Radia Client Automation Enterprise Configuration Server

For the Windows® operating systems

Software Version: 9.00

Reference Guide

Document Release Date: April 2013

Software Release Date: June 2013

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notice

Trademark Notices

Acknowledgements

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

This product includes software written by Tim Hudson (tjh@cryptsoft.com).

This product includes software written by Daniel Stenberg (daniel@haxx.se).

This product includes OVAL language maintained by The MITRE Corporation (oval@mitre.org).

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

http://support.persistentsys.com/

This site requires that you register for a Persistent Passport and sign in. Register online at the above address.

For more details, contact your Persistent sales representative.

Support

Persistent Software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by being able to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Submit enhancement requests online
- Download software patches
- Look up Persistent support contacts
- Enter into discussions with other software customers
- Research and register for software training

To access the Self-solve knowledge base, visit the Persistent Support home page.

Note: Most of the support areas require that you register as a Persistent Support user and sign in. Many also require an active support contract. More information about support access levels can be found on the Persistent Support site.

To register for a Persistent Support ID, go to: Persistent Support Registration.

Contents

Chapter 1

Introduction

The Configuration Server is a main component and essential function of Core and Satellite servers. In conjunction with the Configuration Server Database (CSDB), it configures and maintains the desired state for your enterprise devices and agent computers.

The Configuration Server can:

- **Manage a RCA agent's Desired State**
  Dynamically create an RCA agent's **desired state** based on situation-specific data, thereby creating a software environment that automatically adapts to user and device changes.
- **Distribute Configuration Server Database Objects**
  Synchronize distributed objects (such as application components, packages, device configurations, and policy relationships) across the network, and automatically manage object transfer between Radia Client Automation components.
- **Deliver and Maintain RCA Policies**
  Maintain enterprise policies in the CSDB. When a device that is being managed by RCA connects to the Configuration Server all current policies are automatically applied.
- **Issue Connects and Fulfill Requests**
  Contact devices causing them to initiate data requests and desired-state requests. These requests can be according to a schedule or on notification from an administrator.

Configuration Server Database

The CSDB is stored on the Configuration Server, and is accessed and modified via the RCA Administrator CSDB Editor (Admin CSDB Editor). It houses Client Automation products, and is where administrators save and maintain an enterprise's service-entitlement policies.

The CSDB includes the following information:

- The digital assets that are distributed by Radia Client Automation.
- The policies that determine package assignment to managed devices and subscribers.
- Security and access rules for administrators of Radia Client Automation.

Configuration Server Database Documentation

Consult the following Radia Client Automation publications for more information on the structure, use, and components of the CSDB.

- *Radia Client Automation Enterprise Configuration Server Database Reference Guide*
- *Radia Client Automation Enterprise Administrator User Guide*

Abbreviations and Variables

Abbreviations Used in this Guide

Abbreviation	Definition
RCA	Radia Client Automation
Core and Satellite	RCA Enterprise environment consisting of one Core server and one or more Satellite servers.
CSDB	Configuration Server Database

Variables Used in this Guide

Variable	Description	Default Values
`InstallDir`	Location where the RCA server is installed	For a 32-bit OS: `C:\Program Files\Hewlett-Packard\HPCA` For a 64-bit OS: `C:\Program Files(x86)\Hewlett-Packard\HPCA`
`SystemDrive`	Drive label for the drive where the RCA server is installed	C:

Configuration Server Information

Configuration Server Directories

The table below lists the directories that are automatically created during the Configuration Server installation.

Configuration Server Directories

Directory	Contents
`Bin`	The shell scripts that enable you to start, stop, clean up, and query the Configuration Server.
`DB`	The CSDB files.
`Exe`	The Configuration Server methods and executable files.
`internet`	The internet HTML and graphics files.
`lib`	The files for proper Configuration Server operation. Note: Do not modify or delete these files.
`Log`	The Configuration Server log.
`Rexx`	This directory is for storing customized REXX methods. Note: Its sub-folder, `NOVADIGM`, contains the default Configuration Server REXX methods. See the caution that follows.

Note: Altering the Configuration Server methods that are in the `ConfigurationServer\rexx\NOVADIGM` directory could adversely effect RCA processing.
Therefore, it is recommended that a method be copied up one level, to the `ConfigurationServer\rexx` directory, before being modified.

IP Networking Support

With this release, Radia Client Automation adds support for **IPv6**—the latest version of the internet protocol addressing structure—to its Windows-based Core and Satellite servers. The Core and Satellite servers can now use either IP version 4 (**IPv4**) or IP version 6 (**IPv6**) for server-to-server communications. RCA agent communications, however, are currently limited to IPv4. For details, refer to the appendix, *IPv6 Networking Support*, in the *Radia Client Automation Enterprise User Guide*.

Note: Radia Client Automation environments that use the traditional, component-based, RCA server installations will continue to be supported on IPv4 only.

Backing up the Configuration Server

It is recommended that the Configuration Server (and CSDB) be periodically backed up. To facilitate doing so, a list of the Configuration Server directories (that are installed by default) and their applicable platforms and contents are provided in the table below.

Note: It is recommended that backups be done in accordance with each environment’s in-house, corporate protocol.

Configuration Server Directories

Directory	Contents
bin	The Configuration Server binary files and the `edmprof` file.
DB	The Configuration Server Database files. The DB directory is created under <*datadir*>.
lib	This directory contains files for proper Configuration Server operation; do not modify or delete these files.
log	The Configuration Server log.
modules	The `.tkd` files for OS Manager.
rexx	This directory is for storing customized REXX methods. **Note:** Its subfolder, `NOVADIGM`, contains the default Configuration Server REXX methods.

Configuration Server as a Windows Service

During the installation, the Configuration Server was set up to run as a Windows Service. Confirm this by checking the Services area of the Windows machine. As a Windows Service, the Configuration Server can be started, stopped, and queried, as described in this section.

Windows Service Options

If the Configuration Server is set up to run as a Windows Service, it can be started, stopped, and queried, as described in this section.

To start the Configuration Server as a Windows Service:

1. On the RCA Core server, click **Start**>**Control Panel**>**Administrative Tools**>**Services**.
2. Right-click the service RCA Configuration Server and select **Start**.

The Configuration Server will start.

To stop the Configuration Server as a Windows Service:

1. On the RCA Core server, click **Start**>**Control Panel**>**Administrative Tools**>**Services**.
2. Right-click the service RCA Configuration Server and select **Stop**.

It will take approximately two minutes for the Configuration Server to shut down.

Using the Windows Event Viewer with the Configuration Server

If the Configuration Server is running as a Windows Service, the Windows Event Viewer can be used to see key Configuration Server messages, such as start-ups and shutdowns.

The Event Viewer displays three types of messages: error, warning, and informational. There are five Event Viewer messages (two informational and three errors) that work with Configuration Server processing. These are described in "Configuration Server Messages in Event Viewer" on next page.

