to indicate that the test systems will

- communicate only with other test systems using XCF group RTCSTEST
- share registry \textit{hilovel.TEST.REGISTRY}

\begin{verbatim}
SREGVLDS=hivel.TEST.REGISTRY
REGISTRY-XCF-GROUP=RTCSTEST
ELIGIBLE-OWNER
NOPRIVATE-REGISTRY
SHARED-REGISTRY
\end{verbatim}

**Example of adding an additional XCF group**

The following statements would be used in OSZINIx to define an additional XCF group that will

- communicate only with other systems using XCF group RTCSGRPn
- share registry \textit{hivel.GRPn.REGISTRY}

\begin{verbatim}
SREGVLDS=hivel.GRPn.REGISTRY
REGISTRY-XCF-GROUP=RTCSUSER
ELIGIBLE-OWNER
NOPRIVATE-REGISTRY
SHARED-REGISTRY
\end{verbatim}

**Example of a nonshared RTCS system registry**

The following statements would be used in OSZINIx to define an RTCS system registry that will

- co-exist with any other RTCS in a sysplex
- communicate with no other RTCS
- allocate and open but not expose, a private registry unique to the image where this RTCS is started

\begin{verbatim}
SREGVLDS=hivel.&SYSID..BMCMQ.REGISTRY
ELIGIBLE-OWNER
PRIVATE-REGISTRY
NOSHARED-REGISTRY
\end{verbatim}
About the registry VLDS owner

An XCF group member which has dynamically allocated the registry VLDS in a shared registry configuration is said to be the Local Owner. If the Local Owner of the registry has exposed the registry, the member is said to have acquired the registry VLDS. Exposed means that the other members in the XCF group are aware of the fact that the Local Owner member has dynamically allocated the registry VLDS and is allowing other, remote (non-Local Owner XCF group members) to access data in the registry.

The registry VLDS can be unexposed on the Local Owner member, in which case remote XCF group members will not be able to access data in the registry, but address spaces and applications on the Local Owner's z/OS system image will be able to. After being unexposed, the registry can be exposed again, which allows remote XCF group members to again access data in the registry.

When the Local Owner needs to relinquish ownership if the registry VLDS, the registry can be released, in which case any other member of the XCF group that is eligible to acquire the registry will attempt to do so. The first one that acquires the registry becomes the new Local Owner. If a specific new Local Owner is desired, that member of the XCF group can be designated in a REGISTRY TRANSFER command and it alone will then attempt to reacquire the registry VLDS. If the registry has been drained and unallocated, a REGISTRY ACQUIRE command may be used to have the XCF member on which it was entered attempt to dynamically allocate the registry VLDS, expose the registry, and reestablish remote instance access to the data in the registry for all members of the registry’s XCF group.

When to unallocate the registry VLDS

The purpose of being able to unallocate the registry VLDS is so that its contents can be backed up using normal VSAM utility and DASD backup programs, or so that DASD maintenance activities can take place, which might even include renaming or moving the location of the VSAM LDS. If the DSNAME of the registry VLDS is changed, the new DSNAME can be specified by the REGISTRY ALTER DSNAME command.

NOTE

Changing the DSNAME by using the REGISTRY ALTER DSNAME command does not alter the RTCS subsystem initialization parameters in Logical PARMLIB, nor in any parameter specification for a product that has its own registry. If you want to permanently change the DSNAME of the registry VLDS, you must appropriately update the location where this DSNAME is originally and uniquely specified.
To change the RTCS system registry VLDS to a different RTCS system registry VLDS while RTCS is running

1 Allocate a new RTCS system registry of the proper size.

2 Issue the following commands:

   F RTCS,REGISTRY UNEXPOSE
   F RTCS,REGISTRY UNALLOC
   F RTCS,REGISTRY ALTER DSNAMEnewSregvldsnName
   F RTCS,REGISTRY ACQUIRE

3 Change the OSZINlxx SREGVLDS parameter on all sharing systems to the new registry data set name.

Registry instance state transition

Figure 1 on page 26 illustrates the transition state of the registry VLDS, and includes the following information:

- RELEASE instructs the current Local Owner to unallocate the registry VLDS and to inform Remote XCF group members that it has done so, at which point all remote members that are eligible to be a Local Owner attempt to acquire the registry.

- TRANSFER instructs the current Local Owner to unallocate the registry VLDS and specifically indicates which Remote XCF group member is to attempt to acquire the registry in preference to all others.
Figure 1   Registry instance state transition

- VLDS Allocated
  - Exposed
  - Available on Local Owner and from Remotes
  - EXPOSE
  - UNEXPOSE
  - UNALLOCATE
  - REALLOCATE
  - RELEASE (note 1)
  - TRANSFER (note 2)
  - ACQUIRE

- VLDS Allocated
  - Not exposed
  - Available on Local Owner only (not from Remotes)

- Not allocated
  - Not exposed
  - Not available
Using RTCS product libraries

This chapter describes the product libraries that are used by the RTCS product family, and how those product libraries are created and used.

Library categories

RTCS is installed by using the BMC Installation System, which is SMP/E based. The BMC Installation System dictates the use of sets of distribution and target libraries for installation and maintenance of product function SYSMODs and PTFs. Most SMP/E-installed products can normally be run directly out of the SMP/E target libraries. For RTCS, BMC recommends using copies of the target libraries that are needed at runtime. These copies are called *production libraries* and might need to be z/OS image-specific.
Distribution libraries

Distribution libraries are constructed by SMP/E as part of function SYSMOD ACCEPT processing. They contain the individually-distributed and replaceable elements that are used to construct program objects and members in the target libraries. Distribution libraries are read-only and should not be modified or customized.

The RTCS distribution libraries are as follows:

- AOSZRTCS: RTCS Initiator, RTCS subsystem, and RTCS kernel
- AOSZLINK: RTCS utilities and HTTP server
- AOSC70D: only used by certain RTCS-based products
- AOSZHTML: hypertext documents
- AOSZRXML: XML documents
- AOSZCNTL: PROCs, sample JCL, and PARMLIB members
Target libraries

Target libraries are constructed by SMP/E as part of function SYSMOD or PTF APPLY processing. They contain program objects, hypertext documents, parameters, procedures, and samples. The target libraries should not normally be made available for production use on a target z/OS image.

The RTCS target libraries are as follows:

- TOSZRTCS: RTCS Initiator, RTCS subsystem, and RTCS kernel
- TOSZLINK: RTCS utilities and HTTP Server
- TOSZHTML: hypertext documents
- TOSZXML: XML documents
- TOSZCNTL: PROCs, sample JCL, and PARMLIB members

BMC recommends making copies of the following RTCS target libraries for use as production libraries on each z/OS image. These libraries (or, preferably, their production copies) are required on an z/OS image in production to execute RTCS. The following libraries are referred to collectively in this book as the runtime libraries:

- TOSZRTCS
- TOSZLINK
- TOSZHTML

For more information about each RTCS target library, see “RTCS target libraries” on page 31.

Runtime production libraries

Because the runtime libraries are PDSE data sets, consider making a dedicated copy of the runtime libraries for each target z/OS image on which you want to run RTCS in production. These copies are referred to as production libraries. There is no reason for the production libraries to be allocated as SMS-managed data sets. For example, they might be allocated on one of the target z/OS image’s SYSRES volumes. But they can be SMS-managed if you desire.

When you use production libraries, maintenance can be applied to the original SMP/E target libraries by using an allocation that specifies DISP=OLD. Subsequently, maintenance can be copied to the production libraries for use on one or more specific z/OS images.
Runtime production libraries

Applying maintenance to a product using SMP/E while the SMP/E target libraries are in use on an z/OS image might produce incorrect results or ABENDs when library members in the target library are accessed that are out of sync with other members that have already been accessed. Because of this possibility, BMC recommends that you do not configure the RTCS started task PROCs and the RTCS system registry to use your actual SMP/E target libraries on any production machine. However, you can configure a test z/OS image or LPAR to use the SMP/E-maintained target libraries directly if the consequences of an unexpected failure as a result of applying maintenance to libraries that are being used concurrently are deemed acceptable.

**NOTE**

You do not need to make production copies of the TOSZRXMNL and TOSZCNCNTL libraries because they contain samples and are used only during RTCS system registry initialization.

**Sharing PDSEs in a sysplex**

Because of PDSE sharing requirements, you might want to consider making a unique production copy of each runtime library for exclusive use on each target z/OS image. If your systems, however, are configured properly for PDSE sharing across all members of the sysplex, you can share the runtime libraries across multiple z/OS images if you adhere to the IBM® z/OS requirements for sharing PDSEs in a sysplex. These requirements include the following:

- The libraries can only be shared among members of a sysplex (never outside).
- The GRSplex must match the sysplex.
- No additional systems can access the PDSEs from outside of the sysplex or outside of the GRSplex. CA-MIM cannot be used to mediate shared access to PDSE libraries, even in read-only mode.

**NOTE**

If you do not strictly adhere to the IBM z/OS requirements, the PDSE might become corrupted, and products might subsequently fail in unpredictable ways. A corrupt PDSE library might cause system outages on one or more images on your sysplex.
The RTCS product family uses a limited number of product libraries. The contents of these target libraries are established by an SMP/E-based installation process.

This topic describes the RTCS target libraries. For more information, see “Target libraries” on page 29.

**RTCS subsystem program library (TOSZRTCS)**

The RTCS subsystem program library (TOSZRTCS) must be a PDSE and can be used as is in the RTCS Initiator PROC, where it must be defined in the STEPLIB DD statement. This library contains only RTCS subsystem PM3 (IBM DFSMS/MVS Release 4 compatible) program objects. This library is used only during RTCS subsystem initialization by the RTCS Initiator address space, and then the data set is dynamically allocated to the RTCS subsystem address space. After the RTCS kernel has been initialized, this library is dynamically deallocated, and it is not referenced again unless the RTCS kernel needs to be refreshed.

**RTCS product program library (TOSZLINK)**

The RTCS product program library (TOSZLINK) must be a PDSE. It can be used as is by specifying its DSNAME in the POSZLINK option in the RTCS initialization member of Logical PARMLIB. This library contains RTCS component PM3 (DFSMS/MVS release 4 compatible) program objects and contains no RTCS subsystem code.

This library is referenced and validated by the RTCS Initiator, which records its DSNAME. However, no programs in the library are executed during RTCS subsystem initialization. This data set is dynamically allocated by the RTCS Generalized Server program (OSZEXEC[n]) when an RTCS component is executed in a started task or batch job. Thus, this data set will be allocated to any address space that is running an RTCS component program or utility.
RTCS hypertext document library (TOSZHTML)

The members of the RTCS hypertext document library (TOSZHTML) are HTML documents or binary, byte-stream files. The RTCS hypertext document library contains variable-length record files used by RTCS utilities and product address spaces, including batch address spaces. RTCS batch utility programs, such as the Registry Import Utility, reference members of this library.

The RTCS hypertext document library is dynamically allocated by the RTCS Generalized Server, but only when it is known to be needed, whether in a started task or in batch. It is also used in a DD statement in the JCL used to execute certain RTCS utility programs in batch. The data set remains allocated to the Generalized Server or batch address space as long as it continues to execute. This data set is dynamically allocated by the RTCS Initiator address space to verify its existence and some of its contents, but it is not allocated by the RTCS subsystem address space.

Maintenance is more convenient if this data set is allocated as a PDSE. This data set can be used as is by specifying its DSNAME in the POSZHTML option in the RTCS initialization member of Logical PARMLIB.

RTCS XML document library (TOSZRXML)

The RTCS XML document library (TOSZRXML) is used by the RTCS Registry Maintenance Utility (RMU). This data set is never used by the RTCS subsystem control address space, nor by a Generalized Server address space because it only contains XML documents that are used in the initial setup of the RTCS system registry. The DSNAME of this library needs to be specified in a SET statement in the SYSIN input stream each time the RTCS RMU is executed. For more information, see Appendix A, “RTCS Registry Maintenance Utility.”
Sample JCL and parameter library (TOSZCNTL)

This library contains several samples that you can use to configure the RTCS started task PROCs and the RTCS initialization member. The library also contains sample JCL to execute the RTCS RMU and sample parameter input members for RTCS registry definitions.

The following TOSZCNTL library members are of particular interest after RTCS is installed and when a target z/OS image is being configured to run RTCS:

- OSZ$INIT
- OSZ$PARM
- OSZ$RTCS
- OSZ$EXEC
- OSZ$REGI
- OSZRBACK
- OSZRREST
- OSZSCHED

The following topics provide instructions on how to use these members.

**OSZ$INIT**

OSZ$INIT is sample RTCS Initiator started task procedure (PROC) JCL. Copy this member into SYS1.PROCLIB (or another Master Subsystem started task procedure library that is specified in the z/OS image’s Master JCL). BMC recommends naming this copy OSZINIT; however, this is not required.

Customize the STEPLIB DD statement to specify the DSNAME of the RTCS subsystem program library.

- If OSZINIT is to be started under the z/OS Master Subsystem (MSTR), the PROC must be located in SYS1.PROCLIB (or equivalent). You would then use the following operator command to start the RTCS Initiator:

  ```plaintext
  S OSZINIT, SUB=MSTR
  ```

- But if OSZINIT is to be started under the Primary JES, the PROC can be located in any JES-managed started task PROCLIB. You would then use the following operator command to start the RTCS Initiator:

  ```plaintext
  S OSZINIT
  ```
**OSZ$RTCS**

OSZ$RTCS is sample RTCS subsystem started task procedure (PROC) JCL.

If OSZINIT is started under the Master Subsystem, the PROC must be located in SYS1.PROCLIB (or equivalent) because OSZRTCS will also, by default, be started by the RTCS Initiator (internally) under the z/OS Master Subsystem (MSTR).

Because no parameters, data sets, or DD statements exist in this PROC, customization is unnecessary. Copy this member into SYS1.PROCLIB (or another Master Subsystem started task procedure library that is specified in the z/OS image’s Master JCL) if RTCS is to be started under the Master Subsystem, or into a JES-managed PROCLIB if RTCS is to be started under the primary JES. BMC recommends naming the copy OSZRTCS. You must specify the RTCS subsystem PROC name in the OSZRTCS-PROC option in the RTCS initialization member of Logical PARMLIB.

**OSZ$EXEC**

OSZ$EXEC is sample RTCS Generalized Server address space started task procedure (PROC) JCL.

Because no parameters, data sets, or DD statements exist in this PROC, customization is unnecessary.

Copy this member to both of the following locations:

- any JES started task procedure library
- SYS1.PROCLIB (or another Master Subsystem started task procedure library that is specified in the z/OS image’s master JCL)

BMC recommends naming the copy OSZEXEC. You must specify the RTCS Generalized Server PROC name in the OSZEXEC-PROC option in the RTCS initialization member of Logical PARMLIB.

**OSZ$REGI**

OSZ$REGI is sample JCL that is used to execute the RTCS Registry Maintenance Utility (RMU). Copy this member to a site-specific data set. In addition to the JOB statement, the JCL has DD statements that need to be customized to specify the DSNAMES of the SMP/E TOSZRXXML target XML document library as well as any customized XML document library.
OSZRBBACK

OSZRBBACK is sample JCL that is used to perform RMU backup processing. Copy this member to a site-specific data set. The BACKUP control statement in OSZRBBACK writes a portable, sequential backup of the data in an RTCS registry to the data set defined by the specified ddname. You must specify the ddname of the backup data set.

OSZRREST

OSZRREST is sample JCL that is used to perform RMU restore processing. Copy this member to a site-specific data set. The RESTORE control statement in OSZRREST restores the contents of the sequential backup data set defined by the specified ddname to the currently active registry. You must specify the ddname of the backup data set that you want to restore.

OSZSCHED

OSZSCHED is a sample set of required SYS1.PARMLIB PPT additions for RTCS. You can copy this member as is into the appropriate z/OS target image-specific PARMLIB data set. Copy it as a new member named SCHEDxx, or copy its contents into any existing PPT member.
This chapter explains how to configure a target z/OS image to be able to start the RTCS Initiator and initialize the RTCS subsystem.

Introduction

RTCS is installed by using the BMC Installation System. During the installation, you also configure RTCS by using OZI Customization.

- BMC recommends using OZI Customization to configure RTCS, as described “Using OZI Customization” on page 38.
- If you need to modify the RTCS configuration after using OZI Customization, use the information in “Modifying an RTCS configuration” on page 38.
- If you need to manually configure RTCS, use the information in “Manually configuring RTCS” on page 38.

The z/OS image configuration process must be performed so that the RTCS Initiator and the RTCS subsystem started task programs can receive control in the proper state and key from the z/OS initiator. This enables the RTCS subsystem control address space to establish RTCS in the z/OS image as a formal z/OS subsystem. Started task PROCs to start these address spaces must be placed in the proper STC procedure libraries (PROCLIBs). In addition, required ESM attributes must be established for the RTCS subsystem and Generalized Server address spaces. The required product data sets (or production copies that were made) must be accessible from the target z/OS image.
Using OZI Customization

When you use OZI Customization, it creates a default environment for a shared RTCS system registry at a sysplex level. You provide the name of the JCL procedures to start the RTCS initiator started task, the RTCS subsystem address space, and the RTCS Generalized Server started task. You also supply the sysplex name and the names of the z/OS images in the sysplex.

OZI Customization provides the ability to define one or more RTCS system registries all at the same time, you just specify the different sysplex names and z/OS image names on which you want the environments defined. The Online Help explains how to use OZI Customization to define multiple instances of an environment.

After you specify the necessary information, OZICustomization:

- Creates the jobs you need to run to implement the environment
- Creates one or more checklists for you to follow to complete any manual task that are needed

For more information about OZI Customization, see the Installation System User Guide.

Modifying an RTCS configuration

At times, you might need to modify the default RTCS environment created by OZI Customization. For example, if you do not want to share an RTCS system registry.

Use the information in Appendix B, “Sample JCL and z/OS image configuration members,” and Appendix C, “RTCS initialization member,” as a reference to modify the RTCS started task procedures, the z/OS program property table, or the RTCS initialization member as needed.

**NOTE**
Before making any modifications, complete OZI Customization, run the created jobs, and complete the tasks on the created checklists. At any time, you can rerun the created jobs to return to the default environment created by OZI Customization.