Accessing the Event Viewer

The Event Viewer is invoked in a variety of ways on the various Windows operating systems. For instructions on how to access the Event Viewer on your operating system, consult the documentation that was distributed with it.

Filtering for Configuration Server Messages

After the Event Viewer is open, locating messages that are specific to the Configuration Server is easy.

To locate Configuration Server messages:

Note: The steps outlined below are applicable to most Windows operating systems, and are accurate as of this writing.

1. Select **Application**.
2. From the Menu bar, click **View** and select **Filter**.
3. From the Event Source drop-down list, select **ZTopTask**, and click **OK**.

Now, the right panel of the Event Viewer window displays only those messages that result from Configuration Server processing. Verify this by checking that **ZTopTask** is listed in all the rows of the Source column.

Configuration Server Messages in Event Viewer

Detailed below are the five Event Viewer messages that are related to the Configuration Server processing.

- Start-up Message
  `Configuration Server HAS STARTED`
  The Configuration Server for Windows is beginning its start-up processing by running the Configuration Server program.
- Error Message 1
  `START: Configuration Server CONTROL DISPATCHER FAILED`
  The Configuration Server for Windows failed during service initialization. Start-up processing has stopped.
- Error Message 2
  `Configuration Server CONTROL HANDLER REGISTRATION FAILED IN SERVICE MAIN`
  During second stage initialization, the Windows service controller failed to update the registry keys. Start-up processing has stopped.
- Error Message 2
  `SET Configuration Server FAILED IN REPORT STATUS`
  An attempt was made to report the status of the Configuration Server for Windows start-up process to the Configuration Server service. However, the Configuration Server service did not receive the status report.
  The Configuration Server service may actually be running, or the start-up service has stopped. The Configuration Server is unable to discern what the actual condition is.
- Shut-down Message
  `Configuration Server STOPPED BY STOP REQUEST OR SHUT-DOWN REQUEST`
  The Configuration Server has shut down.

Chapter 2

Tuning the Configuration Server

The performance of the Configuration Server depends on a number of factors, such as the number of RCA agents being concurrently processed, the complexity of the configurations for those RCA agents, the volume of the data being processed, and network bandwidth. The configuration of the Configuration Server log, which documents system status for informational purposes and problem determination, can also alter performance.

The Configuration Server operational parameters are contained in its `edmprof` file. The performance of the Configuration Server can be tuned by working with the settings of the `edmprof` file.

This chapter details the structure of the `edmprof` file and the options that can be specified.

Configuration Server Settings Overview

The Configuration Server `edmprof` file contains the parameters that determine how the Configuration Server will operate. This file is organized into **sections**—with each section containing keywords, called **settings**. The sections can be categorized into functional areas. Further detail for the settings in each of the sections is provided in this chapter.

- **Identification**
  - MGR_STARTUP specifies the Configuration Server by ID, name, type, and communications port.
  - MGR_DIRECTORIES identifies the directory paths for the Configuration Server Databases, REXX methods, and non-REXX methods.
  - MGR_LICENSE contains your unique license string.
- **Specification**
  Configuration Server settings establish application wide technical parameters.
  - MGR_CACHE contains cache-processing options.
  - MGR_TPINIT identifies communications packet sizes.
- **Initialization**
  - MGR_ATTACH_LIST determines which Configuration Server programs are initiated at startup, and has options to define their functioning.
  - MGR_CLASS specifies which classes and instances of the CSDB are cached during the Configuration Server initialization process.
  - SECTION_DELIMITERS specifies which characters are used to distinguish sections in the `edmprof` file.
- **Operations**
  A number of sections in the Configuration Server `edmprof` file contain parameters for system operations.

- MGR_ACCESS determines Configuration Server access to administrator and console functions.
- MGR_DB_VERIFY specifies the parameters for automatic CSDB verification at initialization.
- MGR_DIAGNOSTIC specifies the timing, size, and logging options for verifying adequate disk space for CSDB operations.
- MGR_DMA specifies parameters for Radia Client Automation Distributed Configuration Server (Distributed Configuration Server).
- MGR_METHODS identifies options for method processing.
- MGR_NOTIFY specifies the Configuration Server defaults for RCA agent notifications.
- MGR_OBJECT_RESOLUTION specifies parameters used in object resolution.
- MGR_POOLS (*Windows only*) allocates available pools of memory (in different sizes) for system tasks.
- MGR_RETRY values define how long a RCA agent is to wait before attempting to reconnect to the Configuration Server.
- MGR_TASK_LIMIT identifies the maximum concurrent Configuration Server tasks, ongoing and deferred, system related and/or RCA agent-connect related.
- MGR_TIMEOUT specifies the amount of time a Configuration Server will wait for an inactive RCA agent.

- **Monitoring**
  - MGR_LOG and MGR_TRACE identify where the Configuration Server log is located, how flexible it is, and which individual system traces are to be captured in the log.
  - MGR_SMTP_MAIL specifies the parameters for using SMTP mail messages to support Configuration Server monitoring.
  - MGR_MESSAGE_CONTROL specifies where log messages are to be sent and if they are to be suppressed.
  - MGR_USERLOG enables an administrator to establish a user logging facility.

Viewing and Editing Configuration Server Settings

The `edmprof` file can be opened in a text editor so that its parameters can be viewed and, if necessary, adjusted. This section describes how to access the `edmprof` file, as well as important information about editing and saving the file.

Accessing the EDMPROF File

The `edmprof` file can be found in varying locations, based on operating system, as described below:

- **Windows**
  The `edmprof` file settings are located in the `edmprof.dat` file, located in the **bin** subdirectory of the Configuration Server directory.

Caution: While the settings in the `edmprof` file are readily accessible and easy to modify, some of the values are critical to the operation of the Configuration Server. Therefore, it is imperative that you *do not*:

- alter any `edmprof` file settings unless without consulting this guide first, and then using the recommended values.
- delete any `edmprof` file settings unless instructed to do so by a member of Persistent Technical Support.

Editing the EDMPROF File

The `edmprof` file can be edited in a standard text-editing application.

Caution: Be sure to review the important information in this section before editing this file. Failure to do so could adversely effect your RCA environment.

It is recommended to back up the `edmprof` file before editing it.

- You must use a UTF8-aware text editor when editing the `edmprof` file.
- Be sure to select `UTF-8` as the *encoding type* when saving the file.
  - Windows users are advised to use **Notepad**.

Failure to use a recommended text-editing application and saving the file as recommended in this section could result in the file's changes not being correctly applied to the Configuration Server. This could adversely effect your RCA environment.

Configuration Server Settings

Most of the sections of the `edmprof` file can be independently configured, whereas some are based on operating system and communications requirements. While many of the sections are optional, a number are required for proper Configuration Server functioning.

Note: Since some of the sections in the `edmprof` file are optional, and others are platform-specific, it's possible that not all of the settings will be visible in every `edmprof` file.

The `edmprof` file is created during the installation of the Configuration Server. Much of its information is derived directly from parameters that are specified during the installation; while others are automatically entered during the installation.

Two types of values are in the `edmprof` file.

- An **as-installed** value represents a manual input, specified during installation or a derived entry for a required setting.
- A **default** value is established by the Configuration Server if there is a blank value for that setting, whether it is required or manually entered.

Note: If the value of a setting/parameter is documented as **N/A**, there is no value of that type for that setting/parameter.
If a value of a setting/parameter has **NONE** (all uppercase) specified, then this is an accepted, optional value for that setting/parameter.

Format of the EDMPROF File

The `edmprof` file is organized into *sections*. Each section contains individual keywords called *settings*. Each setting receives an acceptable *value*, which can be numeric, alphabetic, or alphanumeric.

Caution: Some of the settings and values are critical to the operation of the Configuration Server. Do not alter or delete these unless instructed to do so by a member of Persistent Technical Support.

The following is an example of the format of a section of the `edmprof` file.

Example

```
[MGR_SECTION]
SETTING = VALUE
SETTING = VALUE
SETTING = VALUE
```

The following table presents a list of the `edmprof` file sections, a brief description, and whether the section is required or optional.

Sections of the EDMPROF File

Section Name and Description	Required or Optional
MGR_ACCESS Specifies access to the Administrator and Console functions.	Optional
MGR_ATTACH_LIST Specifies the Configuration Server attach list that defines the programs to be attached when the Configuration Server is started.	Required
MGR_CACHE Specifies cache-processing options, such as cache segments, size, and statistics.	Optional
MGR_CLASS Specifies processing parameters for classes and instances.	Optional
MGR_DB_VERIFY Specifies the extent of automatic database verification.	Optional
MGR_DIAGNOSTIC	Optional

Specifies the parameters for diagnostic Manager (zdiagmgr) tasks.	
MGR_DIRECTORIES Specifies the path for the databases, REXX methods, and non-REXX methods.	Recommended but not required
MGR_DMA Specifies the parameters for Distributed Configuration Server operations.	Optional
MGR_ERROR_CONTROL Specifies how to process errors that are encountered while the Configuration Server is running.	Optional
MGR_LOG Specifies the logging directory and options for the Configuration Server logging facility.	Recommended, but not required
MGR_MESSAGE_CONTROL Specifies where messages are to be sent and whether they are to be suppressed.	Optional
MGR_METHODS Specifies options for method execution.	Recommended, but not required
MGR_NOTIFY Specifies the parameters for notify processing.	Optional
MGR_OBJECT_RESOLUTION Specifies the parameters used in object resolution.	Optional
MGR_POLICY Specifies the IP address/name, port, and details of Policy Resolution process of the Radia Client Automation Policy Server (Policy Server), if it has been enabled.	Optional
MGR_POOLS Specifies the allocation of pool sizes on startup.	Optional
MGR_RESOLUTION_FILTERS Specifies the filtering rules for resolution, based on the connection type.	Optional
MGR_RETRY Specifies when an RCA agent should attempt to reconnect to the Configuration Server, following rejection.	Optional
MGR_RIM Specifies the IP address/name and port of the Inventory Manager, if it has been enabled.	Optional
MGR_ROM Specifies information used for OS management.	Optional
MGR_RMP Specifies the IP address/name and port of the Portal, if it has been enabled.	Optional

MGR_SMTP_MAIL Specifies the parameters the Configuration Server uses to interface with SMTP.	Optional
MGR_SNMP Contains parameters to specify where SNMP traps are to be sent, and controls the behavior of the built-in SNMP agent.	Optional
MGR_SSL Specifies the parameters for the Radia Client Automation Configuration Server SSL Manager task.	Optional
MGR_STARTUP Specifies the Configuration Server ID and TCP/IP port number.	Required
MGR_TASK_LIMIT Specifies the number of concurrent tasks permitted.	Required
MGR_TIMEOUT Specifies how long the Configuration Server will wait for a request from a connected RCA agent before disconnecting it.	Optional
MGR_TPINIT Specifies communications packet sizes.	Required
MGR_TRACE Controls and influences diagnostic logging for the Configuration Server.	Optional
MGR_USERLOG Specifies the logging directory and options for the user logging facility.	Optional
OBJECT_SIZES Specifies the number of heaps and the heap size for CSDB objects that are being created on the Configuration Server as in-storage CSDB objects.	Optional
RCS_TUNING_CONTROL Provides a mechanism to override the default values that are specified in the Configuration Server self-tuning tool.	Optional
SECTION_DELIMITERS Specifies the left and right delimiters that are to be used for enclosing the section names within the Configuration Server `edmprof` file.	Optional

The remainder of this chapter documents each section of the `edmprof` file, covering the individual settings and the values for each. Also described is the effect of tunable settings on Configuration Server performance.

Caution: Before editing this file, be sure to review the important information in the section, "Editing the EDMPROF File" on page 28. Failure to do so could adversely effect your RCA environment.

MGR_ACCESS

This section determines access to the Administrator and Console.

MGR_ACCESS Settings

Setting	Description
ADMIN	Specifies access to the Administrator. The values are DENY, ALLOW, and IGNORE.
CONSOLE	Specifies access to the Console. The values are DENY, ALLOW, and IGNORE.

Example

```
[MGR_ACCESS]
ADMIN = DENY
CONSOLE = DENY
```

MGR_ACCESS Values

Setting	Value as Installed	Default Value
ADMIN	DENY	DENY
CONSOLE	DENY	DENY

Note: Access can be controlled by native operating system security features also.

Performance and Usage Considerations

- ADMIN=ALLOW will provide access to the Administrator without checking the ZACCESS domain of the CSDB. Therefore, if ADMIN=ALLOW, and an Administrator attempts an action for which no access rules have been defined, the attempted action will be permitted.
- When ADMIN=DENY, access rules governing Administrator actions are defined in the ZACCESS domain of the CSDB. Therefore, if ADMIN=DENY, the Administrator will be unable to perform an action unless there is an access rule specifically allowing it.
- If ADMIN=IGNORE the Administrator will be able to perform any action because the access rules will be ignored by the Configuration Server. Setting ADMIN=IGNORE essentially disables all access rules as they relate to Administrator functions.
- Access to the Console is determined by local password security policy. If CONSOLE=ALLOW and local security is not configured, read-only access is granted. If CONSOLE=IGNORE, local Console security is bypassed.

RCA Administrator Access Level Values

Access Value	Definition

ALLOW	This value will result in the Configuration Server *not* checking the administrator access rules before granting access to an administrator to perform CSDB administrator functions. This value disables all administrator access-rule checks, even if they are defined in the database.
DENY	This value will result in the Configuration Server checking administrator access rules before granting access to an administrator to perform CSDB administrator functions. An administrator will be *unable* to perform an action unless there is an access rule defined in the database for that administrator specifically *allowing* it. This is the recommended setting when configuring administrator security.
	Note: If this option is specified, undefined administrators will not have access to any CSDB administrator functions, unless the ADMINID._NULL_INSTANCE_ is modified to allow such access.
IGNORE	This value will result in the Configuration Server checking administrator access rules before granting access to an administrator to perform CSDB administrator functions. An administrator will be *able* to perform an action unless there is an access rule defined in the database for that administrator specifically *prohibiting* it. **Note:** If this option is specified, undefined administrators will have full access to all CSDB administrator functions, unless the ADMINID instance exists for that administrator to prohibit such access.