Manually configuring RTCS

If you do not use OZI Customization to configure RTCS, complete the steps in Table 1 on page 39 to manually configure a target z/OS image to run RTCS.
## Manually configuring RTCS

### Tasks to manually configure RTCS

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><em>(optional, but recommended)</em> Create the production RTCS subsystem program library (POSZRTCS).</td>
<td>&quot;Creating POSZRTCS” on page 40</td>
</tr>
<tr>
<td>2</td>
<td><em>(optional, but recommended)</em> Create the production RTCS product program library (POSZLINK).</td>
<td>&quot;Creating POSZLINK” on page 42</td>
</tr>
<tr>
<td>3</td>
<td><em>(optional, but recommended)</em> Create the production RTCS hypertext document library (POSZHTML).</td>
<td>&quot;Creating POSZHTML” on page 43</td>
</tr>
<tr>
<td>4</td>
<td>APF-authorize the RTCS subsystem program library and the RTCS product program library.</td>
<td>“Authorizing the RTCS libraries” on page 45</td>
</tr>
<tr>
<td>5</td>
<td>Allocate a VSAM LDS cluster for the RTCS system registry.</td>
<td>“Allocating the RTCS system registry” on page 45</td>
</tr>
<tr>
<td>7</td>
<td>Ensure the availability of sufficient common area data spaces.</td>
<td>“Ensuring sufficient common area data spaces” on page 47</td>
</tr>
<tr>
<td>8</td>
<td>Ensure the availability of sufficient system linkage indexes (LXs).</td>
<td>“Ensuring sufficient system linkage indexes” on page 47</td>
</tr>
<tr>
<td>9</td>
<td>Add RTCS programs to the z/OS PPT.</td>
<td>“Adding RTCS programs to the z/OS PPT” on page 48</td>
</tr>
<tr>
<td>10</td>
<td>Select an SSID for the RTCS subsystem control address space.</td>
<td>“Selecting an SSID for the RTCS subsystem control address space” on page 49</td>
</tr>
<tr>
<td>11</td>
<td>Guarantee the availability of an SSID for exclusive use by the RTCS subsystem.</td>
<td>“Updating the subsystem name table” on page 49</td>
</tr>
<tr>
<td>12</td>
<td>Create and customize the RTCS Initiator started task PROC.</td>
<td>“Creating and customizing the RTCS Initiator started task PROC” on page 50</td>
</tr>
<tr>
<td>13</td>
<td>Create the RTCS subsystem started task PROC.</td>
<td>“Creating the RTCS subsystem started task PROC” on page 51</td>
</tr>
<tr>
<td>14</td>
<td>Create the RTCS Generalized Server started task PROCs.</td>
<td>“Creating the RTCS Generalized Server started task PROCs” on page 52</td>
</tr>
<tr>
<td>15</td>
<td>Establish ESM user IDs for RTCS address spaces.</td>
<td>“Establishing ESM user IDs for RTCS address spaces” on page 53</td>
</tr>
<tr>
<td>16</td>
<td>Start RTCS.</td>
<td>“Starting RTCS” on page 62</td>
</tr>
<tr>
<td>17</td>
<td>Add the RTCS start command to SYS1.PARMLIB or the logical PARMLIB data set for the z/OS image.</td>
<td>“Adding the RTCS start command to SYS1.PARMLIB” on page 62</td>
</tr>
</tbody>
</table>
Creating POSZRTCS

Because of PDSE sharing requirements, consider making a unique copy of this library for exclusive production use on each individual target z/OS image on which RTCS is to be started. For more information about production libraries and their advantages, see “Runtime production libraries” on page 29. Alternatively, you can use a single data set, shared among all members of a single sysplex. For information about sharing this library in a sysplex, see “Sharing PDSEs in a sysplex” on page 30.

Requirements

The following requirements apply when creating POSZRTCS:

- The POSZRTCS library that you create must be a PDSE library. BMC recommends that it be allocated as a non-SMS-managed PDSE (that is, no STORCLAS specified), but it can be SMS-managed if desired.

- The name of the data set that you create must be specified by the DSNAME parameter on the STEPLIB DD statement in the RTCS Initiator PROC (see “Creating and customizing the RTCS Initiator started task PROC” on page 50).

  This is the only RTCS production library whose DSNAME needs to be specified in a DD statement in a started task PROC. If each of the remaining RTCS production libraries are allocated with a DSNAME prefix that matches this library, along with its recommended low-level index, the RTCS Initiator will be able to derive a matching, default DSNAME for them, which the RTCS subsystem will then use to dynamically allocate these data sets. If not, then the DSNAMEs of the non-conforming production libraries will need to be specified by the appropriate parameters in the RTCS Initialization member (which is usually, OSZINI100) of the z/OS Logical PARMLIB data set.

- Because it will be dynamically allocated by the RTCS Allocator (OSZMOSYS, the first program to run in the RTCS Subsystem address space), this library must be cataloged. The catalog must be available when RTCS is STARTed.

- The low-level index of the DSNAME selected for this production library should be POSZRTCS or OSZRTCS.

To create POSZRTCS

To allocate and copy the target SMP/E RTCS subsystem program library (TOSZRTCS) into a new POSZRTCS PDSE, you can customize the sample JCL shown in Figure 3 on page 41. The text that should be customized is shown in bold. This sample JCL is contained in member OSZINJ71 in the TOSZCNTL target library.
Figure 3  Sample JCL to create RTCS subsystem program library

```jcl
//OSZINJ71 JOB (acct),'CREATE POSZRTCS PDSE',
// MSGLEVEL=(1,1),CLASS=A,TIME=1
//*
//* acct  JOB statement accounting information
//* xxxxxxx VOLSER on which to allocate .POSZRTCS PDSE
//* tlibprfx DSN prefix of RTCS SMP/E target libraries
//*
//*ALLOC EXEC PGM=IEFBR14
//*
//* Allocate a new non-SMS-managed cataloged PDSE for
//* the RTCS Subsystem Library. This data set may be
//* SMS-managed, but the JCL will have to be changed
//* to allocate an SMS-managed data set. There is no
//* particular need for the library to be SMS-managed.
//* This data set CAN be shared by more than one system
//* if MVS sysplex PDSE sharing requirements are met.
//*
//* //OSZRTCS DD DSN=SYS2.RTCS.POSZRTCS,
/// DISP=(NEW,CATLG),
/// UNIT=3390,VOL=SER=xxxxxx,
/// DSNTYPE=LIBRARY,SPACE=(CYL,(16,5,01)),
/// DSORG=PO,RECFM=U,LRECL=0,BLKSIZE=23476
//*
//*COPY EXEC PGM=IEBCOPY,TIME=1,REGION=4M
//*
//* Copy SMP/E Target RTCS Subsystem LIBRARY
//* to the newly-allocated "production" PDSE.
//*
//*SYSPRINT DD SYSOUT=*  
//*TOSZRTCS DD DISP=SHR,DSN=tlibprfx.TOSZRTCS
//*OSZRTCS DD DISP=SHR,DSN=*.ALLOC.OSZRTCS
//*SYSIN DD *
//COPY INDD=((TOSZRTCS,R)),OUTDD=OSZRTCS
//*
```
Creating POSZLINK

Because of PDSE sharing requirements, consider making a unique copy of this library for exclusive production use on each individual target z/OS image on which RTCS is to be started. For more information about production libraries and their advantages, see “Runtime production libraries” on page 29. Alternatively, you can use a single data set, shared among all members of a single sysplex. For information about sharing this library in a sysplex, see “Sharing PDSEs in a sysplex” on page 30.

Requirements

The following requirements apply when creating POSZLINK:

- The POSZLINK library that you create must be a PDSE library.

- The low-level index of the DSNAME selected for this production library should be POSZLINK or OSZLINK.

- The name of the data set that you create will need to be specified in the RTCS Initialization member in the POSZLINK parameter if its low-level index is not POSZLINK or OSZLINK.

- Because it will be dynamically allocated by the RTCS Allocator (OSZMOSYS) to the RTCS subsystem address space using only its DSNAME, this library must be cataloged.

To create POSZLINK

To allocate and copy the target SMP/E RTCS product program library (TOSZLINK) into a new POSZLINK PDSE, you can customize the sample JCL shown in Figure 4 on page 43. The text that should be customized is shown in bold. This sample JCL is contained in member OSZINJ73 in the TOSZCNTL target library.
Because of PDSE sharing requirements, consider making a unique copy of this library for exclusive production use on each individual target z/OS image on which RTCS is to be started. For more information about production libraries and their advantages, see “Runtime production libraries” on page 29. Alternatively, you can use a single data set, shared among all members of a single sysplex. For information about sharing this library in a sysplex, see “Sharing PDSEs in a sysplex” on page 30.

Requirements

The following requirements apply when creating POSZLINK:

- The POSZHTML target library that you create must be a PDSE library.
■ The low-level index of the DSNAME selected for this production library should be POSZHTML or OSZHTML.

■ The name of the data set that you create will need to be specified in the RTCS Initialization member in the POSZHTML option if its low-level index is not POSZHTML or OSZHTML.

■ Because it will be dynamically allocated using only its DSNAME, this library must be cataloged.

To create POSZHTML

To allocate and copy the target SMP/E RTCS hypertext document library (TOSZHTML) into a new PDSE, you can customize the sample JCL shown in Figure 5. The text that should be customized is shown in bold. This sample JCL is contained in member OSZINJ75 in the TOSZCNTL target library.

Figure 5  Sample JCL to create RTCS hypertext document library

```
//OSZINJ75 JOB (acct),'CREATE POSZHTML PDSE',
  MSGLEVEL=(1,1),CLASS=A,TIME=1
/**
  acct      JOB statement accounting information
  zzzzzz    VOLSER on which to allocate .POSZHTML PDSE
  tlibprfx  DSN prefix of RTCS SMP/E target libraries
/**
//ALLOC   EXEC PGM=IEFBR14
/**
  Allocate a new non-SMS-managed cataloged PDSE for
  the RTCS HTML Doc Library. This data set may be
  SMS-managed, but the JCL will have to be changed
  to allocate an SMS-managed data set. There is no
  particular need for the library to be SMS-managed.
  This data set CAN be shared by more than one system
  if MVS sysplex PDSE sharing requirements are met.
/**
//OSZHTML   DD DSN=SYS2.RTCS.POSZHTML,
  DISP=(NEW,CATLG),
  UNIT=3390,VOLSER=zzzzzz,
  DSNTYPE=LIBRARY,SPACE=(CYL,(25,5,1)),
  DSORG=PO,RECFM=VB,LRECL=260,BLKSIZE=6504
/**
//COPY    EXEC PGM=IEBCOPY,TIME=1,REGION=4M
/**
  Copy SMP/E Target RTCS HTML Doc LIBRARY
  to the newly-allocated "production" PDSE.
/**
//SYSPRINT DD SYSOUT=* 
//TOSZHTML DD DISP=SHR,DSN=tlibprfx.TOSZHTML
//OSZHTML DD DISP=SHR,DSN=*.ALLOC.OSZHTML
//SYSIN    DD *
  COPY INDD=((TOSZHTML,R)),OUTDD=OSZHTML
/**
```
Authorizing the RTCS libraries

Temporarily or permanently APF-authorize the RTCS subsystem program library and the RTCS product program library.

**NOTE**
You need to completed this task only once per sysplex if you are sharing data sets. Otherwise, this task needs to be completed for each target z/OS image.

In the following examples, `libraryName` should be `POSZRTCS` for the RTCS subsystem program library or `POSZLINK` for the RTCS product program library, and `zzzzzz` is the VOLSER for the library. The following command examples assume that the production libraries were allocated with the DSNAMEs indicated in the sample JCL in Figure 3 on page 41 and Figure 4 on page 43. If you used different DSNAMEs, change the names in the commands accordingly.

**To temporarily APF-authorize these libraries**

Issue one of the following commands to add each library to the APF list:

- If the library is not SMS-managed (recommended)
  ```
  SETPROG APF,ADD,LIBRARY=SYS2.RTCS.libraryName,VOLUME=zzzzzz
  ```
- If the library is SMS-managed
  ```
  SETPROG APF,ADD,LIBRARY=SYS2.RTCS.libraryName,SMS
  ```

**To permanently APF-authorize these libraries**

1. Add the library’s DSNAME to the PROGxx member of SYS1.PARMLIB (or the z/OS image-specific Logical PARMLIB data set) that is appropriate for your site.
2. IPL the z/OS image or issue the `SET PROG=xx` z/OS operator command, which causes z/OS to rebuild the list of APF-authorized data sets in storage.

**Allocating the RTCS system registry**

Allocate a VSAM LDS cluster for the RTCS system registry, either by allocating

- one for each z/OS image if you are not going to share the registry
- one to be shared sysplex-wide by all members of the sysplex
Perform the following steps to allocate the RTCS system registry:

1 Select a VSAM LDS cluster DSNAME for the REGISTRY data set. Its low-level index should be REGISTRY, but this is not required. If its DSNAME does not follow the same naming convention that you have already used for the POSZRTCS, POSZLINK, and POSZHTML data sets, or its low-level index is not REGISTRY, you will have to specify the name of the LDS cluster in the SREGVLDS option of the RTCS Initialization member of Logical PARMLIB.

To allocate a new VSAM LDS to back the RTCS system registry, you can customize the sample JCL shown in Figure 6. The text that should be customized is shown in bold. This sample JCL is contained in member OSZINJ80 in the TOSZCNTL target library.

**Figure 6 Sample JCL to create the RTCS system registry**

```cll
//OSZINJ80 JOB (acct),'ALLOC REGISTRY VLDS',
// MSGLEVEL=(1,1),CLASS=A,TIME=1
//*
// acct      JOB statement accounting information
// yyyy      VOLSER on which to allocate .REGISTRY
//*
//CREATE  EXEC PGM=IEFBR14
//*
// Allocate a new VSAM Linear Data Set
// CLUSTER for the RTCS System Registry
//*
//REGISTRY DD DISP=(NEW,CATLG).
// VOL=SER=yyyy,UNIT=3390,
// DSN=SYS2.RTCS.REGISTRY,
// RECOROG=LS,AMP=AMORG,
// SPACE=(4096,(17472,0))
//*
```

2 Determine the amount of space, expressed in terms of the number of 4K blocks, to allocate for the VSAM LDS.
The minimum amount of space you should allocate for the RTCS system registry VSAM LDS is 17,472 blocks (68.25 MB), which requires 98 cylinders on a 3390.

The RTCS system registry is never likely to require more than 371 cylinders (260.85 MB) of space, even in very large configurations. For sites that do not run more than one MainView CAS (or do not run a CAS at all), 188 cylinders (132.1 MB) should be adequate. Sites that run more than one CAS or that have very large configurations might require 371 cylinders.

Allocate the library using the information you have gathered.

**Ensuring sufficient common area data spaces**

The RTCS subsystem requires two SCOPE=COMMON data spaces.

---

**NOTE**

Complete this task for each target z/OS image.

---

The MAXCAD parameter in member IEASYSxx of SYS1.PARMLIB or a z/OS system image Logical PARMLIB data set specifies the maximum number of SCOPE=COMMON data spaces that will be available for the life of the IPL.

If you have difficulty starting RTCS because no SCOPE=COMMON data spaces are available, increase the value of the MAXCAD parameter, and IPL the z/OS image.

**Ensuring sufficient system linkage indexes**

The RTCS subsystem requires only one system linkage index (LX).

---

**NOTE**

Complete this task for each target z/OS image.

---

The NSYSLX parameter in member IEASYSxx of SYS1.PARMLIB or a z/OS system image Logical PARMLIB data set specifies the number of additional system LXs that are to be reserved for use by authorized applications and products on this system image.
If you have difficulty starting RTCS because no system LXs are available, increase the value of the NSYSLX parameter, and reIPL the z/OS image.

## Adding RTCS programs to the z/OS PPT

The RTCS Initiator, the RTCS subsystem, and the RTCS Generalized Server require that they be executed in a started task address space with certain z/OS-assigned special attributes. Definitions for the job step program used in these PROCs in the z/OS Program properties table (PPT) cause the required attributes to be assigned by z/OS. Before RTCS can be started, the z/OS PPT must be updated to include the entries required for RTCS.

**NOTE**

You need to completed this task only once per sysplex if you are sharing data sets. Otherwise, this task needs to be completed for each target z/OS image.

Use one of the following methods to add RTCS programs to the PPT by using member OSZSCHED of the SMP/E TOSZCNTL target library.

- Update an existing SCHED<xx> member by performing the following tasks:
  1. Merge the contents of member OSZSCHED into an existing SCHED<xx> member.
  2. Refresh the PPT by issuing the following z/OS operator command:

    `SET SCH=(<xx>)`

    The variable <xx> is the suffix of any existing SCH member.

- Create a new SCHED<y> member by performing the following tasks:
  1. Copy the sample OSZSCHED member into SYS1.PARMLIB (or a logical parameter library data set), and name it SCHED<y>.
  2. Update the appropriate IEASYS<xx> member to specify this new SCHED<y> member and any existing SCH members.
  3. Refresh the PPT to have the additional entries in the new member become effective by issuing the following z/OS operator command:

    `SET SCH=(<xx>,<y>)`

    The variable <xx> is the suffix of any existing SCH member (named SCHED<y>).
Selecting an SSID for the RTCS subsystem control address space

Select an available, 4-character z/OS subsystem ID (SSID) for use by the RTCS subsystem control address space.

BMC recommends using **RTCS** for the SSID, but you can select any sequence of four characters.

---

**NOTE**
You need to completed this task only once per sysplex if you are sharing data sets. Otherwise, this task needs to be completed for each target z/OS image.

---

**WARNING**
The contents of all SCH member suffixes in a SET command replace any existing entries in storage. Therefore, you **must** include any existing PPT member suffixes in the list in addition to any new suffixes that you add.

---

**Updating the subsystem name table**

*(optional)* To guarantee the availability of an SSID for exclusive use by the RTCS subsystem, update the z/OS subsystem name (SSN) table.

---

**NOTE**
You need to completed this task only once per sysplex if you are sharing data sets. Otherwise, this task needs to be completed for each target z/OS image.

---

1. Add the SSID that you selected to the IEFSSN.xx member in SYS1.PARMLIB (or the appropriate Logical PARMLIB data set for the target z/OS image).

   For example, to add an entry with the SSID of **RTCS**, add the following statement to the appropriate IEFSSN.xx member of SYS1.PARMLIB:

   ```
   SUBSYS SUBNAME(RTCS)
   ```

2. Update the z/OS Subsystem Name Table by either performing an IPL of the target z/OS image or by using the following z/OS operator command:

   ```
   SETSSI ADD,SUBNAME=RTCS
   ```
Creating and customizing the RTCS Initiator started task PROC

Create and customize the RTCS Initiator started task PROC by performing the following steps.

**NOTE**
- A model RTCS Initiator started task PROC (OSZINIT) is provided in Figure 19 on page 99.
- You need to complete this task only once per sysplex if you are sharing data sets. Otherwise, this task needs to be completed for each target z/OS image.
- You will use the RTCS Initiator started task PROC that is created in this step to start RTCS. The RTCS subsystem address space is *not* started directly. When you run the RTCS Initiator started task PROC, it performs certain z/OS system checks, validates that the product’s production data sets have been properly allocated, and internally starts the RTCS subsystem address space.

The DSNAMEs that are specified in the RTCS Initialization member (or defaulted to based on the DSNAME of the RTCS subsystem program library) and dynamically allocated by the RTCS Initiator started task are recorded in ECSA storage. That information is then used to dynamically allocate these data sets, when and as required, in the RTCS subsystem and Generalized Server address spaces.

1. Select a name for the RTCS Initiator started task PROC.

   This name will be used in an z/OS START command to start the RTCS Initiator address space, preferably under the Master Subsystem (MSTR). BMC recommends using the name OSZINIT, but this is not required.

2. Copy member OSZ$INIT from the SMP/E TOSZCNTL target library to SYS1.PROCLIB or a JES-managed started task procedure library using the member name that you selected in step 1.

   BMC recommends that you start RTCS under the Master Subsystem (by using SUB=MSTR on the START command). But this is not absolutely required.

   If you start RTCS under the Master Subsystem and you use an unmodified MSTJCL00, you must use SYS1.PROCLIB specifically for those started tasks that are started under the Master Subsystem. Otherwise, the PROC does not need to be created in SYS1.PROCLIB, and you can use any started task procedure library that is defined in the IEFPDSI DD statement concatenation in the Master Scheduler JCL (member MSTJCLxx of SYS1.PARMLIB).

   If you start RTCS under your primary JES, you can place the PROC in any JES-managed started task PROCLIB.
3 Modify the STEPLIB DD statement in the new PROC to specify the DSNAME of the RTCS subsystem program library (POSZRTCS). For example, you can use the PDSE that was created in Figure 3 on page 41.