MGR_ATTACH_LIST

In this section, specify which programs (Configuration Server tasks) are to be attached at startup, and set the options for these processes.

MGR_ATTACH_LIST Settings

Setting	Description
ATTACH_ LIST_ SLOTS	Number of slots for the attach list kept in shared memory of the Configuration Server. Every entry is 132 bytes long. It is recommended that this setting be one more than the number of CMD_LINE settings used. For example, if there are seven CMD_LINE settings, set this value to 8.Note: No process starts are required. However, system ability is limited if ZTCPMGR, ZREXXMGR, and ZNFYTMGR are not attached.
CMD_LINE	Command line to use when starting processes. Blanks are not permitted in CMD_ LINE= substring. A second format is allowed when multiple instances of the same task are required and need to be separately identified. This format is: CMD_ LINE=(NAME=,ADDR=,PORT=). These names should be unique within the Configuration Server.
LIMIT (=CLOSE)	The Configuration Server will drop any RCA agent connection attempts when LIMIT=CLOSE is specified, *and* either the Task Limit or Storage Limit of the Configuration Server is reached. The Configuration Server will not return an object to the RCA agent (no return EDMLOCTP), nor will it request the RCA agent retry the connection.

	Note: This setting is valid with ztcpmgr only.
RESTART	Use this setting to determine if a Configuration Server task will be restarted when abnormally terminated. Blanks are not allowed in RESTART= substring. The default is **NO**.
RESTART_ LIMIT	Number of attempts to restart an attached process that has terminated.
VERIFY_ INTERVAL	Interval (in minutes) between verifications that attached processes are still running. If this value is 0, no verification will occur. The default is **1** (minute).

The following table lists the Configuration Server tasks.

Configuration Server Tasks

Task Name	Description
zbldpmgr	Patch Build Manager task
zdiagmgr	Diagnostic Manager task
znfytmgr	TCP Notify Manager task
zrexxmgr	REXX Manager task
zrtrymgr	Notify Retry Manager task
zsmtrmgr	SMTP receive Manager task
zsmtsmgr	SMTP send Manager task
zsnmpmgr	SNMP Manager task
zsslmgr	SSL Manager task
ztcpmgr	TCP Manager task
zutilmgr	Utility Manager task

Example

```
[MGR_ATTACH_LIST]
```

`ATTACH_LIST_SLOTS`	`= 15`
`RESTART_LIMIT`	`= 7`
`VERIFY_INTERVAL`	`= 5`
`CMD_LINE`	`= (zutilmgr) RESTART=YES`
`CMD_LINE`	`= (zrexxmgr) RESTART=YES`

`CMD_LINE`	`= (zsnmpmgr) RESTART=YES`
`CMD_LINE`	`= (zsmtrmgr) RESTART=YES`
`CMD_LINE`	`= (zsmtsmgr) RESTART=YES`
`CMD_LINE`	`= (znfytmgr) RESTART=YES`
`CMD_LINE`	`= (ztcpmgr) RESTART=YES`
`CMD_LINE`	`= (ztcpmgr LIMIT=CLOSE) RESTART=YES`
`CMD_LINE`	`= (zbldpmgr)`

Table 10 MGR_ATTACH_LIST Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
ATTACH_ LIST_SLOTS	15	15	Number of CMD_LINEs	Number of CMD_LINEs + 1
CMD_LINE	Determined by installation options	Determined by installation options	N/A	N/A
RESTART	YES	NO	N/A	N/A
RESTART_ LIMIT	7	7	0 = No restart	3200

Performance and Usage Considerations

- If the ATTACH_LIST_SLOTS value is too low, the Configuration Server will attach as many tasks as there are slots available. Any remaining tasks will not be attached until a slot becomes vacant. A too-high value could degrade overall system performance by unnecessarily setting aside resources that remain unused.
- The RESTART_LIMIT value should be set higher when critical Configuration Server functions are being performed. Note that regardless of the value in RESTART_LIMIT, the task will not be reinitiated if RESTART=NO.
- To ensure that vital processes continue running, set VERIFY_INTERVAL lower when critical Configuration Server functions are being performed. A higher setting, on the other hand, might save CPU cycles when total demand is a critical factor.
- The ztcpmgr task supports virtual IP addresses. It accepts the IP address and port number on the command line, as in:
  ```
  CMD_LINE = (ztcpmgr addr=1.1.1.94,port=4438)
  ```
  If the address is not specified, the machine address is used.

MGR_CACHE

This section specifies cache-processing options, such as cache size, statistics, load type, and error response.

Note: The settings in this section should be established based on operating system environment and performance needs.

MGR_CACHE Settings

Setting	Description
AVERAGE_ OBJECT_ SIZE	Average size of an object that will be cached.
CACHE_ SEGMENTS	Number of cache segments.
CACHE_ SIZE	Size of each cache segment.
CACHE_ STATS	A YES/NO switch to accumulate statistics.
ICACHE_ COUNT_ ERROR	Instructs the Configuration Server on what action to take (shut down or log a warning) if the ICACHE instance counts don't match the DCS instance counts when it (the Configuration Server) is starting up. The default is **SHUTDOWN**. **Note:** The default is SHUTDOWN because running the Configuration Server without the correct number of cached items it is not recommended. SHUTDOWN directs the Configuration Server program (ZTOPTASK) to shut down. WARN instructs the Configuration Server program to continue loading and record the event in the Configuration Server log.
ICACHE_ LOAD_ TYPE	The DOMAIN.CLASS entries that are specified in the MGR_CLASS section are loaded, one class at a time, into Index cache via one of the following methods. **AUTOMATIC** instructs the Configuration Server to auto-determine the loader method (FULL or CHUNKY) to be used. This is the default. **FULL** instructs the Configuration Server to always use the FULL loader method—loading the database as a single entity. **CHUNKY** instructs the Configuration Server to always use the CHUNKY loader method—loading each CSDB service one at a time. For more information, see "ICACHE_LOAD_TYPE Considerations" on page 38.
ICACHE_ SIZE	Size of the Index cache.