Creating the RTCS subsystem started task PROC

Create the RTCS subsystem started task PROC by performing the following steps.

NOTE

- You need to completed this task only once per sysplex if you are sharing data sets. Otherwise, this task needs to be completed for each target z/OS image.

- A model RTCS subsystem started task PROC (OSZRTCS) is provided in Figure 20 on page 101. Other than selecting the actual PROC name and possibly making some additions to the EXEC statement to adhere to local conventions and requirements, such as adding an ACCT parameter, no further changes should be made to the model PROC. In particular, no DD statement should be present.

1 Select a name to be used for the RTCS subsystem started task PROC.

BMC recommends using the name OSZRTCS, but this is not required. This name will be used in an z/OS START command that the RTCS Initiator (OSZSIRIS) issues to start the RTCS subsystem address space.

2 Copy member OSZ$RTCS from the SMP/E TOSZCNTL target library to SYS1.PROCLIB or a JES-managed started task procedure library using the member name that you selected in step 1.

If you start RTCS under the Master Subsystem and you use an unmodified MSTJCL00, you must use SYS1.PROCLIB specifically for those started tasks that are started under the Master Subsystem. Otherwise, the PROC does not need to be created in SYS1.PROCLIB, and you can use any started task procedure library that is defined in the IEPDSI DD statement concatenation in the Master Scheduler JCL (member MSTJCLxx of SYS1.PARMLIB).

If you start RTCS under your primary JES, you can place the PROC in any JES-managed started task PROCLIB.
Creating the RTCS Generalized Server started task PROCs

The RTCS Generalized Server PROC must reside in some data set (but not necessarily in the same data set) that is allocated to each of the following PROCLIB concatenations:

- Master Subsystem procedure library
- JES-managed started task procedure library

The PROC might reside in a single data set that appears in both concatenations, depending on your site’s PROCLIB configuration. If the PROC is to reside in two separate data sets (one for each PROCLIB concatenation), then each PROC must be identical and must have the same member (PROC) name.

**NOTE**
- You need to completed this task only once per sysplex if you are sharing data sets. Otherwise, this task needs to be completed for each target z/OS image.
- A model RTCS Generalized Server started task PROC (OSZEXEC) is provided in Figure 21 on page 102.

Perform the following steps to create the RTCS Generalized Server started task PROCs:

1. Select a name to be used for the RTCS Generalized Server started task PROCs.
   - BMC recommends using the name OSZEXEC, but this is not required.
   - If this PROC needs to be placed in two concatenations, you must use the same name for both PROCs. This name will be used in START commands that will be issued internally by RTCS.

2. Copy member OSZ$EXEC from the SMP/E TOSZCNTL target library to any valid Master Subsystem PROCLIB and use the member name that you selected in step 1.
   - This PROC must exist in a Master Subsystem PROCLIB, because RTCS Generalized Server address spaces are sometimes started by RTCS with a specification of SUB=MSTR.
   - If you use an unmodified MSTJCL00, you must use SYS1.PROCLIB specifically for those started tasks that are started under the Master Subsystem. Otherwise, the PROC does not need to be created in SYS1.PROCLIB, and you can use any started task procedure library that is defined in the IEFPPDSI DD statement concatenation in the Master Scheduler JCL (member MSTJCLxx of SYS1.PARMLIB).
Copy member OSZ$EXEC from the SMP/E TOSZCNTL target library to any valid JES-managed started task PROCLIB and use the member name that you selected in step 1 on page 52.

This PROC must exist in a JES-managed PROCLIB, because RTCS Generalized Server address spaces are usually started by RTCS without specifying SUB=MSTR. In this case, the RTCS Generalized Server runs under the primary JES, and its PROC is expanded by using the usual JES-managed started task PROCLIBs.

Establishing ESM user IDs for RTCS address spaces

Establish ESM user IDs for RTCS address spaces using the procedure that is correct for the ESM that is executing on the z/OS image.

NOTE
You need to complete this task only once per sysplex if you are sharing data sets. Otherwise, this task needs to be completed for each target z/OS image.

Appropriately defined, valid ESM user IDs must be assigned to the three RTCS started task address spaces (RTCS Initiator, RTCS subsystem, and RTCS Generalized Server) by z/OS and the ESM when these address spaces are started. The procedure to assign these user IDs depends on the specific ESM that is executing on the z/OS image. These user IDs must be given access to the production product libraries that will be accessed by the address spaces that are running under the assigned user IDs.

Figure 7 (IBM RACF®), Figure 8 on page 56 (CA ACF2), and Figure 9 on page 59 (CA Top Secret) provide examples that you can customize. Use the example that applies to the ESM that is executing on the z/OS image. Because of the complex and sometimes subtle interaction between the data set naming conventions in place and existing profiles, permits, or rules, the following examples might require extensive modifications for use by your site. The sample JCL is contained in member OSZ$RACF, OSZ$ACF2, or OSZ$TSS in the TOSZCNTL target library.

Figure 7 RACF example (part 1 of 4)
Establishing ESM user IDs for RTCS address spaces

NASDAQ RACF example (part 2 of 4)
/* Define data set profiles for the PDSE libraries & System Registry. */
/* We define them initially with UACC(ALTER) so that the current ID */
/* does not have to have the SPECIAL attribute to issue the PERMIT */
/* commands that immediately follow. We will alter the UACC in the */
/* data set profiles after ALTER access by the current user is set. */
ADDSD 'SYS2.RTCS.POSZRTCS' -
   GENERIC UACC(ALTER)
ADDSD 'SYS2.RTCS.POSZLINK' -
   GENERIC UACC(ALTER)
ADDSD 'SYS2.RTCS.POSZHTML' -
   GENERIC UACC(ALTER)
ADDSD 'SYS2.RTCS.POSZPSWD' -
   GENERIC UACC(ALTER)
ADDSD 'SYS2.RTCS.REGISTRY' -
   GENERIC UACC(ALTER)
/* Ensure that the system programmer executing the RTCS configuration */
/* JOBs has ALTER access to the RTCS Subsystem and Product Libraries, */
/* the Hypertext Document Library, and the RTCS System Registry VLDS. */
/* The absence of OWNER(id) does not grant such access unless the "*" */
/* UserID has the SPECIAL attribute, anyway. The point of all of this */
/* convolution is to obviate the requirement that a system programmer */
/* running these JOBs have the SPECIAL attribute in the first place. */
PERMIT 'SYS2.RTCS.POSZRTCS' -
   GENERIC ID(*)    ACCESS(ALTER)
PERMIT 'SYS2.RTCS.POSZLINK' -
   GENERIC ID(*)    ACCESS(ALTER)
PERMIT 'SYS2.RTCS.POSZHTML' -
   GENERIC ID(*)    ACCESS(ALTER)
PERMIT 'SYS2.RTCS.POSZPSWD' -
   GENERIC ID(*)    ACCESS(ALTER)
PERMIT 'SYS2.RTCS.REGISTRY' -
   GENERIC ID(*)    ACCESS(ALTER)
/* Update data set profiles for the PDSE libraries & System Registry. */
/* We set the minimum UACC needed by a typical, non-strict, customer. */
ALTDSD 'SYS2.RTCS.POSZRTCS' -
   GENERIC UACC(NONE)
ALTDSD 'SYS2.RTCS.POSZLINK' -
   GENERIC UACC(READ)
ALTDSD 'SYS2.RTCS.POSZHTML' -
   GENERIC UACC(READ)
ALTDSD 'SYS2.RTCS.POSZPSWD' -
   GENERIC UACC(READ)
ALTDSD 'SYS2.RTCS.REGISTRY' -
   GENERIC UACC(NONE)
/* Define an STC UserID for use by RTCS Initiator AND RTCS Subsystem. */
/* Take care to customize this properly for your RACF configuration. */
/* In particular, take care to properly specify the OMVS parameters. */
Establishing ESM user IDs for RTCS address spaces

Chapter 3 Configuring RTCS

Figure 7  RACF example (part 3 of 4)

ADDUSER OSZRTCS -
  NAME('RTCS Subsystem') -
  NOPASSWORD -
  DFLTGRP(SYS1) -
  UACC(READ) -
  LANGUAGE(PRIMARY(ENU)) -
  OMVS( UID(10010) HOME('/home') PROGRAM('/bin/sh') )

/* Define an STC UserID for use by RTCS Generalized Server (products).* /
/* Take care to customize this properly for your RACF configuration. */
/* In particular, take care to properly specify the OMVS parameters. */
ADDUSER OSZEXEC -
  NAME('RTCS General Server') -
  NOPASSWORD -
  DFLTGRP(SYS1) -
  UACC(READ) -
  LANGUAGE(PRIMARY(ENU)) -
  OMVS( UID(10011) HOME('/home') PROGRAM('/bin/sh') )

/* Connect these two new STC UserIDs to the STC PROCs used by them. */
SETROPTS GENERIC(STARTED) /* We assume this is in effect already. */
RDEFINE STARTED OSZINIT.* STDATA( USER(OSZRTCS) )
RDEFINE STARTED OSZRTCS.* STDATA( USER(OSZRTCS) )
RDEFINE STARTED OSZEXEC.* STDATA( USER(OSZEXEC) )
SETROPTS RACLIST(STARTED) REFRESH

/* Allow READ access by the RTCS Subsystem to RTCS Subsystem Library, */
/* the RTCS Product Library, and Hypertext Document Library PDSEs. */
PERMIT 'SYS2.RTCS.POSZRTCS' -
  GENERIC ID(OSZRTCS) ACCESS(READ)
PERMIT 'SYS2.RTCS.POSZLINK' -
  GENERIC ID(OSZRTCS) ACCESS(READ)
PERMIT 'SYS2.RTCS.POSZHTML' -
  GENERIC ID(OSZRTCS) ACCESS(READ)

/* Allow READ access by the RTCS Subsystem to the */
/* Product Authorization (License) Table Library. */
PERMIT 'SYS2.RTCS.POSZPSWD' -
  GENERIC ID(OSZRTCS) ACCESS(READ)

/* Allow UPDATE access by RTCS Subsystem to the RTCS System Registry. */
PERMIT 'SYS2.RTCS.REGISTRY' -
  GENERIC ID(OSZRTCS) ACCESS(UPDATE)

/* Allow READ access by RTCS Client Product Generalized Server STCs */
/* to the RTCS Product Library and Hypertext Document Library PDSEs. */
PERMIT 'SYS2.RTCS.POSZLINK' -
  GENERIC ID(OSZEXEC) ACCESS(READ)
PERMIT 'SYS2.RTCS.POSZHTML' -
  GENERIC ID(OSZEXEC) ACCESS(READ)

/* Allow SYSLOG access by RTCS Client Product Generalized Server STCs */
Figure 7  RACF example (part 4 of 4)

```
RDEFINE JESSPOOL -
  &RACLNE._+MASTER+.SYSLOG.*.*.? UACC(READ)
PERMIT &RACLNE._+MASTER+.SYSLOG.*.*.? -
  CLASS(JESSPOOL) ID(OSZEXEC) ACCESS(READ)
/* Identify applications defined by RTCS and RTCS client products     */
/* and allow access by all users in general.                          */
RDEFINE APPL (RTCSHTTP) UACC(READ)
RDEFINE APPL (EXPLORER) UACC(READ)
/* Define profiles for modules defined and loaded by Dynamic LPA ADD */
/* Permit RTCS Initiator and RTCS Subsystem to add OSZ modules to LPA */
SETROPTS GENERIC(FACILITY)
RDEFINE FACILITY CSVVDYLPA.ADD.OSZ* UACC(NONE)
PERMIT CSVVDYLPA.ADD.OSZ* -
  CLASS(FACILITY) ID(OSZRTCS) ACCESS(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
/* Allow access to certain TCP/IP resources by the RTCS             */
/* Subsystem and the RTCS HTTP Server which runs in the              */
/* RTCS Generalized Server address space.                           */
SETROPTS CLASSACT(SERVAUTH)
RDEFINE SERVAUTH EZB.STACKACCESS.* UACC(READ)
RDEFINE SERVAUTH EZB.PORTACCESS.* UACC(READ)
RDEFINE SERVAUTH EZB.NETACCESS.* UACC(READ)
RDEFINE SERVAUTH EZB.TN3270.* UACC(READ)
PERMIT EZB.STACKACCESS.* -
  CLASS(SERVAUTH) ID(OSZRTCS) ACCESS(READ)
PERMIT EZB.PORTACCESS.* -
  CLASS(SERVAUTH) ID(OSZRTCS) ACCESS(READ)
PERMIT EZB.NETACCESS.* -
  CLASS(SERVAUTH) ID(OSZRTCS) ACCESS(READ)
PERMIT EZB.TN3270.* -
  CLASS(SERVAUTH) ID(OSZRTCS) ACCESS(READ)
PERMIT EZB.STACKACCESS.* -
  CLASS(SERVAUTH) ID(OSZEXEC) ACCESS(READ)
PERMIT EZB.PORTACCESS.* -
  CLASS(SERVAUTH) ID(OSZEXEC) ACCESS(READ)
PERMIT EZB.NETACCESS.* -
  CLASS(SERVAUTH) ID(OSZEXEC) ACCESS(READ)
PERMIT EZB.TN3270.* -
  CLASS(SERVAUTH) ID(OSZEXEC) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

Figure 8  CA ACF2 example  (part 1 of 4)

```
ACF
* These commands are intended as EXAMPLES ONLY, to be edited by the
* customer system programmer or security administrator for each MVS
* image or shared ESM data base, as required by your configuration.
```
Establishing ESM user IDs for RTCS address spaces

Chapter 3 Configuring RTCS

Figure 8  CA ACF2 example (part 2 of 4)

* The commands provided may not work at your site exactly as they have been illustrated here, or they may not even be necessary.

* Define a started task LOGONID for use by the RTCS Initiator.
* Take care to customize this properly for your ACF2 configuration.
* In particular, take care to properly specify the OMVS parameters.

```
SET LID
INSERT USING(ACFSTCID) OSZINIT -
STC NOJOB NOTSO -
NAME(RTCS Initiator) -
RESTRICT -
GROUP(SYS1)

SET PROFILE(USER) DIV(OMVS)
INSERT OSZINIT UID(07827) HOME(//) OMVSPGM(/BIN/SH)
SET PROFILE(GROUP) DIV(OMVS)
INSERT OSZINIT GID(07827)
```

* Define a started task LOGONID for use by the RTCS Subsystem.
* Take care to customize this properly for your ACF2 configuration.
* In particular, take care to properly specify the OMVS parameters.

```
SET LID
INSERT USING(ACFSTCID) OSZRTCS -
STC NOJOB NOTSO -
NAME(RTCS Subsystem) -
RESTRICT -
GROUP(SYS1)

SET PROFILE(USER) DIV(OMVS)
INSERT OSZRTCS UID(17827) HOME(//) OMVSPGM(/BIN/SH)
SET PROFILE(GROUP) DIV(OMVS)
INSERT OSZRTCS GID(17827)
```

* Define an STC ACID for use by RTCS Generalized Server (products).
* Take care to customize this properly for your ACF2 configuration.
* In particular, take care to properly specify the OMVS parameters.

```
SET LID
INSERT USING(ACFSTCID) OSZEXEC -
STC NOJOB NOTSO -
NAME(RTCS General Server) -
MUSASS -
RESTRICT -
GROUP(SYS1)