Example

```
[MGR_CACHE]
```

`AVERAGE_OBJECT_SIZE`	`= 2048`
`CACHE_SEGMENTS`	`= 2`
`CACHE_SIZE`	`= 5242880`
`CACHE_STATS`	`= NO`
`ICACHE_COUNT_ERROR`	`= SHUTDOWN`
`ICACHE_LOAD_TYPE`	`= AUTOMATIC`
`ICACHE_SIZE`	`= 0`

MGR_CACHE Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
AVERAGE_ OBJECT_SIZE	2048 Bytes	2048	2048	6144
CACHE_ SEGMENTS	2	0 (No caching)	0 (No caching)	System resource dependent
CACHE_SIZE	5,242,880 Bytes	0	0	System resource dependent
CACHE_STATS	NO	NO	N/A	N/A
ICACHE_ COUNT_ERROR	SHUTDOWN	SHUTDOWN	N/A	N/A
ICACHE_ LOAD_TYPE	AUTOMATIC	AUTOMATIC	N/A	N/A
ICACHE_SIZE	N/A	N/A	0 (No Index caching)	N/A
OBJECTID_ VERDB_INITIAL_ ENTRIES	N/A	N/A	Approximat-ely 1.2 times the total number of instances in the database	N/A

Performance and Usage Considerations

Note: When modifying any of the cache parameters in this section, take care not to exceed the amount of virtual memory that is available.

Also, sufficient virtual memory must be available to handle the maximum concurrent resolutions workload.

- Classes that are to be cached are defined in the MGR_CLASS section.
  At a minimum, the SYSTEM.PROCESS and SYSTEM.ZMETHOD classes should be cached.
- If the value of AVERAGE_OBJECT_SIZE is too low, the Configuration Server will have to reconfigure the cache until the object is accommodated.
  If the value of AVERAGE_OBJECT_SIZE is too high, the caching function might not be used efficiently.
- The ICACHE_SIZE setting should be used in conjunction with the CACHE_SEGMENTS setting and the "MGR_CLASS" on next page section.
  ICACHE_SIZE will be enabled only if the value of CACHE_SEGMENTS is greater than zero.
- If ICACHE_COUNT_ERROR=SHUTDOWN and the Configuration Server shuts down, areas to check are:
  - Check the Configuration Server’s log file,
  - Check the integrity of the connection to the database,
  - Verify the load type for the database connection method, and
  - Verify the validity of the database.

ICACHE_LOAD_TYPE Considerations

This section presents information that should be considered when specifying a value for ICACHE_ LOAD_TYPE.

- If ICACHE_LOAD_TYPE=FULL, the Configuration Server will attempt to load into ICACHE, as a single element, all of the CSDB classes that are specified in the "MGR_CLASS" on next page section.
  The Configuration Server sequentially accesses all instances in the specified directories.
- If ICACHE_LOAD_TYPE=CHUNKY, the Configuration Server will attempt to load all of the CSDB classes that are specified in the "MGR_CLASS" on next page section—but as individual entities.
  - **Component Classes**
    The Configuration Server uses the PACKAGE class in the current domain to control the loading of Component class instances—that is, all instances that are associated with a PACKAGE instance will get loaded into ICACHE, but orphan instances (those not associated with a PACKAGE instance) will not get loaded into ICACHE.
  - **Non-Component Classes**
    The Configuration Server uses a systematic approach to load the instances of instances of non-Component classes.

    Note: It is recommended to use the CHUNKY method for large databases because it minimizes the load on the network, thereby decreasing the likelihood of problems and the possibility of the database integrity being compromised.

- If a Configuration Server has a UNC connection (database path starts with \\) to the CSDB and the DMA count exceeds 300K instances, ICACHE_LOAD_TYPE=AUTOMATIC will choose the CHUNKY load method. See "UNC Connectivity Issues" on page 45.

Purging Dynamic Cache

This section provides precautionary information about purging the dynamic cache on a Configuration Server. It includes information about protection that Radia has introduced in order avoid purging the CSDB when a Radia Client Automation Proxy Server (Proxy Server) is co-located with the Configuration Server and dynamic cache is enabled.

Caution: It is recommended not to use dynamic cache for a co-located Proxy Server.

If the dynamic cache root is a CSDB then, by default, this parameter (default=0) prevents automatic dynamic cache purging of aged files when the dynamic cache index is saved. By default, the new parameter automatically safeguards against purging dynamic cache files from a CSDB.

To remove the safeguard and permit a purge of shared resource, dynamic cache files, set the parameter to 1, as shown in the following example.

Example

```
-dynamic -allow -shared -resource -purge 1
```

MGR_CLASS

This section specifies which classes and instances will be cached during the initialization of the Configuration Server.

MGR_CLASS Settings

Setting	Description
CLASS	Specifies which classes and instances will be cached at initialization.
	Format: DOMAIN.CLASS={VALUE1,VALUE2,VALUE3,VALUE4} If multiple domains have identical class names, the class templates must be identical. If they are not, performance will be adversely effected and objects, larger than necessary, might be created from the resolution process. *Cache_Class_Template_&_Base_Instance*, *Cache_All_Instances*, *Heap_Size*, *Maximum_Number_of_Heaps*.
	VALUE1 To cache (at Configuration Server startup) the _BASE_INSTANCE_ and class template of the associated class, specify `Y`. Otherwise, specify `N`.
	VALUE2 To cache (at Configuration Server startup) all instances in the associated class, specify `Y`. Otherwise, specify `N`.

	VALUE3 This value is numeric and represents the initial size of the resolved object for the associated class. This is the size that each heap in the associated DOMAIN.CLASS will occupy in persistent objects. It should include any attributes that have been classed into the in-storage object as the result of resolution. Typically, the size of the transient class instance can be used for transient objects (such as, ZLOCMGR, ZLOCCLNT, ZSCHEDULE). The default (**2048 bytes**) can be refined for the ZSERVICE and PACKAGE classes, which are typically 3–4 KB in size. Examine the RCA agent object resulting from a representative resolution to determine the most appropriate value. Values specified are generally in 512-byte increments.
	VALUE4 This is a numeric value that indicates an estimate of the number of heaps that are required for resolution. As each RCA agent begins resolution, memory is allocated in blocks equal to the sum of all of the products of VALUE3*VALUE4. When memory is exhausted for a persistent class that is being cached, the next increment of storage is obtained in a block determined by the product of VALUE3*VALUE4 for the associated class. For example, if **SOFTWARE.FILE = Y,Y,2048,100** was specified, the class template and _BASE_INSTANCE_ are cached at startup. All of the instances of the SOFTWARE.FILE class are cached in memory (if there is sufficient room in CACHE_SEGMENTS*CACHE_SIZE for the whole class). The working value for the size of each FILE instance after resolution is estimated to be 2048 bytes. An initial allocation for the in-storage FILE object is made for 100 heaps that will require 100*2048 bytes (20 KB). If the resolution mode for the average RCA agent requires 1000 FILE instances, then one initial allocation of 20 KB and nine subsequent allocations of 20 KB would be required to satisfy the entire 1000 FILE instance in-storage objects.

Note: At startup, classes are cached in the order they appear if VALUE1=Y and VALUE2=Y.

Example

```
[MGR_CLASS]
```

`SYSTEM.PROCESS`	`= Y,Y,2048,1`
`SYSTEM.ZMETHOD`	`= Y,Y,2048,1`
`SOFTWARE.FILE`	`= Y,N,4096,1`
`SOFTWARE.REGISTRY`	`= Y,Y,4096,1`
`SOFTWARE.DESKTOP`	`= Y,Y,2048,1`
`SOFTWARE.ZSERVICE`	`= Y,Y,3072,1`

MGR_CLASS Values

Setting	Value as Installed	Default Value
CLASS	N/A	N/A

Performance and Usage Considerations

- You can also specify your own unique classes in this section.
- Note that class instance size and number of instances specified in MGR_CLASS have an effect on storage and performance. A class instance size larger than the actual class size represents wasted storage. A class instance size smaller than the actual class size results in continual resolution performance degradation.
- Only the classes listed will be cached. All instances of listed classes are icached, if required, regardless of the values of the first two parameters in the list.
- You can use the following wildcard characters when you specify the DOMAIN and CLASS:

Wildcard Character	Behavior
&	Matches any string
!	Excludes whatever follows it

The ampersand (&) character is used instead of the asterisk (*), because the asterisk is used for commenting in the `edmprof.dat` file.

For example:

DOMAIN&.CLASS	!DOMAIN&.CLASS
DOMAIN.&	!DOMAIN.&
&.CLASS	!&.CLASS
DOMAIN&.&	!DOMAIN&.&

The following limitations apply:

a. You can only use one & in the DOMAIN name and one & in the CLASS name.

b. The maximum number of entries in MGR_CLASS is 256 (not counting those excluded with !).

c. The maximum number of exclusions is 256.

MGR_DB_VERIFY

This section establishes the level of CSDB verification and whether errors are automatically corrected.

Note: If this function is configured, the Configuration Server will scan the CSDB for Y2K compliance, extended format (ZBASE), and appropriate CSDB ownership (Configuration Server IDs) at startup.
If errors are found, the CSDB utility can be run against the CSDB to correct the error. See "Configuration Server Database Utility (RadDBUtil)" on page 178 for more information.

MGR_DB_VERIFY Settings

Setting	Description
DB_ AUTOFIX	A YES/NO switch to determine if the CSDB should be automatically fixed (where possible) if errors are discovered. The default is **NO**.
VERIFY_ DEPTH	Specifies the level of CSDB verification. The values are DOMAIN, CLASS, INSTANCE, and RESOURCE for this option. The default is **CLASS**. If DOMAIN is selected, the verification process will verify down to the DOMAIN level. If CLASS is selected, the verification process will verify down to the CLASS level. If INSTANCE is selected, the verification process will verify down to the INSTANCE level. If RESOURCE is selected, the verification process will verify the entire CSDB.

Example

```
[MGR_DB_VERIFY]
```

`MGR_VERIFY_DEPTH`	`= CLASS`
`DB_AUTOFIX`	`= YES`

MGR_DB_VERIFY Values

Setting	Value as Installed	Default Value
VERIFY_DEPTH	N/A	CLASS
DB_AUTOFIX	N/A	NO

Performance and Usage Considerations

- A value of VERIFY_DEPTH=RESOURCE will result in longer startup times, as the level of detail is deeper. Conversely, VERIFY_DEPTH=CLASS will result in a much quicker startup time, because the level of verification is decreased.
- Refer to the MGR_ERROR_CONTROL section, which offers handling parameters for errors found during the CSDB verification process.

MGR_DIAGNOSTIC

This section establishes the values that the Configuration Server will use to verify that sufficient disk space is available for CSDB operations and logging, as well as the frequency of verification occurrences.

MGR_DIAGNOSTIC Settings

Setting	Description
DIAGNOSTIC_ INTERVAL	Interval (in seconds) for the diagnostic Configuration Server to verify that sufficient disk space is available for the CSDB and log. If set to 0, monitoring is disabled.
DIAGNOSTIC_ MIN_DB_ BYTES	The minimum number of bytes established for CSDB operations. The maximum value is 2 GB.
DIAGNOSTIC_ MIN_LOG_ BYTES	The minimum number of bytes established for logging operations. The maximum value is 2 GB.

Example

```
[MGR_DIAGNOSTIC]
```

`DIAGNOSTIC_INTERVAL`	`= 900`
`DIAGNOSTIC_MIN_DB_BYTES`	`= 50`
`DIAGNOSTIC_MIN_LOG_BYTES`	`= 25`

MGR_DIAGNOSTIC Values

Setting	Value as Installed	Default Value
DIAGNOSTIC_INTERVAL	N/A	900
DIAGNOSTIC_MIN_DB_BYTES	N/A	50M
DIAGNOSTIC_MIN_LOG_BYTES	N/A	25M

Performance and Usage Considerations

- Include CMD_LINE=(zdiagmgr) in the MGR_ATTACH_LIST section to configure and use this setting.
- On a Configuration Server Database:
  - When the DIAGNOSTIC_MIN_DB_BYTES threshold (2GB) is reached, the following will be issued:
    - an SNMP trap of 2040.
    - a message to the Configuration Server log:
      ```
      (9282 - Warning: The volume containing the Configuration Server
      Database has only %.0lf free bytes).
      ```
  - When the DIAGNOSTIC_MIN_LOG_BYTES threshold (2GB) is reached, the following will be issued:

- an SNMP trap of 2045.
- a message to the Configuration Server log:
  `(9283 - Warning: The volume containing the Configuration Server log has only %.0lf free bytes).`

- On the Configuration Server, you can program a REXX that calls the EDMMAILQ method to send a notification e-mail.
- Set the following line in the MGR_MESSAGE_CONTROL section of the `edmprof` file,
  `9282, 9283=LOG,EVENTLOG`
  (This triggers messages 9282 and 9283 to be created in the Configuration Server log and Windows Event Log.)

MGR_DIRECTORIES

This section specifies the path for the databases, REXX methods, and non-REXX methods. See "MGR_LOG" on page 49 to specify the directory path for the Configuration Server log.

- The EXPORT_PATH setting can be used to define a directory for export operations.
- The five USER_PATH*n* settings can be customized for each Client Automation environment.

MGR_DIRECTORIES Settings

Setting	Description
DBPATH	Fully qualified directory path for object databases.
EXPORT_PATH	Fully qualified directory path for EXPORT operations, accessible via REXX.
METHOD_PATH	Fully qualified directory path for non-REXX methods.
REXX_PATH	Fully qualified directory path for REXX methods.
USER_PATH1	Working directory to be used by users, accessible via REXX.
USER_PATH2	Working directory to be used by users, accessible via REXX.
USER_PATH3	Working directory to be used by users, accessible via REXX.
USER_PATH4	Working directory to be used by users, accessible via REXX.
USER_PATH5	Working directory to be used by users, accessible via REXX.