SET PROFILE(USER) DIV(OMVS)
INSERT OSZEXEC UID(27827) HOME(//) OMVSPGM(/BIN/SH)
SET PROFILE(GROUP) DIV(OMVS)
INSERT OSZEXEC GID(27827)
```

* The following command needs to be entered at the operator console:
* `F ACF2,REBUILD(USR),CLASS(P)`

* The following items document potential additions to existing data set rules that may be necessary for RTCS and RTCS-based products.
Establishing ESM user IDs for RTCS address spaces

Figure 8 CA ACF2 example (part 3 of 4)

- The RTCS Initiator and the RTCS Subsystem need READ access to the RTCS Subsystem Library: SYS2.RTCS.POSZRTCS
- SYS2.RTCS.POSZRTCS UID(OSZINIT) R(A)
- SYS2.RTCS.POSZRTCS UID(OSZRTCS) R(A)

- The RTCS Subsystem needs READ access to the RTCS Product (License) Authorization Table Library: SYS2.RTCS.POSZPSWD
- SYS2.RTCS.POSZPSWD UID(OSZINIT) R(A)
- SYS2.RTCS.POSZPSWD UID(OSZRTCS) R(A)

- All three RTCS Started Task LOGONIDs need READ access to the RTCS Product Library: SYS2.RTCS.POSZLINK
- SYS2.RTCS.POSZLINK UID(OSZINIT) R(A)
- SYS2.RTCS.POSZLINK UID(OSZRTCS) R(A)
- SYS2.RTCS.POSZLINK UID(OSZEXEC) R(A)

- All three RTCS Started Task LOGONIDs need READ access to the Hypertext Doc. Library: SYS2.RTCS.POSZHTML
- SYS2.RTCS.POSZHTML UID(OSZINIT) R(A)
- SYS2.RTCS.POSZHTML UID(OSZRTCS) R(A)
- SYS2.RTCS.POSZHTML UID(OSZEXEC) R(A)

- The RTCS Initiator and the RTCS Subsystem need ALLOC access to the RTCS System Registry: SYS2.RTCS.REGISTRY
- SYS2.RTCS.REGISTRY UID(OSZINIT) R(A) W(A) A(A)
- SYS2.RTCS.REGISTRY UID(OSZRTCS) R(A) W(A) A(A)

* The RTCS Client Product Generalized Server Started Task LOGONID needs READ access to the standard JESSPOOL SYSLOG SYSOUT data set:
* sysname.-.SYSLOG UID(OSZEXEC) SERVICE(READ) ALLOW

* Identify APPLlications defined by RTCS and RTCS client products. Allow access by all users in general. NOTE: By default, ACF2 maps SAF Resource CLASS 'APPL' to ACF2 Resource Rule TYPE(SAF).
* SET RESOURCE(SAF)
* COMPIL * LIST STORE
* $KEY(RTCSHTTP) TYPE(SAF)
* UID(-) ALLOW
* END
* COMPIL * LIST STORE
* $KEY(EXPLORER) TYPE(SAF)
* UID(-) ALLOW
* END
Establishing ESM user IDs for RTCS address spaces

Figure 8 CA ACF2 example (part 4 of 4)

* Permit RTCS Initiator and RTCS Subsystem to add OSZ modules to LPA
  * using Dynamic LPA facility. NOTE: The following assumes that ACF2
  * maps SAF Resource CLASS 'FACILITY' to ACF2 Resource Rule TYPE(FAC).
  *
  * SET RESOURCE(FAC)
  * COMPILE * LIST STORE
  * $KEY(CSVDYLPA) TYPE(FAC)
  * ADD.OSZ- UID(OSZRTCS) ALLOW
  * END

* Allow access to certain TCP/IP resources by the RTCS
  * Subsystem and the RTCS HTTP Server which runs in the
  * RTCS Generalized Server address space.
  *
  * SET RESOURCE(SER)
  * COMPILE * LIST STORE
  * $KEY(EZB) TYPE(SER)
  * NETACCESS.- UID(OSZRTCS) SERVICE(READ) ALLOW
  * NETACCESS.- UID(OSZEXEC) SERVICE(READ) ALLOW
  * STACKACCESS.- UID(OSZRTCS) SERVICE(READ) ALLOW
  * STACKACCESS.- UID(OSZEXEC) SERVICE(READ) ALLOW
  * PORTACCESS.- UID(OSZRTCS) SERVICE(READ) ALLOW
  * PORTACCESS.- UID(OSZEXEC) SERVICE(READ) ALLOW
  * TN3270.- UID(OSZRTCS) SERVICE(READ) ALLOW
  * TN3270.- UID(OSZEXEC) SERVICE(READ) ALLOW
  * END

* In order to use the SERVAUTH SAF resource CLASS, which is
  * assumed to be mapped to ACF2 rule type SER in the example
  * above, the rule set for rule type 'SER' must be resident.
  *
  * SET CONTROL(GSO)
  * INSERT CLASMAP.SERVAUTH RESOURCE(SERVAUTH) +
  * RSRCYPE(SER) +
  * ENTITYLN(64)
  * CHANGE INFODIR REP TYPES(R-RSER)

* The following commands need to be entered at the operator console:
  *
  * F ACF2,REFRESH(GSO)
  * F ACF2,REBUILD(SER)

END

Figure 9 CA Top Secret example (part 1 of 4)

/* These commands are intended as EXAMPLES ONLY, to be edited by the */
/* customer system programmer or security administrator for each MVS */
/* image or shared ESM data base, as required by your configuration. */

/* The commands provided may not work at your site exactly as they */
/* have been illustrated here, or they may not even be necessary. */
/* Select an available Top Secret Facility Matrix entry for use by    */
/* the RTCS Subsystem and Generalized Server address spaces. It is    */
/* renamed to FACILITY(RTCS), and required attributes established.    */
/* In this EXAMPLE, we illustrate that the USER69 Facility Matrix    */
/* entry is not already in use and has been selected for renaming.    */
/* Note that a Facility Matrix entry you may already have defined    */
/* (such as BBI3) for the MAINVIEW CAS and PASs can also be used.    */
TSS MODIFY(FACILITY(USER69=NAME=RTCS))
TSS MODIFY(FACILITY(RTCS=PGM=OSZ))
TSS MODIFY(FACILITY(RTCS=KEY=0))
TSS MODIFY(FACILITY(RTCS=MULTIUSER))
TSS MODIFY(FACILITY(RTCS=NOABEND))
TSS MODIFY(FACILITY(RTCS=ACTIVE))
TSS MODIFY(FACILITY(RTCS=SIGN(M)))
TSS MODIFY(FACILITY(RTCS=NONPWR))
TSS MODIFY(FACILITY(RTCS=NOSOC))
TSS MODIFY(FACILITY(RTCS=NOPROMPT))
TSS MODIFY(FACILITY(RTCS=RES))
TSS MODIFY(FACILITY(RTCS=MAXUSER=500))
TSS MODIFY(FACILITY(RTCS=ASUBM))
TSS MODIFY(FACILITY(RTCS=MODE=FAIL))
TSS MODIFY(FACILITY(RTCS=WARNPW))
TSS MODIFY(FACILITY(RTCS=DORMPW))
TSS MODIFY(FACILITY(RTCS=NORNDPW))
TSS MODIFY(FACILITY(RTCS=SHRPRF))
TSS MODIFY(FACILITY(RTCS=LUMSG))
TSS MODIFY(FACILITY(RTCS=STMSG))
TSS MODIFY(FACILITY(RTCS=TRACE))
TSS MODIFY(FACILITY(RTCS=LCFCMD))
/* This FACILITY Matrix Entry definition should be put in TSSPARMS    */
/* to ensure that it is always defined, but MODIFY works to define    */
/* the entry temporarily so that TSS does not have to be REINITed.    */
TSS CREATE(OSZRTCS) +
   NAME('RTCS Subsystem') +
   FACILITY(STC) +
   TYPE(USER) +
   PASSWORD(NOPW,0) +
   MASTFAC(RTCS) +
   DEPT(STCPROCS)
TSS ADDTO(OSZRTCS) +
   UID(17827) HOME(/) +
   OMVSPGM(/BIN/SH) +
   DFLTGRP(OMVSGRP) +
   GROUP(OMVSGRP)
/* Define an STC ACID for use by RTCS Initiator and RTCS Subsystem. */
/* Take care to customize this properly for your TSS configuration. */
/* Define an STC ACID for use by RTCS Generalized Server (products).*/
/* Take care to customize this properly for your TSS configuration. */
Establishing ESM user IDs for RTCS address spaces

TSS CREATE(OSZEXEC) +
  NAME('RTCS General Server') +
  FACILITY(STC) +
  TYPE(USER) +
  PASSWORD(NOPW,0) +
  MASTFAC(RTCS) +
  DEPT(STCPROCS)
TSS ADDTO(OSZEXEC) +
  UID(27827) HOME(/) +
  OMVSPGM(/BIN/SH) +
  DFLTGRP(OMVSGRP) +
  GROUP(OMVSGRP)

/* Connect these two new STC UserIDs to the STC PROCs used by them. */
TSS ADDTO(STC) PROCNAME(OSZINIT) ACID(OSZRTCS)
TSS ADDTO(STC) PROCNAME(OSZRTCS) ACID(OSZRTCS)
TSS ADDTO(STC) PROCNAME(OSZEXEC) ACID(OSZEXEC)

/* Allow READ access by the RTCS Subsystem to RTCS Subsystem Library. */
TSS PERMIT(OSZRTCS) DSN(SYS2.RTCS.POSZRTCS) ACC(READ)
TSS PERMIT(OSZRTCS) DSN(SYS2.RTCS.POSZLINK) ACC(READ)
TSS PERMIT(OSZRTCS) DSN(SYS2.RTCS.POSZHTML) ACC(READ)

/* Allow READ access by the RTCS Subsystem to the */
/* Product Authorization (License) Table Library. */
TSS PERMIT(OSZRTCS) DSN(SYS2.RTCS.POSZPSWD) ACC(READ)

/* Allow READ access by the RTCS Subsystem to the */
/* Product Authorization (License) Table Library. */
TSS PERMIT(OSZRTCS) DSN(SYS2.RTCS.POSZPSWD) ACC(READ)

/* Allow ALL access by RTCS Subsystem to the RTCS System Registry. */
TSS PERMIT(OSZRTCS) DSN(SYS2.RTCS.REGISTRY) ACC(ALL)

/* Allow READ access by RTCS Client Product Generalized Server STCs */
/* to the RTCS Product Library and Hypertext Document Library PDSEs. */
TSS PERMIT(OSZEXEC) DSN(SYS2.RTCS.POSZLINK) ACC(READ)
TSS PERMIT(OSZEXEC) DSN(SYS2.RTCS.POSZHTML) ACC(READ)

/* Allow SYSLOG access by RTCS Client Product Generalized Server STCs */
TSS PERMIT(OSZEXEC) JESSPOOL(sysname.+MASTER+.SYSLOG) ACC(READ)

/* Establish minimal access for the PDSE libraries & System Registry. */
/* We set the minimum access needed by a typical non-strict customer. */
TSS PERMIT(ALL) DSN(SYS2.RTCS.POSZRTCS) ACC(NONE)
TSS PERMIT(ALL) DSN(SYS2.RTCS.POSZLINK) ACC(READ)
TSS PERMIT(ALL) DSN(SYS2.RTCS.POSZHTML) ACC(READ)
TSS PERMIT(ALL) DSN(SYS2.RTCS.POSZPSWD) ACC(READ)
TSS PERMIT(ALL) DSN(SYS2.RTCS.REGISTRY) ACC(NONE)

/* Identify applications defined by RTCS and RTCS client products */
/* and allow access by all users in general. */
TSS ADDTO(OSZRTCS) APPL(RTCSHTTP)
TSS ADDTO(OSZEXEC) APPL(EXPLORER)
TSS PERMIT(ALL) APPL(RTCSHTTP)
TSS PERMIT(ALL) APPL(EXPLORER)
Starting RTCS

Start RTCS by issuing the following MVS operator command:

```
S OSZINIT,SUB=MSTR
```

This command starts the RTCS Initiator, which internally starts the RTCS subsystem. The OSZINIT value is the name of the RTCS Initiator started task PROC that you selected in “Creating and customizing the RTCS Initiator started task PROC” on page 50.

**NOTE**

Ensure that the RTCS Initiator address space and the RTCS subsystem address space start successfully. The RTCS subsystem must be successfully initialized before you can proceed with the next configuration step.

Adding the RTCS start command to SYS1.PARMLIB

Determine the appropriate COMMNDxx member of SYS1.PARMLIB (or the logical PARMLIB data set for the z/OS image), and modify the member to include one of the following lines:

```
COM='S OSZINIT,SUB=MSTR'

COM='S OSZINIT'
```
In the commands, the OSZINIT value represents the name of the RTCS Initiator started task PROC.

---

**NOTE**

The RTCS subsystem should usually be started before the primary JES but after the ESM has initialized.

At some sites, this will not be possible because the ESM is started after or under the primary JES. In such cases, the RTCS subsystem should be started as soon as possible, even if it is to be started under the Master Subsystem (SUB=MSTR). If this is still not possible (because the RTCS subsystem needs to be started before the primary JES and run under the Master Subsystem so that it will be available for product address spaces that also start before the primary JES and which are also started under the Master Subsystem), after the ESM has completely initialized on the z/OS image, issue the following operator command:

F RTCS,REFRESH,SECURITY
RTCS administration

This chapter contains information about administrative functions that you might need to perform at your site to ensure continued availability and proper functioning of the RTCS subsystem.

Backing up the RTCS system registry

Because the registry is a read/write data set that is updated during normal system and product operation, consider making a backup of the volume on which the registry resides.

You usually cannot make a backup of the VSAM LDS using data-set-oriented utilities while the RTCS subsystem has it allocated and open. You can, however, unallocate the registry temporarily by using the `REGISTRY UNALLOCATE` command (see “REGISTRY subcommand [registry] [operand]” on page 73). You can then use any data set-level VSAM cluster backup utility to make a backup of the registry VLDS.

If you do not regularly back up the DASD volume on which the registry resides and you make any site-specific customizations to the parameters that are held in the registry, retain the original input that you used to allocate the LDS and all of the subsequent customizations in case you need to reallocate and reinitialize the LDS in the event of a failure.

To make a backup copy of the registry

1. Issue the following command:

   `F RTCS,REGISTRY UNEXPOSE`

2. Issue the following command:

   `F RTCS,REGISTRY UNALLOC`
Recovering the RTCS system registry

3 Run your backup job.

4 After the back up is completed, issue the following command:

   F RTCS,REGISTRY ACQUIRE

Recovering the RTCS system registry

You can re-create the RTCS and product configuration information in the registry by executing the RTCS Registry Maintenance Utility (RMU) and using the INCLUDE statement to reapply your original customizations.

If the DASD volume on which the registry VSAM LDS for an z/OS image was allocated fails, use one of the following methods to recover or rebuild the VSAM LDS contents:

- If a DASD volume-level backup is not available, allocate another VSAM LDS to back the registry. Reallocation or acquire the registry VLDS by using the REGISTRY operator command (see “REGISTRY subcommand [registry] [operand]” on page 73). If necessary, reinitialize the registry LDS by using the RTCS RMU, providing the original customized input.

- If a DASD volume-level or data set-level backup is available, use the backup to restore the registry VSAM LDS. Reallocation or acquire the registry VLDS by using the REGISTRY operator command (see “REGISTRY subcommand [registry] [operand]” on page 73).

If the VSAM LDS that is used to back the registry on an z/OS image or its catalog becomes damaged or is unavailable for some reason when you IPL the z/OS image and the RTCS initiator is started, the RTCS subsystem will not initialize. As a result, products cannot execute. If this situation occurs, use one of the following methods to resolve the problem:

- Correct the problem and start RTCS.

- Restore the VSAM LDS and start RTCS.

- Allocate a new VSAM LDS, start RTCS, and initialize the new registry using any existing, retained customizations before starting any products.
Starting RTCS

RTCS should be started as soon as possible during the IPL of a z/OS image. RTCS should be started either prior to or concurrently with the primary JES. RTCS does not use any JES services.

The RTCS subsystem must be available for products to execute. If the RTCS kernel has not been loaded and initialized, products that use RTCS services usually terminate abnormally (abend) during their initialization process. Products that use the RTCS Generalized Server (program OSZEXEC[n]) to obtain control in the proper execution key abend immediately, even before any product code is loaded. This action alerts you that the RTCS facilities are not available for the involved z/OS image.

To ensure that products will function properly, BMC recommends starting RTCS after the external security manager (ESM) has been started and successfully initialized. If you have a valid reason to start RTCS prior to ESM initialization, ensure that the ESM or your automation facility issues the following command after the ESM has completed initialization:

F RTCS,REFRESH,SECURITY

In the preceding command, RTCS represents the SSID of the RTCS subsystem.

Start RTCS by using the z/OS START command to initiate the RTCS Initiator address space. The actual START command depends on the member name of the RTCS Initiator started task PROC (usually located in SYS1.PROCLIB, but can be in a JES-managed started task PROCLIB) that was selected during z/OS image configuration.

For example, if the member name OSZINIT is selected and the OSZINIT PROC JCL has been properly customized to specify the correct POSZRTCS DSNAME to be used by RTCS, one of the following z/OS START commands is used to start the RTCS Initiator address space:

S OSZINIT,SUB=MSTR

S OSZINIT

The RTCS Initiator performs the following tasks before internally starting the RTCS subsystem address space:

■ reads and processes the RTCS Initialization member from the Logical PARMLIB data set

■ determines and validates the DSNAMEs of the production data sets to be used by the RTCS subsystem and the Generalized Server address spaces
A series of messages is issued to SYSLOG. Figure 10 provides an example of a subset of the messages.

**Figure 10  RTCS subsystem messages**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSZ0050I</td>
<td>RTCS ALLOCATOR VER 01.02.00 PTF BP00076 INITIALIZATION</td>
</tr>
<tr>
<td>OSZ0069I</td>
<td>//TASKLIB DD DISP=SHR,DSN=SYS1.RTCS.POSZRTCS</td>
</tr>
<tr>
<td>OSZ0001I</td>
<td>RTCS SUBSYSTEM INITIALIZATION IN PROGRESS</td>
</tr>
<tr>
<td>OSZ0005I</td>
<td>RTCS SUBSYSTEM PC SERVICES NOW AVAILABLE</td>
</tr>
<tr>
<td>OSZ0029I</td>
<td>OBJECT MANAGEMENT SERVICES NOW AVAILABLE</td>
</tr>
<tr>
<td>OSZ0143I</td>
<td>NONBACKED DATA SPACE ACCESSED USING DSP BY RTCS REGISTRY 246</td>
</tr>
<tr>
<td>OSZ0001I</td>
<td>RTCS SUBSYSTEM MEMORY REGISTRY IS NOW AVAILABLE</td>
</tr>
<tr>
<td>OSZ0069I</td>
<td>//SYS00001 DD DISP=OLD,DSN=SYS1.RTCS.REGISTRY</td>
</tr>
<tr>
<td>OSZ0149I</td>
<td>SYSSUBS DEF. ALET=01FF001A DIV SERVICES ATTACHED TCB=006E4C58 251</td>
</tr>
<tr>
<td>OSZ0022I</td>
<td>RTCS SUBSYSTEM MEMORY REGISTRY IS NOW AVAILABLE</td>
</tr>
<tr>
<td>OSZ0432I</td>
<td>SECURITY ESMTYPE(RACF) SELECTED BY ESMTYPE(AUTO)</td>
</tr>
<tr>
<td>OSZ0443I</td>
<td>RTCS SECURITY - SAFSUBSYS(&amp;SSID ) APPLID(&amp;PRODUCT)</td>
</tr>
<tr>
<td>OSZ0444I</td>
<td>UNDEFINEDUSERINHERIT(ACCEPT) UNDEFINEDUSERSIGNON(REJECT)</td>
</tr>
<tr>
<td>OSZ0445I</td>
<td>GROUPINHERIT(ALWAYS) DEFAULTUSERID(&lt;BLANKS&gt;)</td>
</tr>
<tr>
<td>OSZ0446I</td>
<td>SECTRACE(NONE ) TESTMODE(0A)</td>
</tr>
<tr>
<td>OSZ0447I</td>
<td>ESM(RACF) RCVT ID(RCVT) VRMN(7730) FLGS(033D0800000000FFF080)</td>
</tr>
<tr>
<td>OSZ1083I</td>
<td>RTCS LICENSE MANAGER ACTIVE</td>
</tr>
<tr>
<td>OSZ0039I</td>
<td>CONSOLE COMMAND INTERFACE AVAILABLE FOR OSZRTC RxSCB=RTCS</td>
</tr>
<tr>
<td>OSZ0030I</td>
<td>RTCS SUBSYSTEM (RTCS) IS NOW AVAILABLE ASID=001D OSZA=13175000</td>
</tr>
<tr>
<td>OSZ0030I</td>
<td>ASVT=025DE40</td>
</tr>
</tbody>
</table>

**Stopping RTCS**

RTCS provides non-stop services to client applications, which requires that all critical control blocks be anchored in the RTCS address space. If the RTCS subsystem is taken down, its services will not be available and tasks or address spaces using RTCS will abnormally terminate.

BMC does not recommend that the RTCS subsystem address space be shut down for any reason. Nevertheless, installations have found it desirable to shut RTCS down in order to release the dynamic allocation on the system registry VSAM linear data set (LDS) so that it could be backed up or reallocated, or to change the DSNAMES of the OSZLINK or OSZRTCS production libraries defined in the OSZINIT started task PROC.
To stop RTCS when no client products are running

If no client product address spaces are running, use the standard STOP command to stop RTCS.

stoP cscbName

cscbName is the name of the Command Scheduling Control Block (CSCB) that was built by z/OS. For the RTCS subsystem address space, the CSCB name is the RTCS z/OS Subsystem ID (SSID). For detailed information about the cscbName variable, see “RTCS operator commands” on page 70.

To stop RTCS when client products are running

If any client product address space is using RTCS services, the standard STOP command will not stop the RTCS subsystem address space. The STOP command first checks for any active client product address spaces, and if any are found, they are listed by using message OSZ106E and the STOP command is ignored.

Take the following steps to stop RTCS when client products are running:

1. Stop all MainView address spaces, including PAS, CAS, and TAS (Alternate Access) address spaces.

   If a MainView address space fails to shutdown, see the product’s documentation for additional information.

   **NOTE**
   
   Failure to stop all of these address spaces prior to stopping RTCS can result in additional problems.

2. Have all MainView TSO users logoff.

   If you cannot determine which TSO users have used MainView, stop all TSO users.

3. Issue a STOP command to RTCS, as described in “To stop RTCS when no client products are running.”
If RTCS does not respond with further OSZ0106E messages and does not terminate in a timely manner, you might have to use the FORCE command to stop RTCS.

**NOTE**

See the warning associated with using the FORCE command in the z/OS V1R12.0 MVS System Commands book.

If a FORCE is deemed necessary issue the following commands (for detailed information about the `cscbName` variable, see “RTCS operator commands.”):

```
C cscbName
FORCE cscbName,ARM
```

RTCS terminates.

---

**RTCS operator commands**

The RTCS subsystem address space and Generalized Server address spaces support a number of operator commands. Each command is entered by using the z/OS MODIFY command as follows:

```
F cscbName,command,operand1[...],operandn
```

**cscbName parameter**

In the modify command, `cscbName` is the name of the Command Scheduling Control Block (CSCB) that was built by z/OS for the address space to which the command is being directed. The name is derived as follows:

- For the RTCS subsystem address space, the CSCB name is the RTCS z/OS subsystem ID (SSID).
- For any Generalized Server address space started by RTCS by using the RTCS START,`productIdentifier` command, the CSCB name is the three-character product identifier.
- For a batch job, the CSCB name is the job name.
■ For any other started task that was not started by RTCS, the CSCB name is determined by z/OS and is usually one of the following:

— the name that is specified after the period following the started task PROC name (if any)

— the same as the PROC name

Figure 11 provides a sample subset of the operator messages that are issued when the Generalized Server begins execution. You can use these messages to determine the CSCB name for the address space. For example, in Figure 11, the CSCB name of the Generalized Server address space is ZSE, as shown in bold.

**Figure 11  Generalized Server messages**

```
F RTCS,START,ZSE
START OSZEXEC.ZSE,KEY=4,P=ZSE9INIT,C=BMCPROD
OSZ0031I COMMAND WAS PROCESSED
$HASP100 OSZEXEC ON STCINRDR
$HASP373 OSZEXEC STARTED
OSZ0104I GENERALIZED SERVER STARTING 270
   IN OSZEXEC  (STARTED TASK)  ASID=X'0026'  CSCB=ZSE
OSZ0124I GENERALIZED SERVER PARAMETERS 271
   PACKAGE=ZSE9INIT COMMAND=(NONE)  CONTEXT=BMCPROD
OSZ0039I CONSOLE COMMAND INTERFACE AVAILABLE FOR OSZEXEC  CSCB=ZSE
```

**command parameter**

The following are the valid RTCS subsystem address space and Generalized Server address space operator commands.

**REFRESH,KERNEL**

Valid context: RTCS subsystem only

This command directs RTCS to determine whether an updated RTCS kernel package exists in the RTCS subsystem program library (.POSZRTCS) that was specified when the RTCS Initiator address space was started. If this is the case, the updated RTCS kernel is loaded into extended CSA to replace the older RTCS kernel.

Use this command to refresh the current kernel, if necessary, after applying maintenance to the kernel package in the production RTCS subsystem program library. If no updated kernel exists, no action is performed.
**REFRESH,LIBRARY**

Valid context: RTCS subsystem only

This command directs RTCS to determine whether updated DLL packages exist in the RTCS subsystem program library (.POSZRTCS) that was specified when the RTCS Initiator address space was started. If this is the case, updated DLL packages are loaded into extended CSA to replace the older DLLs.

Use this command to refresh the current DLL packages, if necessary, after applying maintenance to DLL packages in the RTCS subsystem program library. If no updated DLL packages exist, no action is performed.

**REFRESH,SECURITY**

Valid context: RTCS subsystem only

This command directs the RTCS Security Manager to reinitialize every Global Security Parameter from values that are stored in the RTCS system registry.

This command is normally used only when any global security parameters have been changed by the RMU.

Under normal circumstances, you will not need to issue this command. Under specific circumstances, however, if RTCS must be started before the ESM has completely initialized, this command may need to be issued (typically by an automation product) as soon as the ESM has done so.

**SECTRACE,{NONE | ON | SIMPLE | EXTENDED | COMPLETE}**

Valid context: RTCS subsystem only

This command indicates the level of diagnostic trace messages for the RTCS Security Manager component to issue.

NONE indicates that no diagnostic trace messages are to be issued. SIMPLE, EXTENDED, and COMPLETE represent three different levels of message trace. ON is equivalent to COMPLETE.

**WARNMSG,{ON | OFF}**

Valid context: RTCS subsystem or Generalized Server address spaces

This command indicates whether RTCS functions that are executing in the address space should issue the normally-suppressed warning messages.
The default status for every address space is OFF.

**SET field=value**

Valid context: RTCS subsystem only

This command specifies the names of various resources. Table 2 describes the available statements.

**Table 2  SET statements**

<table>
<thead>
<tr>
<th>Statement</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET OSZEXEC=procname</td>
<td>sets the started task PROC name of the Generalized Server address space&lt;br&gt;In the RTCS initialization member, this value was specified by the OSZEXEC-PROC option.</td>
</tr>
<tr>
<td>SET POSZRTCS=dsname</td>
<td>sets the DSNAME of the RTCS subsystem program library&lt;br&gt;If a REFRESH command is subsequently entered, this data set is dynamically allocated to search for updated RTCS program objects.</td>
</tr>
<tr>
<td>SET POSZLINK=dsname</td>
<td>sets the DSNAME of the RTCS product program library&lt;br&gt;The RTCS product program library is the data set that is dynamically allocated by the RTCS Generalized Server in order to load RTCS utility programs.</td>
</tr>
<tr>
<td>SET POSZHTML=dsname</td>
<td>sets the DSNAME of the RTCS hypertext document library&lt;br&gt;This library contains variable-length record members used by RTCS utility programs.</td>
</tr>
<tr>
<td>SET POSZPSWD=dsname</td>
<td>sets the DSNAME of the Product Authorization Table library&lt;br&gt;MainView products do not use this library, so normally you would not need to change this data set name.</td>
</tr>
</tbody>
</table>

**REGISTRY subcommand [registry] [operand]**

Valid context: RTCS subsystem, Generalized Server, or MainView PAS

The **REGISTRY** command performs various functions to assist in the management of an RTCS system registry VSAM linear data set (VLDS).

**subcommand**

The **subcommand** variable indicates the action you want to perform. Table 3 on page 74 describes the valid **subcommand** keywords.
Table 3  Subcommands (part 1 of 2)

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Description</th>
</tr>
</thead>
</table>
| STATUS      | The current status of the RTCS system registry.  
Valid for shared and nonshared RTCS system registries, RTCS memory registries, and RTCS product registries. |
| EXPOSE      | The RTCS system registry is exposed if all of the following criteria are met:  
- the registry is shared within a sysplex  
- the Local Owner is the current system  
- the RTCS system registry is not already exposed  
Exposing the RTCS system registry makes it available to be accessed from other, remote members of the registry's XCF group.  
Valid for shared RTCS system registries and RTCS product registries. |
| UNEXPOSE    | The RTCS system registry is unexposed (hidden) if all of the following criteria are met:  
- the registry is shared within a sysplex  
- the Local Owner is the current system  
- the RTCS system registry is already exposed  
Unexposing the RTCS system registry makes it unavailable for access from other, remote members of the registry's XCF group.  
Valid for shared RTCS system registries and RTCS product registries. |
| UNALLOCATE  | If the RTCS system registry is currently allocated to this XCF member  
- the registry is drained  
- pending updates are hardened in the VLDS  
- the VLDS is dynamically unallocated  
The registry VLDS can be unallocated, say for DASD management or VSAM cluster backup purposes, then reallocated. While the registry VLDS remains unallocated, its DSNAME can be altered.  
Valid for shared and nonshared RTCS system registries and RTCS product registries. |
| TRANSFER operand | Transfers ownership of the RTCS system registry.  
For more information, see “operand parameter” on page 77. |
The `registry` operand indicates the registry you want to take action against.

Use the following syntax to specify a registry:

```
[ SYSTEM | MEMORY ]
```

SYSTEM and MEMORY are only valid when issuing commands to the RTCS address space.

---

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELEASE</td>
<td>If the RTCS system registry is shared within a sysplex and the Local Owner is the current system</td>
</tr>
<tr>
<td></td>
<td>- the registry is drained</td>
</tr>
<tr>
<td></td>
<td>- pending updates are hardened in the VLDS</td>
</tr>
<tr>
<td></td>
<td>- the VLDS is dynamically unallocated</td>
</tr>
<tr>
<td></td>
<td>- other members in the registry’s XCF group that are eligible to become the Local Owner attempt to acquire the registry VLDS and expose it</td>
</tr>
<tr>
<td></td>
<td>Valid for shared RTCS system registries and RTCS product registries.</td>
</tr>
<tr>
<td>REALLOCATE</td>
<td>If the RTCS system registry VLDS is not currently owned by any member of the XCF group, this member dynamically allocates the registry and establishes the registry data space for access on this member only. Other members in the XCF group are not able to access data in the registry until the registry is exposed. Valid for shared and nonshared RTCS system registries and RTCS product registries.</td>
</tr>
<tr>
<td>ACQUIRE</td>
<td>If the RTCS system registry VLDS is not currently owned by any member of the XCF group, this member dynamically allocates the registry, establishes the registry data space, and exposes the registry for access by all members of the registry’s XCF group. Valid for shared RTCS system registries and RTCS product registries.</td>
</tr>
<tr>
<td>ALTER operand</td>
<td>The ALTER subcommand performs various functions based on the operand used with it. Valid for product registries and. For more information, see “operand parameter.”</td>
</tr>
</tbody>
</table>
If the registry operand is not used and the command is issued in
- the RTCS subsystem address space, the default is SYSTEM
- a PAS, the default is the product’s registry

**PACKAGE,\{MAINTenance|STATus\},\{package|group\}**

Valid context: RTCS subsystem, Generalized Server, or MainView PAS

Using **PACKAGE** with the **MAINT** operand requests that the indicated package or group of packages are to be examined and maintenance that has been applied are to be displayed. Elements that have not had any PTFs or ZAPs applied to them will not be listed in the resulting display.

Using **PACKAGE** with the **STAT** operand displays the status of all elements contained in the indicated package or group of packages. Because all SMP/E elements in the package or in all packages in the group will be displayed, the resulting response will contain a very large number of lines.

The **package** variable is the name of an RTCS package currently in storage, or previously loaded into DLPA (packages loaded into DLPA currently are: OSZLPALB OSZKERNL, OSZLIBC, OSZLIBM, and OSZGCC2).

The **group** variable is KERNEL or LIBRARY.

--- **EXAMPLE** ---

Use the following command to display the RTCS maintenance levels:

```
F RTCS, PACKAGE MAINT KERNEL
```

Use the following command to display the level of all RTCS modules:

```
F RTCS, PACKAGE STATUS KERNEL
```

**START,productIdentifier[,J=cscb]**

Valid context: RTCS subsystem only

This command directs RTCS to use the z/OS START command to start a Generalized Server address space.

RTCS constructs the necessary START command and issues it internally. The CSCB name for the Generalized Server is the same as its product identifier, unless you specify another value by using the J operand.
Two of the subcommands have optional operands as described in Table 4.

Table 4 Operands (part 1 of 2)

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Operand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSFER</td>
<td>TO(targetSystem)</td>
<td>If the RTCS system registry is shared within a sysplex and the Local Owner is the current system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ the registry is released by this member</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ the target system is explicitly instructed to acquire the RTCS system registry VLDS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The targetSystem variable is the XCF member name, which can be determined from the output of the REGISTRY STATUS command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid for shared RTCS system registries and RTCS product registries.</td>
</tr>
</tbody>
</table>
The RTCS Security Manager provides internal interfaces for products to take advantage of the facilities that are offered by the three external security managers for z/OS: RACF, CA Top Secret, and CA ACF2. Products use the RTCS Security Manager for all ESM interface functions, and security for the products is configured by using the RTCS RMU. Products might use all or part of the RTCS Security Manager or not use it at all. MainView Infrastructure uses the RTCS Security Manager, but maintains its security configuration separately (that is, not in the RTCS system registry).

### Table 4 Operands (part 2 of 2)

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Operand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER</td>
<td>DSNAME(dsname)</td>
<td>If the RTCS system registry VLDS is in an UNALLOCATED state, a new SREGVLDS DSNAME is establish for the registry VLDS. The new SREGVLDS DSNAME is used when a subsequent ACQUIRE or REALLOCATE REGISTRY command is issued. Valid for shared and nonshared RTCS system registries and RTCS product registries.</td>
</tr>
<tr>
<td></td>
<td>DIV-SAVE-MINIMUM(nnn) DIV-SAVE-MAXIMUM(nnn) DIV-SAVE-IDLE(nnn) DIV-SAVE-LIMIT(nnnn)</td>
<td>These variants of the ALTER keyword set the DIV SERVICES performance parameters. For more information on these specific parameters, see Appendix C, “RTCS initialization member.” The scope of all of these parameters are an individual XCF group member. They can be set independently for each member of the group. The values can be changed at any time (allocated or unallocated). Altering the values on one XCF member does not affect any other member of the XCF group. Valid for shared and nonshared RTCS system registries and RTCS product registries.</td>
</tr>
<tr>
<td></td>
<td>ELIGIBLE-OWNER(YES</td>
<td>NO)</td>
</tr>
</tbody>
</table>
The RTCS Security Manager uses the RTCS system registry to store its configuration structures and any site-specific preferences that have been established during configuration. Two types of configuration structures are supported by the RTCS Security Manager and stored in the RTCS system registry: global security parameters and global class properties.

Each RTCS Security Manager registry structure type consists of two paired, distinct records: BASE and CUST. Each record of each structure type is added to, and updated in, the RTCS system registry by means of the RTCS RMU. Import files in XML format are provided in the TOSZHTML SMP/E target library for each type and each record.

The BASE records of each type are established during the initial RTCS system registry initialization during RTCS configuration for each z/OS image. BASE records contain the default values for each parameter or option that is provided by RTCS. BASE records are never modified by RTCS or customized by the site.

CUST records contain any altered parameters or option values that are customized by the site for that particular RTCS system registry (and the specific z/OS image).

**NOTE**

MainView products maintain their own security configuration and resource definitions independently of RTCS, and do not store this information in the RTCS system registry. This product configuration information is provided to the RTCS Security Manager internally, which uses it to perform the same functions as if the MainView security parameters were stored in the RTCS system registry.

---

**Global security parameters**

The RTCS Security Manager maintains global (rather than product-specific) configuration preferences in the global security parameters (GSP) RTCS system registry structure.

The RTCS Security Manager provides default global security parameters that are suitable for most sites, ESM products, and z/OS configurations. These global security parameters can be customized when necessary.

The RTCS global security parameters structure supports the following parameters and options:

**ESMTYPE**

The ESMTYPE parameter specifies the type of ESM in use on the z/OS image. Supported values include RACF, ACF2, TSS, NONE, and AUTO. The default value is AUTO, which means that RTCS dynamically determines which ESM is active on the z/OS image.
SAFSSUBSYS

The SAFSSUBSYS parameter specifies the SAF SUBSYS name to be passed to the ESM by using RACROUTE requests.

APPLID

The APPLID parameter specifies the default APPL name to be passed to the ESM by using RACROUTE VERIFY and AUTH requests when a client product specification is not provided.

DEFAULTUSERID

The DEFAULTUSERID parameter specifies the default USERID, LOGONID, or ACID to be passed to the ESM when an undefined user ID is specified by a user or is being inherited, and undefined user IDs are indicated as being supported. See also “UNDEFINEDUSERINHERIT” and “UNDEFINEDUSERSIGNON.”

UNDEFINEDUSERINHERIT

The UNDEFINEDUSERINHERIT parameter specifies whether an undefined USERID, LOGONID, or ACID can be inherited on the system. This parameter can have the following values:

■ REJECT indicates that such an attempt is to be rejected.

■ ACCEPT indicates that it is allowed; in this case, the DEFAULTUSERID can be used if the RTCS client product does not provide a product-, context-, or server- (instance-) specific default.

UNDEFINEDUSERSIGNON

The UNDEFINEDUSERSIGNON parameter specifies whether an undefined USERID, LOGONID, or ACID may be signed on to the system. This parameter can have the following values:

■ REJECT indicates that the signon is to be rejected.

■ ACCEPT indicates that the signon is allowed; in this case, the DEFAULTUSERID can be used if the RTCS client product does not provide a product-, context-, or server- (instance-) specific default.
GROUPINHERIT

The GROUPINHERIT parameter specifies the GROUP IDENT inherit scope option. This parameter can have the following values:

- **ALWAYS**, the default value, indicates that when a USERID, LOGONID, or ACID is inherited, the originating security environment’s GROUP IDENT is also inherited.

- **NEVER** indicates that the originating GROUP IDENT is to be ignored, and the GROUP IDENT is allowed to default on the system (that is, as if no GROUP IDENT was specified).

SECTRACE

The SECTRACE parameter specifies the default SECTRACE option, which is the one that is set by the `F RTCS, SECTRACE, option` command. The supported values for this option include: NONE, SIMPLE, EXTENDED, and COMPLETE.

Customizing RTCS global security parameters

You can customize RTCS global security parameters by creating or updating a CUST global security parameter data structure record in the RTCS system registry by using the following steps.

**NOTE**

RTCS global security parameters do not normally need to be customized. RTCS is capable of determining all of the relevant ESM configuration information properly. In addition, MainView products independently maintain their own security parameters and resource definitions and provide this information to the RTCS Security Manager for its use.

1. Execute the RTCS RMU to import the registry IMPORT file (member) that defines the contents of the fields in the global security parameter data structure.

Member OSZRSGSP of the TOSZRXXML SMP/E target library contains a model registry IMPORT file, and member OSZJSGSP of the TOSZCNTL SMP/E target library contains a sample job to import the file. The SYSIN input stream of the OSZJSGSP sample job describes each of the individual options and parameters and their default and supported optional values.
After you make changes to the global security parameters, make the changes effective by entering the following operator command:

\[ F \text{RTCS,REFRESH,SECURITY} \]

In the preceding command, RTCS is the SSID of the RTCS subsystem.

Global class properties

For each supported SAF resource CLASS name, the RTCS Security Manager typically provides a global (that is, common to all products) class property definition, which can be customized for each site, in the global class properties (GCP) RTCS system registry structure. The RTCS Security Manager provides default global class property records for all standard z/OS SAF resource class names that are used in all currently supported products.

The RTCS global class properties structure supports the following parameters and options:

**CLASS**

The CLASS parameter specifies the 8-character SAF resource CLASS name.

**DESC**

The DESC parameter specifies a 64-byte maximum description of the CLASS.

**TSSCLASS**

The TSSCLASS parameter specifies the 8-character internal TSS RDT CLASS name.

**ACTIVE**

The ACTIVE parameter indicates whether the CLASS is active. This parameter can have the following values:

- If the value is YES, authorization requests are processed (passed to the ESM) for the CLASS.
- If the value is NO, authorization requests are not processed and a return code 4 is returned for authorization requests. This return code indicates that the resource manager decides whether to allow resource access. Access is typically allowed, but this is resource CLASS-specific.
MAXLEN

The MAXLEN parameter specifies the maximum resource ENTITY name length. You can specify a value of 1 through 253, CDT, STD, or LONG.

- CDT indicates that the maximum length that is specified in the RACF Class Descriptor Table is to be used.
- STD specifies that the standard TSS RDT length, 8, is to be used.
- LONG indicates that the long TSS RDT length (which is usually 44) is to be used.

LOGOPT

The LOGOPT parameter specifies the default logging option. This parameter can have the following values:

- ASIS, the default value, indicates that the rule, profile, or permit specification is to be used.
- NOLOG indicates that no logging is to be performed.
- NOFAIL indicates that access authorization failures are not to be logged.
- NOAUTH indicates that access authorization successes are not to be logged.

Customizing RTCS global class properties

You can customize RTCS global class properties by creating or updating the Global Class Property data structure CUST record for any given resource class name in the RTCS system registry.

Perform this update by executing the RTCS RMU to import the registry IMPORT file that defines the contents of the fields in the Global Class Property registry data structures.

Member OSZRSGCP of the TOSZRXM SMP/E target library contains a model registry IMPORT file, and member OSZJSGCP of the TOSZCNTL SMP/E target library contains a sample job to import the file. The SYSIN input stream of the OSZJSGCP sample job describes the individual global class properties that may be specified for a SAF resource class name and describes their possible values.
RTCS Registry Maintenance Utility

This chapter provides information about how to use the RTCS Registry Maintenance Utility (RMU).

Introduction

The RMU can be used to import, backup, restore, and manipulate data in an RTCS registry, including the RTCS system registry. Product installation and configuration data are imported into a registry by using an XML document. The RMU program (OSZRGMT) processes XML documents (such as members of the SMP/E TOSZRXXML target library or other product target libraries) and updates the registry data structures.

Data structures that are to be stored in the RTCS subsystem SYSTEM registry can be defined in an RMU IMPORT file XML document. Field values in these structures can be parameterized at RMU execution time by using symbol substitution. A set of product configuration data can be described and shipped with the product and then used to configure each individual system or product instance. The values of these substituted symbols are specified by SET control statements in the RMU control statement (SYSIN) input stream.

The RMU verifies the integrity of the complex data structures against their unique type definition before updating the registry. In addition, substituted data values are validated based on the specific data type that is required within a data structure. This action protects the registry data from corruption at the syntax level.

Data structures in an RTCS registry can be deleted by using a combination of SELECT and DELETE control statements in the RMU SYSIN input stream.

A backup of an RTCS registry’s contents can be taken, and that backup can subsequently be restored to the same or another RTCS registry. The RTCS registry backup and restore capability is useful not only to facilitate recovery from DASD failure, but to enable an existing RTCS registry to be enlarged.
Registry basics

An RTCS registry is a hierarchical set of keys (or nodes) and values used to store data for access by a product from multiple address spaces. A *registry key* is a named container of other keys and values. A *registry value* is a named data element that has a specific data type (basic or complex).

Each registry key is uniquely identified by a *path name*, which is the set of individual key names descending from the *root key*, down to the key that is being examined. This registry path can be written in the same way as a hierarchical file system path. For example, a node named *software* that is defined at the root node is written `/software` and a value named *foo* defined at that node is written `/software/foo`.

A *registry value* can be a complex structured data type that contains multiple fields (each of which can be a basic data type or even another complex data type), or it can be a single, predefined, basic data type (such as an integer or string data element). As a result, binary data structures can be easily stored and directly retrieved for use by programs. All registry keys and most registry value data types can be represented in an XML document.

Executing the RMU

Figure 12 illustrates the execution JCL for the RMU. This sample JCL is contained in member OSZ$REGI in the TOSZCNTL target library.