Example

For Windows:

```
[MGR_DIRECTORIES]
```

`DBPATH`	`= D:/MGR/DB`
`EXPORT_PATH`	`= D:/MGR/BIN/EXPORT`

`METHOD_PATH`	`= D:/MGR/BIN`
`REXX_PATH`	`= D:/MGR/REXX`
`USER_PATH1`	`= D:/MGR/BIN/USER1`
`USER_PATH2`	`= D:/MGR/BIN/USER2`

MGR_DIRECTORIES Values

Setting (REXX Name)	Value as Installed	Default Value
DBPATH	As specified during installation	*Current_directory*
EXPORT_PATH (EXPTPATH)	As specified during installation	*Current_directory*
METHOD_PATH	As specified during installation	*Current_directory*
REXX_PATH	As specified during installation	*Current_directory*
USER_PATH1 (USRPATH1)	As specified during installation	*Current_directory*
USER_PATH2 (USRPATH2)	As specified during installation	*Current_directory*
USER_PATH3 (USRPATH3)	As specified during installation	*Current_directory*
USER_PATH4 (USRPATH4)	As specified during installation	*Current_directory*
USER_PATH5 (USRPATH5)	As specified during installation	*Current_directory*

Performance and Usage Considerations

- The REXX directory specified in this section is further defined by the samples subdirectory, which contains a set of sample REXX methods.

Radia supports Universal Naming Code (UNC), which enables paths in the `edmprof` file to be specified as shown below:

```
\\VFH_LAPTOP\DRIVE_D\CMCSDB
```

Note: When using UNC, the address must be preceded by two backslashes (\\ ), with a single backslash (\ ) used to separate each subfolder.

UNC Connectivity Issues

Interruptions in UNC connectivity to a CSDB might result in critical database classes not being accessed during the resolution process. This failed access could lead to incomplete and erroneous resolution of Client Automation services and, possibly, the inadvertent removal of applications. If this happens, the following errors will appear in the Configuration Server log:

```
NVD7005E 06:19:52 <172.26.132.24 /1930> Radia Client--! ERROR: CLASS
<PRIMARY.POLICY.USER> NOT FOUND
```

```
NVD7005E 06:19:52 <172.26.132.24 /1930> Radia Client--! ERROR: CLASS
<PRIMARY.POLICY.ZBASE> NOT FOUND

ERROR: CLASS <LICENSE.80d40ad0fd2a4ded8c9d26254e8daf0c.MDEVICE> NOT
FOUND

NVD5113E 02:25:17 <172.31.18.5 /668> Radia Client--! ERROR RC=<4>
CREATING INSTANCE<LICENSE.80d40ad0fd2a4ded8c9d26254e8daf0c.MDEVICE.
3B06A9B26C5047D886942D1E2EC4A46E>
```

Also, during Configuration Server startup, the following will be seen in the Configuration Server log:

```
Configuration Server Database is on <Remote> <NTFS> <Uncompressed>
Drive <\\example\RCSDB\>

NVD9268I 02:17:34 <ztoptask /FB4> System Task--- Drive
<\\example\RCSDB\> supports <255> character file names

NVD9269I 02:17:34 <ztoptask /FB4> System Task--- Drive
<\\example\RCSDB\> supports <Case-Sensitivity Case-Preservation
Unicode File-Compression>

NVD9271I 02:17:34 <ztoptask /FB4> System Task--- Database resides in
<\\example\RCSDB\DB\>
```

Note: The UNC-mapped drive will be reflected in the DBPATH setting of the MGR_ DIRECTORIES section in the `edmprof` file.

Recommended Preventive Measures

Note: In addition to the preventive measures detailed here, it is recommended that the Configuration Server be monitored—especially if UNC disconnects are frequently occurring.

It is recommended to take the following measures to minimize the effect of UNC connectivity interruptions.

- At Configuration Server startup, cache all CSDB classes that are used for resolution. (See "MGR_CACHE" on page 36 , and "MGR_CLASS" on page 39.)
- In the MGR_CACHE section of the `edmprof` file, type `ICACHE_LOAD_TYPE=CHUNKY`. (See "MGR_CACHE" on page 36.)
- In the CSDB ensure the following settings for the PRIMARY.SYSTEM.ZMETHOD.LDAP_ RESOLVE method:
  - ZMTHTYPE (method type) is set to `REXX`
  - ZMTHNAME (method name) is set to `RADISH`

  Caution: If policy resolution is being driven by a method other than LDAP_RESOLVE, and the configuration is uncertain, contact Persistent Technical Support before making any changes to the CSDB.

These settings will result in the RCA agent connects and resolutions being stopped—preventing the RCA agents from doing anything, thereby protecting them in the event they are presented an empty catalog. This also prevents the removal of applications.

MGR_DMA

This section specifies the timeout parameter and the directory path for the Distributed Configuration Server (formerly known as the Distributed Manager Adapter, DMA), and enables you to specify values for these options.

Note: You must have the Distributed Configuration Server installed to enable these parameters.

MGR_DMA Settings

Setting	Description
ADMIN_LIST	If SECURITY_METHOD is specified this setting is required. It defines the list of administrator user IDs that are enabled to do login. The format is comma-separated, no spaces, and case-sensitive. For EDMSIGNR, the user IDs must be defined in the Configuration Server's native security system.
DMA_TIMEOUT	Maximum interval (in seconds) that the Distributed Configuration Server will wait for tasks to complete before allowing a commit. A value of 0 means wait indefinitely.
DMA_STAGE_PATH	Path in which staging directories will be created.
SECURITY_METHOD	Name of the security method to be used to verify logins. (Optional) If Distributed Configuration Server security is wanted, the recommended value is EDMSIGNR. If not specified, no login is required.

Examples

Windows Example:

```
[MGR_DMA]
```

`DMA_TIMEOUT`	`= 0`
`DMA_STAGE_PATH`	`= D:\MGR\`

MGR_DMA Values

Setting	Value as Installed	Default Value
ADMIN_LIST	N/A	None
DMA_TIMEOUT	N/A	600

DMA_STAGE_PATH	N/A	ZTOPTASK Path
SECURITY_METHOD	N/A	None

Performance and Usage Considerations

- To decrease the chance of the Distributed Configuration Server synchronization timing out without having committed database updates, increase the value of DMA_TIMEOUT.

MGR_ERROR_CONTROL

This section specifies error-handling parameters for the Configuration Server.

MGR_ERROR_CONTROL Settings

Setting	Description
DBERROR	Describes the type of error and the response. The format is: (*Error Type*) = (*Response*) Valid values for (Response) are: [SHUTDOWN, IGNORE], [NOEMAIL, EMAIL], [NOSNMP, SNMP]
UserEmailErrorsTo	E-mail address of the administrator to whom error messages should be sent.

Note: The actions taken by the settings in this section depend on the levels specified for VERIFY_DEPTH and the errors discovered during MGR_DB_VERIFY processing.

Example

```
[MGR_ERROR_CONTROL]
```

`DBERROR`	`= IGNORE,EMAIL`
`UserEmailErrorsTo`	`= administrator@yourcompany.com`

MGR_ERROR_CONTROL Values

Setting	Value as Installed	Default Value
DBERROR	N/A	[SHUTDOWN] [NOEMAIL] [NOSNMP]
UserEmailErrorsTo	N/A	N/A

Performance and Usage Considerations

- Currently, the only DBERROR supported is DBError. Examples of DBErrors are instances with 0 length, attempting to load a template that does not exist, and so forth.
- The UserEmailErrorsTo value must be a valid e-mail address.