```plaintext
//jobname JOB (acct,room),programmer NOTIFY=&SYSUID
//IMPORT EXEC PGM=OSZEXEC8,REGION=OM.
//            PARM='P=OSZRMGMT,C=BMCPROD'
//SYSPRINT DD SYSOUT=*  
   (other DD statements as necessary)
//SYSIN DD *       
   (comments and RMU control statements)
/***/
```

The RMU program (OSZRMGMT) is invoked by executing the RTCS Generalized Server (OSZEXEC8). You do not need to specify the DSNAME for the RTCS product program library anywhere in this JCL. The data set will be dynamically allocated by the RTCS Generalized Server, which is loaded into the dynamic LPA (DLPA) area of common storage during RTCS subsystem initialization.
Required DD statements

The RMU requires certain data set allocations in order to function properly. This topic lists the DD statements (and ddnames) for each required allocation and explains their purpose.

SYSIN

The RMU executes by interpreting control statements in the data set that is specified by the SYSIN DD statement. Each record is read and interpreted as either an individual control statement or a comment.

A comment record begins with an asterisk (*) as the first non-blank character. Comments are ignored and can be used for documentation purposes as appropriate for the control statements.

Parameter values that contain embedded blanks can be delimited by using the reverse slash character (\), and a statement can be continued to a subsequent record by using a comma.

For more information about control statement records, see “Control statements” on page 88.

SYSPRINT

The RMU generates standard listing-type output to the data set that is specified by the SYSPRINT DD statement. This output includes informational and error messages that occur during the processing of the RMU control statements that are read from SYSIN.

BACKUP

When a backup operation is performed, the RMU writes a sequential data set that contains a portable form of the registry content by using the specified DD statement. Any ddname can be specified (it does not have to be BACKUP) and the data set can be written to tape or DASD. However, the DCB characteristics of the data set must be:

DSORG=PS,RECFM=VB,LRECL=32756,BLKSIZEx=32760
When a restore operation is performed, the RMU reads the sequential data set that contains the portable form of the registry content by using the ddname specified in the RESTORE control statement. Any ddname can be specified (it does not have to be RESTORE).

Control statements

Control statement records that are supplied in the SYSIN input stream drive all RMU processing. A control statement is one or more records of input that begin with an action such as SET, IMPORT, SELECT, BACKUP, or RESTORE. The action must be the first non-blank characters on the record.

Each control statement is processed completely before the RMU advances to the next statement. If an error is detected in the syntax of a control statement, all processing terminates. An error message is generated to SYSPRINT to indicate the nature of the problem and provide the SYSIN input record number to help you to locate the problem.

Some control statements, such as OPTIONS, SET, and SELECT, do not cause the RMU to take action immediately, but merely specify parameters that will be used on subsequent actions, such as DELETE, BACKUP, RESTORE, or IMPORT.

SET

The SET control statement creates or replaces a name/value mapping that is used by the RMU to:

- substitute elements in registry IMPORT files
- control the behavior of the RMU when subsequent control statements are processed

SET variable = value

NOTE

A BLKSIZE of 32760 is not the most efficient value for a data set written to DASD. This requirement is temporary, and will be removed in a future version of the RMU.
Substitution *variable* names can begin with any alphabetic character (A-Z) or national symbol ($, #, or @ in United States English).

The *value* that is established by this control statement is used for subsequent actions in the SYSIN control statement input stream, such as DELETE, BACKUP, RESTORE, and IMPORT.

To include one or more blank characters in a substitution value, surround the value with reverse slash characters (\). To continue the substitution value to the next record, use a comma as the the last character on the record.

The following SET control statement variables are supported:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGISTRY.DSNAME*a</td>
<td>Data set name of a VSAM linear data set (VLDS) that backs an RTCS registry</td>
</tr>
<tr>
<td>REGISTRY.DDNAME*a</td>
<td>Ddname of a DD statement in the step JCL that allocates a VLDS to back an RTCS registry</td>
</tr>
<tr>
<td>REGISTRY.VLDS*a</td>
<td>One of the following values:</td>
</tr>
<tr>
<td></td>
<td>• SYSTEM – RTCS subsystem SYSTEM registry</td>
</tr>
<tr>
<td></td>
<td>• MEMORY – RTCS subsystem MEMORY registry</td>
</tr>
<tr>
<td>REGISTRY.GROUP</td>
<td>If REGISTRY.DSNAME is specified and REGISTRY.SHARE is set to SHARE (or not specified), identifies the XCF GROUP name with which a sysplex-shared VLDS is associated in the product address space (PAS)</td>
</tr>
<tr>
<td>Note:</td>
<td>When the RMU is used to access a sysplex-shared RTCS registry, you must specify the XCF GROUP name by which it is shared.</td>
</tr>
<tr>
<td>REGISTRY.SHARE</td>
<td>Indicates whether or not the registry identified by REGISTRY.DSNAME is being shared sysplex-wide by using XCF</td>
</tr>
<tr>
<td>Note:</td>
<td>If the registry is being shared sysplex-wide, specify REGISTRY.SHARE = SHARE and specify its XCF GROUP name as the value of REGISTRY.GROUP.</td>
</tr>
<tr>
<td></td>
<td>If the registry is not being shared sysplex-wide, specify REGISTRY.SHARE = NOSHARE and do not specify any value for REGISTRY.GROUP.</td>
</tr>
</tbody>
</table>
The SELECT control statement specifies the path name to which subsequent DELETE statements apply.

```
SELECT TREE="/path"
```

For example:

- To delete the entire subtree starting with the node `/config/sysb`, specify:

```
SELECT TREE="/config/sysb"
DELETE TREE FORCE
```

- To delete the value named ScheduleName stored at the node `/product/current/data`, specify:

```
SELECT TREE="/product/current/data"
DELETE VALUE="ScheduleName"
```
DELETE

The DELETE control statement deletes data from an RTCS registry. The data to be deleted is identified by a SELECT control statement.

DELETE NODE

The DELETE NODE control statement deletes a specific subnode from the node identified by a previous SELECT TREE=\path\ control statement.

DELETE NODE=\subnode\ [OKIFNOTEMPTY]

If OKIFNOTEMPTY is not specified and the specified subnode is empty (does not contain any values or subnodes), the subnode is deleted. Otherwise, an error message is issued and RMU processing is terminated.

If OKIFNOTEMPTY is specified, the specified subnode (and any contained values or subnodes) is deleted even if there are values or subnodes defined there.

NOTE

Another way to delete a subnode and everything defined in it (values and subnodes) is to use the DELETE TREE FORCE control statement.

Figure 13 illustrates the JCL for RMU DELETE NODE processing.

Figure 13  RMU DELETE NODE processing JCL

```
//jobname JOB (acct.room),programmer,NOTIFY=&SYSUID
//IMPORT EXEC PGM=OSZEXEC8,REGION=0M,
//            PARM='P=OSZRGMNT,C=BMCPROD'
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SET REGISTRY.DSNAME = prefix.BMC.product.REGISTRY
SELECT TREE = \config\ DELETE NODE = \sysb\ OKIFNOTEMPTY
/*
```

DELETE VALUE

The DELETE VALUE control statement deletes the specified value from the node identified by a previous SELECT TREE=\path\ control statement.

DELETE VALUE=\valuename\`

Figure 14 on page 92 illustrates the JCL for RMU DELETE VALUE processing.
DELETE TREE FORCE

The DELETE TREE FORCE control statement deletes the node identified by a previous SELECT TREE=\path\ control statement, including all values defined at that node, and all subnodes and values defined within it. The entire registry subtree identified by SELECT TREE is deleted.

**NOTE**

Typically, deleting an entire subtree in a registry is performed only as part of a repair and recovery operation at the request of BMC Customer Support. DELETE TREE FORCE might also be used to erase the entire contents of an RTCS registry in preparation for a RESTORE operation.

Figure 14 illustrates the JCL for RMU DELETE VALUE processing.

Figure 15 illustrates the JCL for RMU DELETE TREE FORCE processing.

BACKUP

The BACKUP control statement writes a portable, sequential backup of the data in an RTCS registry to the data set defined by the specified ddname. The backup data set can be used to restore the data to another (or the same) registry.
Figure 16 illustrates the JCL for RMU BACKUP processing. This sample JCL is contained in member OSZRBACK in the TOSZCNTL target library.

**Figure 16  RMU BACKUP processing JCL**

```
//jobname JOB (acct, room), programmer, NOTIFY=&SYSUID
//IMPORT EXEC PGM=OSZEXEC8, REGION=OM,
//            PARM='P=OSZRGMTN, C=BMCPROD'
//SYSPRINT DD SYSOUT=*
//REGBACK DD DISP=(NEW, CATLG),
//            DSN=prefix.BMC.product.REGISTRY.BACKUP,
//            UNIT=3390, SPACE=(CYL, (25, 10), RLSE),
//            DSORG=PS, RECFM=VB, LRECL=32756, BLSIZE=32760
//SYSIN DD *
SET REGISTRY.DSNAME = prefix.BMC.product.REGISTRY
BACKUP TO=REGBACK
/*

**NOTE**
The DCB characteristics of the backup data set must be:

DSORG=PS,RECFM=VB, LRECL=32756, BLSIZE=32760
```

The RESTORE control statement restores the contents of the sequential backup data set defined by the specified ddname to the currently active registry.

**RESTORE FROM=ddname**

Figure 17 illustrates the JCL for RMU RESTORE processing. This sample JCL is contained in member OSZRRREST in the TOSZCNTL target library.

**Figure 17  RMU RESTORE processing JCL**

```
//jobname JOB (acct, room), programmer, NOTIFY=&SYSUID
//IMPORT EXEC PGM=OSZEXEC8, REGION=OM,
//            PARM='P=OSZRGMTN, C=BMCPROD'
//SYSPRINT DD SYSOUT=*
//REGBACK DD DISP=SHR, DSN=prefix.BMC.product.REGISTRY.BACKUP
//SYSIN DD *
SET REGISTRY.DSNAME = prefix.BMC.product.NEWVLDS
RESTORE FROM=REGBACK
/*
```
The IMPORT control statement reads the XML document in the specified data set (or member) and performs the requested actions in the currently active RTCS registry.

The registry import file can be a sequential data set or a partitioned data set member that is record-oriented. You can specify an explicit data set name or a ddname that has been allocated to the job step.

- To specify a ddname, use the ddname that was allocated in the job step, like this:

  IMPORT  ddname

- To specify a ddname that is allocated to a partitioned data set, include the specific member name, like this:

  IMPORT  ddname(member)

- To specify a specific data set name, prefix the data set name with the characters DSN:, like this:

  IMPORT  DSN:datasetName

- To specify a partitioned data set, include the specific member name, like this:

  IMPORT  DSN:datasetName(member)

Figure 18 on page 95 illustrates the JCL used to IMPORT an XML document into the RTCS system registry. This sample JCL is contained in member OSZINJ91 in the TOSZCNTL target library.
OPTIONS

Use the OPTIONS control statement to adjust the behavior of the RMU for all control statements that follow that OPTIONS control statement.

OPTIONS [[NO]option[,...]]

The option variable value can be TRACE, TRACEX, or PRETEND.

The supported options instruct the RMU to generate additional output to describe the actions that are taken while processing control statements. Each of these options has a corresponding negative option that is preceded by NO—NOTRACE, NOTRACEX, and NOPRETEND; the negative values are the default values. You can specify multiple options on a single OPTIONS control statement by separating each option with a comma.

--- EXAMPLE ---

OPTIONS TRACE,NOPRETEND

TRACE

The TRACE option enables the generation of detailed diagnostic information regarding RMU processing. The diagnostic messages provide a record of what was actually done during RMU processing, such as creating, updating, and verifying keys and values. Each of these messages is generated with informational severity (I) and does not affect any actual RMU processing.

Figure 18  RMU system registry IMPORT processing JCL

```plaintext
//jobname JOB (acct,room),programmer,NOTIFY=&SYSUID
//IMPORT EXEC PGM=OSZEXEC8,REGION=OM,
//       PARM='P=OSZRMNT,C=BMCPROD'
//TOSZXML DD DSN=prefix.BMC.TOSZXML,DISP=SHR
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
SET REGISTRY.VLDS = SYSTEM 
SET ISA.SSID      = RTCS 
SET ISA.POSZHTML  = prefix.BMC.POSZHTML 
SET ISA.POSZLINK  = prefix.BMC.POSZLINK 
SET ISA.TLA       = OSZ 
SET ISA.NAMEKEY   = RTCS 
SET ISA.VENDKEY   = BMC 
SET ISA.PRODUCT   = \Runtime Component System\ 
IMPORT TOSZXML(member) 
//*
```
TRACEX

The TRACEX option enables the generation of detailed diagnostic information regarding the parsing and interpretation of the XML structure of an IMPORT file. This information includes messages regarding the start and end of elements, the attributes processed, and the substitutions that were performed. This information is intended for debugging purposes.

PRETEND

The PRETEND option prevents the RMU from carrying out any updates to any RTCS registry, but otherwise completely processes control statements such as IMPORT. If the TRACE option is enabled, messages are generated that describe the actions that would have been taken against the RTCS registry. The PRETEND option can be used for a trial run of one or more IMPORT files to check for errors before committing the changes.

Return codes and messages

During RMU processing, several different forms of feedback are provided to indicate the success or failure of each invocation. This topic describes the return codes and general messages.

Return codes

When the RMU encounters an error that causes it to terminate without continuing, a return code is generated to indicate the nature of the failure. Each return code is described in Table 5.

Table 5  Registry Import Utility return codes (part 1 of 2)

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>The RMU is unable to open SYSPRINT. Ensure that the SYSPRINT DD statement is specified correctly, and then resubmit the job. SYSPRINT is typically directed to SYSOUT.</td>
</tr>
<tr>
<td>16</td>
<td>The RMU is unable to open SYSIN. Ensure that the SYSIN DD statement is specified correctly, and then resubmit the job.</td>
</tr>
</tbody>
</table>
Messages

During RMU processing, the utility generates a variety of messages. Each message is prefixed with OSZ and ends with one of the following message severity indicators:

- E - Error
- W - Warning
- I - Informational

Most RMU messages are in the range OSZ1500 through OSZ1699x.

If you have not changed the default values for the OPTIONS control statement, the RMU generates a small set of messages for all invocations, including:

- Each control statement, as it is being processed
- Start and end of processing of an XML document by IMPORT
- Completion of the utility program
- Warning and error messages that were generated during processing

When the TRACE option is specified, the RMU generates additional messages detailing the processing of each control statement, as well as the results of each update to an RTCS registry. These additional messages are informational.

When the TRACEX option is specified, the RMU generates additional messages detailing the progress in parsing the elements and attributes of the XML document that is being IMPORTed. These additional messages are informational and are intended for debugging problems with the RMU operation.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>The RMU is unable to open an external data set, such as one that is associated with an IMPORT control statement. Error messages are generated to SYSPRINT to provide more information about which statement specified the failing data set or partitioned data set and member name. If the failing control statement specifies a ddname, check the allocation of that ddname for errors (for example, ensure that the specified member exists).</td>
</tr>
<tr>
<td>8</td>
<td>An error occurred during the processing of either an RMU control statement or an XML document. This return code is accompanied by messages (written to SYSPRINT) that can help diagnose the problem that occurred.</td>
</tr>
</tbody>
</table>
NOTE

A distinct message is always generated by any errors in the XML document input.
Sample JCL and z/OS image configuration members

This appendix provides reference information regarding the RTCS started task PROCs and required z/OS program property table additions. Refer to these examples for assistance in configuring these elements for an z/OS image.

RTCS Initiator started task PROC

Figure 19 is an example of the RTCS Initiator started task PROC. The text in bold represents items that you need to replace with values that are specific to your site; see Table 6 on page 100 for descriptions of these items. Update these statements to reflect the actual DSNAMEs and PROCs to be used. The DSNAMEs are established by the Desktop Installation Assistant, if it is used to configure the target z/OS image. If you copy this PROC for use on another z/OS image, update these DD statements, if necessary. This sample JCL is contained in member OSZ$INIT in the TOSZCNTL target library.

Figure 19  RTCS initiator started task PROC

```
//OSZINIT  PROC
/*
//OSZINIT EXEC PGM=OSZSIRIS,          RTCS Initiator Program
//      TIME=1440,              No SMF CPU or wait time limits
//      REGION=0M             No limits on virtual storage
/*
/* STEPLIB:  RTCS Subsystem Program Library (.TOSZRTCS, or a copy)
/*
//STEPLIB DD DISP=SHR,DSN=SYS2.RTCS.POSZRTCS
```

Table 6 on page 100 provides descriptions of the values that are shown in bold in Figure 19.
Copy this member into one of the following locations:

- SYS1.PROCLIB or another PROCLIB that is available for started tasks that are started by using the Master Subsystem (MSTR)
- a JES-managed started task PROCLIB.

The RTCS Initiator address space should be started as early as possible in the z/OS IPL process after the security system has been initialized. The following examples illustrate the appropriate command (the RTCS Initiator started task PROC member name is shown in **bold**).

```
S OSZINIT
S OSZINIT, SUB=MSTR
```

The RTCS Initiator address space is not the same address space as the RTCS subsystem address space. The RTCS Initiator starts the RTCS subsystem address space internally after performing validation procedures on the following items:

- z/OS system image
- STEPLIB data set (which becomes the POSZRTCS parameter, internally)
- POSZLINK data set
- POSZHTML data set
- SREGVLDS VSAM LDS used to back the RTCS system registry (for dedicated, nonshared RTCS system registry configurations)
- POSZPSWSD product authorization table library
**STEPLIB DD statement**

The RTCS Initiator internally passes the DSNAMEs of the data sets that are used by the RTCS subsystem and the RTCS Generalized Server address spaces in z/OS ECSA storage. The RTCS Initiator obtains the DSNAME of the RTCS subsystem program library (POSZRTCS) from the STEPLIB DD statement of the RTCS Initiator address space started task PROC JCL. The RTCS subsystem address space dynamically allocates this data set, loads the RTCS kernel and DLLs, and deallocates the program library to enable maintenance to be performed by using SMP/E.

The DSNAME of the RTCS subsystem program library (which is specified by the STEPLIB DD statement of the RTCS Initiator started task PROC JCL) is used as the model to determine the default DSNAMEs of the other RTCS data sets used by the RTCS subsystem and the RTCS Generalized Server. The data set prefix (the part up to but not including the low-level index) is determined and the low-level index is examined to see if it is xOSZRTCS, where x is any character. If it is xOSZRTCS, the default DSNAMEs of all of the other data sets will be constructed from the POSZRTCS DSNAME prefix and their own individual, appropriate low-level index, such as xOSZLINK, xOSZHTML, xOSZPSWD, and REGISTRY.

**RTCS subsystem started task PROC**

Figure 20 is an example of the RTCS subsystem started task PROC. The text that should be customized is shown in bold. This sample JCL is contained in member OSZ$RTCS in the TOSZCNTL target library.

**Figure 20  RTCS subsystem started task PROC**

```plaintext
//OSZRTCS PROC
/*
//OSZRTCS EXEC PGM=OSZMOSYS, RTCS Subsystem Initialization
// TIME=1440, No SMF CPU or wait time limits
// REGION=0M No limits on virtual storage
```

Copy this member into one of the following locations:

- SYS1.PROCLIB or another PROCLIB that is available to started tasks that are started under the Master Subsystem (MSTR)
- a JES-managed started task PROCLIB
Specify the member name of this PROC in the OSZRTCS-PROC option of the RTCS Initialization member of Logical PARMLIB. If the name of the PROC used to START the RTCS Initiator address space contains the four characters INIT, the default name of the RTCS subsystem started task PROC will be the same, except that the characters RTCS will be substituted for the characters INIT in that PROC name. If you name the started task PROCs in this manner, you do not need to update the RTCS Initialization member (for this purpose).

The RTCS subsystem control address space is started internally by the RTCS Initiator address space, as early as possible in the z/OS IPL process. If the RTCS Initiator address space (by default, PROC OSZINIT) was started under the Master Subsystem using SUB=MSTR, then, by default, the RTCS subsystem address space will be started under the Master Subsystem also.

RTCS Generalized Server started task PROC

Figure 21 is an example of the RTCS Generalized Server started task PROC. This sample JCL is contained in member OSZ$EXEC in the TOSZCNTL target library.

```
//OSZEXEC PROC KEY=8,P=*,C=*,O=
/*
//OSZEXEC EXEC PGM=OSZEXEC&KEY,     Generalized Server execution key
//              TIME=1440,            No SMF CPU or wait time limits
//              REGION=OM,           No limits on virtual storage
//              PARM=('P=&P',      Product or Package to initialize
//                  'C=&C',      General Server RTCS Context Name
//                  '&O',)      Instance-specific Server options
/*
```

Copy this member into the following locations:

- a JES-managed started task procedure library that is available to started tasks that are started under the primary JES
- SYS1.PROCLIB or another PROCLIB that is available to started tasks that are started under the Master Subsystem (MSTR)

Specify the member name of this PROC in the OSZEXEC-PROC option of the RTCS Initialization member of Logical PARMLIB. If the name of the PROC used to START the RTCS Initiator address space contains the four characters INIT, the default name of the RTCS Generalized Server started task PROC will be the same, except that the characters EXEC will be substituted for the characters INIT in that PROC name. If you name the started task PROCs in this manner, you do not need to update the RTCS Initialization member (for this purpose).
Figure 22 shows entries that must be added to the z/OS program properties table. This sample JCL is contained in member OSZSCHED in the TOSZCNTL target library.

Figure 22  z/OS Program Properties Table additions (part 1 of 2)

```plaintext
/** PRODUCT_IDENTIFICATION ******************************************/
/*                                                                   */
/* PRODUCT_NAME: RTCS             COMPONENT_ID: KERNEL           */
/* PRODUCT_FMID: ZOSZ120        DIAGNOSIS_PIDS: ZOSZ-120        */
/* CURRENT_RMI: BASE              LAST_UPDATE: 11/07/07         */
/*                                                                   */
/** MEMBER_DESCRIPTION ***********************************************/
/*                                                                   */
/*      MEMBER: OSZSCHED                                             */
/*      STATUS: EXTERNAL INSTALLATION MATERIALS                      */
/*    CONTENTS: THIS MEMBER CONTAINS THE PROGRAM PROPERTIES TABLE    */
/*              ENTRIES REQUIRED TO PROPERLY INITIATE THE RUNTIME    */
/*              COMPONENT SYSTEM INITIATOR, ALLOCATOR, AND KERNEL     */
/*              STARTED TASKS, PLUS GENERALIZED SERVER JOB STEPS.    */
/*                                                                   */
/*********************************************************************

PPT  PGMNAME(OSZSIRIS)            /* RTCS SUBSYSTEM INITIATOR STC    */
    KEY(0)                       /* RTCS SUBSYSTEM INITIATOR KEY 0  */
    PRIV                         /* WLM COMPATIBILITY MODE PGN=000  */
    SYST                         /* NON-TIMED STARTED SYSTEM TASK   */

PPT  PGMNAME(OSZMOSYS)            /* RTCS SUBSYSTEM ALLOCATOR STC    */
    KEY(0)                       /* RTCS SUBSYSTEM ALLOCATOR KEY 0  */
    PRIV                         /* WLM COMPATIBILITY MODE PGN=000  */
    NOCANCEL                     /* KERNEL    CANNOT BE CANCELLED   */
    NOSWAP                       /* NON-SWAPPABLE, CROSS-MEMORY A/S */
    SYST                         /* NON-TIMED STARTED SYSTEM TASK   */

PPT  PGMNAME(OSZIRIAN)            /* RTCS SUBSYSTEM KERNEL    STC    */
    KEY(0)                       /* RTCS SUBSYSTEM KERNEL KEY 0    */
    PRIV                         /* WLM COMPATIBILITY MODE PGN=000  */
    NOCANCEL                     /* MUST NOT BE CANCELLED           */
    NOSWAP                       /* NON-SWAPPABLE, CROSS-MEMORY A/S */
    SYST                         /* NON-TIMED STARTED SYSTEM TASK   */

PPT  PGMNAME(OSZEXEC0)            /* RTCS SERVER    STARTED TASK - 0 */
    KEY(0)                       /* PRODUCTS WHICH EXECUTE IN KEY 0 */

PPT  PGMNAME(OSZEXEC1)            /* RTCS SERVER    STARTED TASK - 1 */
    KEY(1)                       /* PRODUCTS WHICH EXECUTE IN KEY 1 */

PPT  PGMNAME(OSZEXEC2)            /* RTCS SERVER    STARTED TASK - 2 */
    KEY(2)                       /* PRODUCTS WHICH EXECUTE IN KEY 2 */

PPT  PGMNAME(OSZEXEC3)            /* RTCS SERVER    STARTED TASK - 3 */
    KEY(3)                       /* PRODUCTS WHICH EXECUTE IN KEY 3 */

PPT  PGMNAME(OSZEXEC4)            /* RTCS SERVER    STARTED TASK - 4 */
    KEY(4)                       /* PRODUCTS WHICH EXECUTE IN KEY 4 */

PPT  PGMNAME(OSZEXEC5)            /* RTCS SERVER    STARTED TASK - 5 */
    KEY(5)                       /* PRODUCTS WHICH EXECUTE IN KEY 5 */
```
### Figure 22  z/OS Program Properties Table additions (part 2 of 2)

<table>
<thead>
<tr>
<th>PPT</th>
<th>PGMNAME(OSZEXEC6)</th>
<th>/* RTCS SERVER STARTED TASK - 6 */</th>
<th>KEY(6)</th>
<th>/* PRODUCTS WHICH EXECUTE IN KEY 6 */</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPT</td>
<td>PGMNAME(OSZEXEC7)</td>
<td>/* RTCS SERVER STARTED TASK - 7 */</td>
<td>KEY(7)</td>
<td>/* PRODUCTS WHICH EXECUTE IN KEY 7 */</td>
</tr>
</tbody>
</table>
This appendix presents the contents of the RTCS initialization member.

**Figure 23  RTCS initialization member  (part 1 of 11)**

<table>
<thead>
<tr>
<th>Runtime Component System (RTCS) Initialization Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSZ$PARM</td>
</tr>
<tr>
<td>OSZ$PARM</td>
</tr>
<tr>
<td>OSZINI00</td>
</tr>
</tbody>
</table>

OSZ$PARM is customized by the installation and then copied into the z/OS image Logical PARMLIB data set (usually DSN=SYS1.PARMLIB). The RTCS Initiator will attempt to locate and read the RTCS Initialization Parameters member from an MVS Logical PARMLIB data set (or SYS1.PARMLIB if no MVS image-specific data set has been established).

The default member name is OSZINI00, which can be changed by specifying the INI=nn parameter in the RTCS Initiator START command parameter field (the 4th positional parameter). For example:

```
START OSZINIT,,,(INI=42),SUB=MSTR
```

---

**== Runtime Component System (RTCS) Initialization Parameters ==**

* MVS Subsystem Name to be used by the RTCS Subsystem.

`* SSID=RTCS`

The MVS Subsystem Name (SSID, or Subsystem ID) that is to be used by the RTCS Subsystem address space (OSZRTCS).

* Installation Verification Procedure (IVP) Mode
Figure 23 RTCS initialization member (part 2 of 11)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| NOIVP     | /* [NO]IVP */
|           | In IVP mode, the RTCS Initiator performs all normal parameter verification and processing but does not START the RTCS Subsystem address space. IVP mode can also be specified in the parameter field (the 4th positional parameter) of the RTCS Initiator START command, as follows: |
|           | START OSZINIT...((IVP=Y, LIST=Y)) |
|           | If IVP mode is requested on the START command, a specification of NOIVP in the Logical PARMLIB member will NOT disable IVP mode. Once IVP mode is in effect (either from the START command or from this member), it cannot then be disabled by specifying NOIVP (in this member). |
| OSZRTCS-PROC=OSZRTCS | RTCS Subsystem address space started task PROC name. |
| OSZEXEC-PROC=OSZEXEC | RTCS Generalized Server started task PROC name. |
| POSZLINK= | RTCS Product Program Library (.TOSZLINK, or a copy) |
|           | RTCS Hypertext Document Library (.TOSZHTML, or a copy) |

* If not specified, the default is the same PROC name that was used to start the RTCS initiator with 'RTCS' substituted for 'INIT', provided that 'INIT' appears in the RTCS Initiator PROC name. Else the default is OSZRTCS. |

* If not specified, the default is the RTCS Subsystem PROC name with 'EXEC' substituted for 'RTCS', provided that 'RTCS' appears in the RTCS Subsystem PROC name. Else the default is OSZEXEC. |

* If not specified, then this value will default to STEPLIB-DSName-prefix.[xosz]LINK, provided that the low-level qualifier of //STEPLIB is '.[xosz]RTCS'. Else the default is the same DSNAME as //STEPLIB.
POSZHTML=

If not specified, then this value will default to STEPLIB-DSName-prefix.[xosz]HTML, provided that the low-level qualifier of //STEPLIB is '.[xosz]RTCS'.

BMC Product (License) Authorization Table Library

POSZPSWD=

If not specified, then this value will default to STEPLIB-DSName-prefix.[xosz]PSWD, provided that the low-level qualifier of //STEPLIB is '.[xosz]RTCS'.

The Product (License) Authorization Table Library is a partitioned data set that contains product license table members that are created & updated by the BMC Licensing Facility batch password processing utility, OSZPATLU, or by the legacy ISPF-based BMC product password update facility.

If no RTCS-based licensed products are being used then this DD statement may define any partitioned data set.

MAINVIEW CAS-based product-only customers should simply allocate an empty, DSORG=PO,RECFM=U PDS[E] for this production library. The indicated data set will by dynamically allocated by the RTCS Initiator to ensure its existence and validity, but no MAINVIEW product will ever cause it to be subsequently dynamically allocated by the RTCS Subsystem or Generalized Server address spaces.

DSNAME of RTCS System Registry VSAM Linear Data Set (VLDS)
+ SREGVLDS=SYS2.SHARED.RTCS.SYSTEM.REGISTRY

If not specified, then this value will default to STEPLIB-DSName-prefix.REGISTRY, provided that the low-level qualifier of //STEPLIB is '.[xosz]RTCS'. Otherwise, there is no default, and this parameter must be specified.

The System Registry contains configuration data for RTCS, the MAINVIEW CAS, RTCS-based products, and any RTCS-dependent product which has elected to use it. It must be a VSAM Linear Data Set (VSAM LDS or VLDS).

The Registry data set MUST be cataloged, since it is a VSAM cluster. It is allocated using only its DSNAME. The System Registry VSAM LDS is read/write and cannot physically be shared, although it may be allocated on shared DASD. Only one RTCS Subsystem will be able to allocate a VSAM LDS for use as a Registry because it will be allocated DISP=OLD as required by the MVS DIV service.

But the data in the System Registry VLDS can be shared among members of a Sysplex using XCF. When the System Registry is being shared among RTCS Subsystems running in a Sysplex, then only one RTCS Subsystem will have dynamically allocated the Registry VLDS. That system is called the Local Owner. Other systems can access data in the Registry (which is allocated to the Local Owner) using XCF to transmit requests and retrieve the requested data in response. An RTCS Subsystem that is accessing data in the System Registry on the Local Owner via XCF is termed a Remote [Registry instance].

It is not recommended, but it is possible to have a private, dedicated System Registry VLDS for each RTCS Subsystem. But the MAINVIEW CAS will not be able to share data with other CASs in the Sysplex if you do.

The first time an RTCS Subsystem becomes the Local Owner of the VLDS and encounters a newly-allocated, uninitialized, never-before-used Linear Data Set, the RTCS Subsystem will initialize the contents of the Registry, populating it with all the structures required for RTCS Subsystem components and products.

* System Registry DIV Services Default Performance Parameters
DIV-SAVE-MINIMUM = 1       /* DIV Services interval 1: minimum */
DIV-SAVE-MAXIMUM = 6       /* DIV Services interval 2: maximum */
DIV-SAVE-IDLE    = 60      /* DIV Services interval 3: idle    */
DIV-SAVE-LIMIT   = 4000    /* DIV Services batch update limit */

The above parameters indicate the time that the Registry DIV Services subtask will wait prior to requesting that changes to the System Registry data space be hardened in the backing VSAM LDS. After the VLDS is updated, it will wait a minimum amount of time before the next request to update the VLDS will again be made, but no longer than the indicated maximum (after which a VLDS update will be forced). If the Registry is idle (not being changed) but potentially only being accessed [existing data retrieved], then the DIV Services subtask will idle for the indicated interval before waking up to check for pending requests. The amount of time that pending Registry VLDS updates are delayed is heuristically determined according to request frequency and arrival pattern, and will never less than the indicated minimum value, nor greater than the maximum. Regardless of the enforced minimum and maximum intervals that will cause the VLDS to be updated, if the number of changes exceeds the specified limit then the backing VLDS will be updated, hardening all pending changes on DASD.

A MINIMUM interval of zero (0) indicates that all changes to the System Registry are to be immediately hardened in the backing VSAM LDS, without waiting or attempting to batch multiple changes together into a single update. We recommend that you do NOT specify DIV-SAVE-MINIMUM = 0.

* * RTCS System Registry Sysplex Sharing Parameters
* -----------------------------------------------
* + REGISTRY-XCF-GROUP = OSZRTCSR       /* System Registry XCF Group Name */
   /* This parameter will be used only */
   /* if some form of Sysplex Registry */
   /* Sharing (see below) is specified */
**ELIGIBLE-OWNER**

/* This member is ELIGIBLE to */
/* ALLOCATE (and then EXPOSE) */
/* the System Registry VLDS. */

/* RTCS Subsystem XCF members which */
/* are not eligible to own the RTCS */
/* System Registry VLDS will not be */
/* able to assist in recovery when */
/* the image that does own the */
/* Registry fails for any reason. */
/* */
/* */
/* RTCS Subsystems on small or */
/* slow MVS images should only */
/* remotely access an exposed */
/* System Registry and should */
/* not be eligible to allocate */
/* and expose/own the Registry. */

* * *

**RTCS System Registry Sysplex Sharing Options**

The following five options are mutually exclusive. Only
one of them should be specified without the "NO" prefix.
The other four may be omitted (or specified with the "NO"
prefix as illustrated below). If more than one positive
[not prefixed with "NO"] option is specified, then the
most restrictive one will become effective. The Registry
sharing options are listed below in that precedence order
(the most restrictive first, the least restrictive last).
In other words, the first (in the order they are listed
below) positive (not prefixed with "NO") option specified
is the one that will be effective and override any others.

**NOPRIVATE-REGISTRY**

/* Exclusively allocate the System */
/* Registry VLDS on this image but */
/* do not establish any capability */
/* to share it with other images. */
/* The Registry cannot subsequently */
/* be exposed to other MVS images. */
/* If the System Registry VLDS can */
/* not be allocated DISP=OLD, then */
/* RTCS initialization will fail. */

**NOALLOC-REGISTRY**

/* ALLOCate the System Registry on */
/* this MVS image, but do not (yet) */
/* EXPOSE it for the REMOTE images */
/* to be able to CONNECT to it. It */
/* can be exposed at a later time */
/* via an RTCS operator command. */
/* If the System Registry VLDS can */
/* not be allocated DISP=OLD, then */
/* RTCS initialization will fail. */
**External Security Manager (ESM) Type**

<table>
<thead>
<tr>
<th>ESMTYPE=</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTO</td>
<td>RTCS is to automatically determine, if possible, which ESM is in use.</td>
</tr>
<tr>
<td>RACF</td>
<td>RTCS is to assume RACF will be used.</td>
</tr>
<tr>
<td>ACF2</td>
<td>RTCS is to assume ACF2 will be used.</td>
</tr>
<tr>
<td>TSS</td>
<td>RTCS is to assume TSS (Top Secret) will be used.</td>
</tr>
</tbody>
</table>
Accept or reject attempts to use undefined ESM User IDs

+ UNDEFINEDUSERINHERIT=ACCEPT

These two parameters specify the behavior when an undefined ESM User ID is provided (either by an end user or an application) as part of original system entry signon, or when attempting to inherit credentials on this image from an existing environment.

The following values can be specified:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPT</td>
<td>RTCS is to allow the INHERIT or the SIGNON to proceed. RTCS or the ESM will provide a default User ID for the security environment that will be created as a consequence.</td>
</tr>
<tr>
<td>REJECT</td>
<td>RTCS is disallow the INHERIT or the SIGNON.</td>
</tr>
</tbody>
</table>

By default, RTCS allows INHERITs to proceed, since it would then be assumed that the User ID being presented was at least valid in the Sysplex or CASplex somewhere, and disallows SIGNONs with an undefined User ID, since it is usually the case that an invalid User ID (that is, one that is not defined) is never to be allowed.

Default ESM User ID used in place of an undefined one

+ DEFAULTUSERID=''

This parameter specifies the ESM User ID that is to be substituted for an undefined/invalid User ID presented for authentication, and the UNDEFINEDUSERINHERIT or the UNDEFINEDUSERSIGNON option, as appropriate, specifies ACCEPT, which indicates that the INHERIT or the SIGNON is to be allowed. The value of this parameter should usually be set to blanks, which triggers ESM-specific behavior to generate its own default User ID (USER, LOGONID, or ACID) according to well-documented ESM behavior or if allowed by ESM-specific security options and parameters.

How to process GROUP IDENT credential during an INHERIT
**GROUPINHERIT=ALWAYS**

This parameter specifies how the GROUP IDENT credential (GROUP name) is to be processed when a security identity is being INHERITed.

The following values can be specified:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALWAYS</td>
<td>RTCS is to pass the GROUP IDENT to the ESM unchanged. In order to be successful, the ESM must allow the use of that specific GROUP IDENT. In the case of RACF, specifically, this means that the User ID being INHERITed must be CONNECTed to the same, exact GROUP IDENT specified in the credentials in the RACF database being used on this system image.</td>
</tr>
<tr>
<td>NEVER</td>
<td>RTCS is to ignore any specification of a GROUP IDENT (GROUP name) in the authentication credentials presented when attempting to INHERIT a User ID.</td>
</tr>
</tbody>
</table>

* ESM Security Interface diagnostic tracing

* SECTRACE=NONE

This parameter specifies the default level of diagnostic tracing that is to be performed for the RTCS External Security Manager interface.

The following values can be specified:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>No ESM interface tracing is to be done.</td>
</tr>
<tr>
<td>SIMPLE</td>
<td>Issue only simple trace messages.</td>
</tr>
<tr>
<td>EXTENDED</td>
<td>Extended tracing is to be performed.</td>
</tr>
<tr>
<td>COMPLETE</td>
<td>Complete tracing is to be performed.</td>
</tr>
</tbody>
</table>

Security diagnostic tracing can be activated dynamically via an RTCS operator command, so NONE should normally be specified. However, if there is a need to perform ESM diagnostic tracing during RTCS initialization, this can be initially activated using this parameter.

* ESM SAF Subsystem (RACROUTE SUBSYS) to be used by RTCS
+ SAFSUBSYS=&SSID

This parameter specifies the RACROUTE SUBSYS (SAF Subsystem ID) that is to be used by the RTCS Security Manager. This value is normally required only when ACF2 is the indigenous ESM. If ACF2 has already been customized (for the MAINVIEW Security Service interface prior to release 6.0) to process RACROUTE invocations using a different SAF SUBSYS, such as BBI3, then that value may still be used, and should be specified here.

The following values can be specified:

<table>
<thead>
<tr>
<th>SAFSUBSYS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;SSID</td>
<td>The MVS SSID that was specified in this initialization member, or was provided as a default by RTCS.</td>
</tr>
<tr>
<td>&amp;PRODUCT</td>
<td>The internal (security) product name in whose product address space an ESM security environment is being created (via SIGNON or INHERIT).</td>
</tr>
<tr>
<td>'' or ' '</td>
<td>Null, or one or more blanks: No SUBSYS= parameter is to be used on any SAF RACROUTE invocation.</td>
</tr>
<tr>
<td>'xxxxxxxx'</td>
<td>A specific RACROUTE SUBSYS= value, consisting of up to 8 characters.</td>
</tr>
</tbody>
</table>

Under most circumstances, RACF and Top Secret will effectively ignore the RACROUTE SUBSYS= specification. However, a value may still be usefully provided here, since the SUBSYS can be used to filter requests to be traced when using the Top Secret SECTRACE facility (as well as the ACF2 SECTRACE facility), and to subset RACROUTE requests to be traced if an IBM or OEM vendor facility (using RACF exits) has been installed to facilitate such tracing. So, a suitable value should be provided, even if RACF or TSS is in use, simply in support of any existing ESM-level diagnostic tracing facility.

* Default ESM RACROUTE system entry validation APPL ID

*
This parameter specifies the RACROUTE APPL (application name) that is to be used as the default SAF APPLication by the RTCS Security Manager (if none is specified by the product or the caller does not have an RTCS product definition, or the definition in the product or System Registry is null or blanks).

The following values can be specified:

<table>
<thead>
<tr>
<th>SAFSUBSYS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;SSID</td>
<td>The MVS SSID that was specified in this initialization member, or was provided as a default by RTCS.</td>
</tr>
<tr>
<td>&amp;PRODUCT</td>
<td>The internal (security) product name in whose product address space an ESM security environment is being created (via SIGNON or INHERIT) or a resource being authorized.</td>
</tr>
<tr>
<td>&amp;PRDAPPL</td>
<td>The default product security APPL name, usually specified in the RTCS product definition in the package, or via the RTCS System Registry product, context, or server (instance) definition.</td>
</tr>
<tr>
<td>'' or ''</td>
<td>Null, or one or more blanks: No APPL= parameter is to be used on any SAF RACROUTE invocation.</td>
</tr>
<tr>
<td>'xxxxxxxx'</td>
<td>A specific RACROUTE APPL= value, consisting of up to 8 characters.</td>
</tr>
</tbody>
</table>
Index

A

ACCEPT
UNDEFINEDUSERINHERIT value 80
UNDEFINEDUSERSIGNON value 80
ACF2
assigning user IDs 53
ESMTYPE value 79
overview of support 78
ACTIVE parameter 82
ALWAYS, GROUPINHERIT value 81
AOSC70D distribution library 28
AOSZCNTL distribution library 28
AOSZHTML distribution library 28
AOSZLINK distribution library 28
AOSZRTCS distribution library 28
AOSZRXML distribution library 28
APF-authorization 45
APPLID parameter 80
ASIS, LOGOPT value 83
AUTO, ESMTYPE value 79

B

BACKUP control statement 92
BACKUP DD statement 87
BASE records 79
batch job, CSCB name 70
BMC Software, contacting 2

C

CA ACF2
assigning user IDs 53
overview of support 78
CA Top Secret
assigning user IDs 53
overview of support 78
CLASS name, SAF resource 82
CLASS name, TSS RDT 82
CLASS parameter 82
Command Scheduling Control Block. See CSCB name commands
MODIFY 70
operator 70
REFRESH,KERNEL 71

commands (continued)
REFRESH,LIBRARY 72
REFRESH,SECURITY 72, 82
SECTRACE 72
START (Generalized Server) 76
START (RTCS) 62, 67
STOP 68
WARNMSG 72, 73
COMMNDxx member 62
common area data spaces, specifying maximum 47
COMPLETE, SECTRACE value 81
components, RTCS 19
configuration
manual 38
modifying 38
overview 37
continuous system operation 17
control statements, RMU
BACKUP 92
DELETE 91
errors 88
IMPORT 94
OPTIONS 95
overview 88
RESTORE 93
SELECT 90
SET 88
conventions, documentation 14
CSCB name
batch job 70
Generalized Server address space 70, 76
Overview 70
RTCS subsystem address space 70
started task 71
CUST records
global class properties 83
global security parameters 81
overview 79
customer support 3
customizing
global class properties 83
global security parameters 81
**D**

Data-In-Virtual (DIV) service 21, 46

DD statements
- **BACKUP** 87
- **IEFPDSI** 50, 51, 52
- **REGISTRY** 101
- **RESTORE** 88
- **STEPLIB** 33, 51, 101
- **SYSIN** 87
- **SYSPRINT** 87
- **TASKLIB** 101

ddnames, RMU
- **BACKUP** 87
- overview 87
- **RESTORE** 88
- specifying in IMPORT control statement 94
- **SYSIN** 87
- **SYSPRINT** 87
- **DEFAULTUSERID** parameter 80
- **DELETE** control statement 91
- **DELETE NODE** control statement 91
- **DELETE TREE FORCE** control statement 92
- **DELETE VALUE** control statement 91
- **DESC** parameter 82
- diagnostic trace messages 72
- distribution libraries 28
- DIV. See Data-In-Virtual (DIV)
- DLL. See dynamic link library (DLL)
- dynamic link library (DLL)
  - date and time stamp 18
  - refreshing packages 18, 72

**E**

electronic documentation 13

ESM. See external security manager (ESM)

ESMTYPE parameter 79

EXTENDED, SECTRACE value 81

external security manager (ESM)
  - specifying type 79
  - starting 67
  - user ID 53

**G**

Generalized Server address spaces
- assigning ESM user IDs 53
- code, executing 21
- CSCB name 70, 76
- messages 71
- multiple 21
- overview 21
- protection keys 21
- sample JCL 34

Generalized Server address spaces (continued)
- starting 76
- warning messages 72

Generalized Server started task
- creating 52
- default value 102
- example 102
- **IEFPDSI DD statement** 52
- location 102
- naming 52
- specifying 102

global class properties
- customizing 83
- overview 82
- parameters 82

global security parameters
- **APPLID** 80
- customizing 81
- **DEFAULTUSERID** 80
- **ESMTYPE** 79
- **GROUPINHERIT** 81
- overview 79
- refreshing 72, 82
- **SAFSUBSYS** 80
- **SECTRACE** 81
- **UNDEFINEDUSERINHERIT** 80
- **UNDEFINEDUSERSIGNON** 80

GROUPINHERIT parameter 81

GRSplex requirement 30

**H**

Help, online 13

HTML documents, library 32

hypertext document library (TOSZHTML)
- allocation 32
- creating 32, 43
- overview 29, 32
- sample JCL 44
- when used 32

**I**

IEASYSxx member 47, 48

IEFPDSI DD statements
- Generalized Server started task 52
- Initiator started task 50
- subsystem started task 51

IEFSSNxx member 49

IMPORT control statement 94

importing system registry files 94

Initiator started task
- assigning ESM user IDs 53
- creating 50
- default values 100
Initiator started task (continued)
  example 99
  location 100
  sample JCL 33
  validation procedures 100
  when to start 100

J
JCL samples
  Generalized Server address space started task 34
  hypertext document library, creating 44
  Initiator started task 33
  library 33
  product program library, creating 42
  RMU backup processing 93
  RMU execution 34, 86
  RMU restore processing 93
  subsystem program library, creating 40
  subsystem started task 34
JES requirement 67
JES-managed started task PROCLIB 52, 53

K
kernel
  date and time stamp 18
  maintenance 18
  overview 19
  refreshing 18, 71
  updating 71
kernel and DLL refresh facility 18
key, registry 86

L
libraries
  APF-authorization 45
  categories 27
  distribution 28
  hypertext document library (TOSZHTML) 32, 43
  illustration 28
  JES-managed started task procedure library 52
  Master Subsystem procedure library 52
  overview 27
  product program library (TOSZLINK) 42
  production 29
  runtime 29
  sample JCL and parameter library (TOSZCNTL) 33
  subsystem program library (TOSZRTCS) 31, 40
  target 29, 31
  XML document library (TOSZXML) 32
LOGOPT parameter 83
LXs 47

M
Master Subsystem procedure library 52
MAXCAD parameter 47
MAXLEN parameter 83
messages
  diagnostic trace 72
  Generalized Server 71
  RMU 97
  subsystem startup 67
  warning 72
MODIFY command 70
MSTJCL00 member
  Generalized Server started task 52
  Initiator started task 50
  subsystem started task 51

N
NEVER, GROUPINHERIT value 81
NOAUTH, LOGOPT value 83
NOFAIL, LOGOPT value 83
NOLOG, LOGOPT value 83
NONE, SECTRACE value 81
NOPRETEND option, RMU 95
NOTRACE option, RMU 95
NOTRACEX option, RMU 95
NSYSLX parameter 47

O
objects, PM3 program 31
online Help 13
operator commands 70
OPTIONS control statements
  NOPRETEND option 95
  NOTRACE option 95
  NOTRACEX option 95
  overview 95
  PRETEND option 96
  TRACE option 95, 97
  TRACEX option 96, 97
OSZEXEC member
  overview 34
  using 52, 53
OSZINIT member
  overview 33
  using 50
OSZREGI member
  overview 34
  using 86
OSZRTCS member
  overview 34
  using 51
OSZEXEC member 31, 34, 52
OSZINIT member 33, 50
OSZJSGCP member 83
OSZJSGSP member 81
OSZMOSYS program 40, 42
OSZRBACK member
  overview 35
  using 93
OSZRGMTN member 85
OSZRRREST member
  overview 35
  using 93
OSZRSSCP member 83
OSZRSSCP member 81
OSZRTCS member 34, 51
OSZSCHED member 35, 48
OZI Customization 38

parameters
  ACTIVE 82
  APPLID 80
  CLASS 82
  DEFAULTUSERID 80
  DESC 82
  ESMTYPE 79
  GROUPINHERIT 81
  LOGOPT 83
  MAXCAD 47
  MAXLEN 83
  SAFSUBSYS 80
  SECTRACE 81
  TSSCLASS 82
  UNDEFINEDUSERINHERIT 80
  UNDEFINEDUSERSIGNON 80

path, registry 86
PDSEs, sharing in sysplex 30
PM3 program objects 31
PRETEND option, RMU 96
primary JES requirement 67
product program library (TOSZLINK)
  APF-authorizing 45
  creating 42
  overview 29, 31
  sample JCL 42
product support 3
production libraries
  applying maintenance 18, 29
  benefits of using 29
  overview 29
  runtime 29
  updating 18
Program Properties Table (PPT)
  entries to add 35, 103
  refreshing 48
  updating existing SCHED member 48

protection keys, in Generalized Server address spaces 21
publications, related 13

R
RACF
  assigning user IDs 53
  ESMTYPE value 79
  overview of support 78
record types
  BASE 79
  CUST 79, 81, 83
REFRESH commands
  KERNEL 71
  LIBRARY 72
  SECURITY 67, 72, 82
refreshing
  DLL packages 18, 72
  global security parameters 72, 82
  kernel 71
  PPT 48
REGISTRY command 73
REGISTRY DD statement 101
registry import file 94
registry key 86
Registry Maintenance Utility (RMU)
  changing behavior 95
  control statements 88
  DD statements, required 87
  diagnostic information 95, 96
  executing 86
  messages 97
  overview 85
  sample execution JCL 34
  testing import 96
registry path 86
registry value 86
REJECT, UNDEFINEDUSERINHERIT value 80
REJECT, UNDEFINEDUSERSIGNON value 80
related publications 13
RESTORE control statement 93
RESTORE DD statement 88
return codes, RMU 96
RMU control statements
  BACKUP 92
  DELETE 91
  errors 88
  IMPORT 94
  OPTIONS 95
  overview 88
  RESTORE 93
  SELECT 90
  SET 88
RMU ddnames
  BACKUP 87
  overview 87
Index 121

RMU ddnames (continued)
   RESTORE 88
   specifying in IMPORT control statement 94
   SYSIN 87
   SYSPRINT 87
RMU. See Registry Maintenance Utility (RMU)
RTCS Allocator (OSZMOSYS) 40, 42
RTCS product registry 21
RTCS Security Manager
   default global security parameters 79
   global class properties default 82
   overview 78
   reinitializing 72
   trace messages 72
RTCS subsystem
   DLL packages, refreshing 72
   kernel, refreshing 71
   started task 51
   warning messages 72
RTCS system registry
   backing up 65
   data accessibility 20
   DSNAME, selecting 46
   end-user profiles 66
   example definitions 22
   global class properties, customizing 83
   global security parameters, customizing 81
   importing files 94
   initialization problems 66
   initializing 72
   overview 20, 86
   recovering 66
   retaining original input 65
   VSAM LDS, backed by 46
RTCS. See Runtime Component System (RTCS)
Runtime Component System (RTCS)
   components 19
   configuration 37
   control address space 19, 49
   kernel 19
   overview 17
   primary JES requirement 67
   product libraries 28
   starting 50, 62, 67
   stopping 68
   runtime libraries 29

S
SAF resource CLASS name 82
SAFSSUBSYS parameter 80
sample JCL and parameter library (TOSZCNTL)
   allocation 33
   members 33, 48
   overview 29, 33
   production copies 30
   sample JCL and parameter library (TOSZCNTL) (continued)
      when used 33
   SCH member suffixes 49
   SCOPE=COMMON data spaces 47
   SECTRACE command 72
   SECTRACE parameter 81
   Security Manager. See RTCS Security Manager
   SELECT control statement 90
   SET command 73
   SET control statement 88
   sharing PDSs in sysplex 30
   SIMPLE, SECTRACE value 81
   SSID, selecting 49
   START command
      Generalized Server address space 76
      RTCS 62, 67
   started tasks
      CSCB name 71
      Generalized Server 52
      Initiator 50
      RTCS subsystem 51
   starting
      Generalized Server address spaces 76
      RTCS 62, 67
   STEPLIB DD statement
      Initiator started task 51, 101
      OSZINIT 33
   stopping RTCS 68
   subsystem address space, RTCS
      allocating 20
      commands 70
      CSCB name 70
      maintenance 18
      operator commands 70
      overview 20
      RTCS system registry VSAM LDS 46
      system registry VSAM LDS 21
   subsystem ID (SSID), selecting 49
   Subsystem Name (SSN) Table, updating 49
   subsystem program library (TOSZRTCS)
      allocation 31
      APF-authorizing 45
      creating 31, 40
      overview 29, 31
      sample JCL 40
      when used 31
   subsystem started task
      creating 51
      example 101
      naming 51
      sample JCL 34
      specifying 102
      when to start 102
   subsystem, RTCS
      assigning ESM user IDs 53
      messages, startup 67

Index 121
subsystem, RTCS (continued)
  starting 62
  updating 18
support, customer 3
syntax statement conventions 14
SYSIN DD statement, RMU 87
SYSIN input streams 85
SYSPRINT DD statement, RMU
  error opening 96
  overview 87
system linkage indexes (LXs) 47

T
target libraries
  list 29
  maintenance, applying 18
  overview 31
TASKLIB DD statement 101
technical support 3
Top Secret
  assigning user IDs 53
  overview of support 78
TOSZCNTL library
  allocation 33
  members 33, 48
  overview 29, 33
  production copies 30
  when used 33
TOSZHTML library
  allocation 32
  creating 32, 43
  overview 29, 32
  sample JCL 44
  when used 32
TOSZLINK library
  allocation 31
  APF-authorizing 45
  creating 42
  overview 29, 31
  sample JCL 42
  when used 31
TOSZRTCS library
  allocation 31
  APF-authorizing 45
  creating 31, 40
  overview 29, 31
  sample JCL 40
  when used 31
TOSZRXML library
  allocation 32
  overview 29, 32
  production copies 30
  when used 32
TRACE option, RMU 95, 97
TRACEX option, RMU 96, 97

TSS
  assigning user IDs 53
  ESMTYPE value 79
  overview of support 78
TSS RDT CLASS name 82
TSSCLASS parameter 82

U
UNDEFINEDUSERINHERIT parameter 80
UNDEFINEDUSERSIGNON parameter 80
updating
  DLL packages 72
  kernel 71
user IDs
  default 80
  ESM 53
  undefined 80
user profile information, recreating 66

V
value, registry 86
VSAM LDS, RTCS system registry
  calculating size 46
  overview 20
  recovering 66
  selecting 46

W
warning messages 72
WARNMSG command 72

X
XML document library (TOSZRXML)
  allocation 32
  overview 29, 32
  production copies 30
  when used 32
Notes