MGR_LOG

This section specifies the logging directory and logging options for the Configuration Server logging facility. It also provides detailed information about reclaiming dormant RCA agent-device licenses.

MGR_LOG Settings

Setting	Description
DIRECTORY	Fully qualified directory path where the Configuration Server log is written.
DISABLE_NT_EVENT_LOGGING	If YES is specified, the Configuration Server logger will disable its Windows Event logging support. This means that messages sent to the Configuration Server log will not be sent to the Windows Event log even if EVENTLOG is specified in the section MGR_MESSAGE_CONTROL. If NO is specified, such messages will be echoed to the Windows Event log if EVENTLOG is specified in the MGR_MESSAGE_CONTROL section. **Note:** This parameter affects only the event logging of Configuration Server log messages. Some Windows Event log records are written without any corresponding Configuration Server log messages. *Windows only*.
DISABLE_SNMP_TRAP_LOGGING	If YES is specified, the Configuration Server logger will disable its SNMP trapping support. This means that messages sent to the Configuration Server log will not be sent to the primary SNMP Manager as traps, even if SNMPTRAP is specified in the section MGR_MESSAGE_CONTROL. If NO is specified, such messages will be sent to the SNMP Manager as traps, if SNMPTRAP is specified in the MGR_MESSAGE_CONTROL section. **Note:** This parameter affects only the trapping of Configuration Server log messages. Some SNMP traps are issued without any corresponding Configuration Server log messages.
FLUSH_SIZE	The number of bytes between automatic flushes of operating system buffers for Configuration Server log file.
MESSAGE_DATE	Allows insertion of the date into every line in the log. Values are JULIAN (YYYYDDD), GREGORIAN (DDMMYYYY), and ISO (YYYYMMDD).
MESSAGE_DELIMITER	The left and right delimiters that are used for enclosing the task name and task ID in the Configuration Server log messages. The format is MESSAGE_DELIMITER=xy, where *x* is the left delimiter and *y* is the right delimiter. The options are: [ ], ( ), < >, and { }. The default is **[ ]**.
MESSAGE_PREFIX	The 3-digit, alphabetic, Configuration Server identifier that will precede Configuration Server log messages. Specify any 3-digit, alphabetic value. The default is NVD.

	Note: If you have log scrapers, verify that they work properly with the NVD setting. If not, change this to `RAD`.
MESSAGE _WIDTH	The maximum width (in bytes) of the messages in the Configuration Server log.
PIPE_SIZE	The maximum amount (in bytes) of log messages that can be queued by the Configuration Server logging facility while the log file is busy. When the value of PIPE_SIZE is reached, any task that issues a log message will freeze until the pipe starts emptying.
SWITCH_ TOD	A time-of-day specification that will, each day, automatically trigger log switching (see "Log Switching" on page 52). The value must be 5 characters, expressed in base-24 time (00:00–23:59), with a colon separator, as in HH:MM. Notes: This is a scheduled event that causes the log to switch; it is independent of, and runs regardless of, any previous log-switching activity. This setting can be used to enable license reclamation using `ZLICUTIL.EXE`. For more information, see "License Reclamation" on page 53. This setting is not part of the Configuration Server installation; it must be manually added. It is recommended that this be set to an off-peak, low-activity time.
THRESHOLD or	The maximum number of lines that will be written to the Configuration Server log before it is automatically switched to the next log. When the limit is reached, a new log file is created, regardless of SWITCH_TOD setting. **Note:** For a more detailed description of this setting and how it can be used for license reclamation, see "Log Switching" on page 52.

Examples

Windows Example:

`[MGR_LOG]`

`DIRECTORY`	`= D:\Program Files\Hewlett-Packard\HPCA\ConfigurationServer\log`
`FLUSH_SIZE`	`= 100000`
`MESSAGE_DATE`	`= JULIAN`
`MESSAGE_ DELIMITER`	`= []`
`MESSAGE_PREFIX`	`= NVD`
`MESSAGE_WIDTH`	`= 256`
`PIPE_SIZE`	`= 1000000`
`SWITCH_TOD`	`= 23:38`

`THRESHHOLD`	`= -5000000`

MGR_LOG Values

Setting	Value as Installed	Default Value	Minimum Value
DIRECTORY	`Program Files\Hewlett-Packard\HPCA \ConfigurationServer \log` `HP/HPCA /Co-nfigurationServer/log`	*`<InstallDir>`* `\Co-nfigurationServer\log` `HP/HPCA/ ConfigurationServer/log`	N/A
DISABLE_ NT _EVENT_ LOGGING	Varies across platforms.	NO	N/A
DISABLE_ SNMP _TRAP_ LOGGING	Varies across platforms.	NO	N/A
FLUSH_SIZE	1000 bytes	5000 bytes	1
MESSAGE_ DATE	N/A	N/A	N/A
MESSAGE _DELIMITER	N/A	[ ]	N/A
MESSAGE_ PREFIX	NVD	NVD	N/A
MESSAGE_ WIDTH	256	90	80
PIPE_SIZE	1000000 bytes	1 MB	1 MB
SWITCH_ TOD	N/A	N/A	N/A
THRESH-OLD	-5000000 lines	100000 lines	1

Performance and Usage Considerations

- Increasing the FLUSH_SIZE will enhance performance, but will delay messages flushed to the log file.
- Increase MESSAGE_WIDTH if log messages are being truncated.
- When closely monitoring system status using the Configuration Server log, set THRESHOLD to a positive value to create and save successive portions of the Configuration Server log. If disk storage space is critical, set the value to a negative number to reuse the allocated log disk space.
- If numerous Configuration Server methods are being invoked, use the MGR_METHODS.LOG_LIMIT setting to control the size of the Configuration Server log for each method. Additionally, the MGR_TASK_LIMIT.TASK_LOG_LIM setting controls the number of messages printed by the execution of each task.
- When modifying parameters in this section as they relate to memory or disk utilization, take care not to exceed the maximum amount of memory or storage space available.

Log Switching

By default, the Configuration Server log will be switched according to the log size value of "THRESHOLD" below and the amount of Configuration Server activity. However, log switching can be configured to automatically occur on a daily basis, at a scheduled time. This is done by specifying the "SWITCH_TOD" below setting and modifying the THRESHOLD setting. THRESHOLD can also dictate what happens to the log that gets rolled over. These sections describe how to do this.

SWITCH_TOD

Specify the time-of-day for the Configuration Server log to automatically roll over. This setting is independent of THRESHOLD, but can be used with it to trigger either of two REXX methods (`ZLOGSWCH.REX` or `ZLOGWRAP.REX`) to launch the user-implemented license-reclamation utility, ZLICUTIL, as described in the section, "License Reclamation" on next page.

THRESHOLD

This setting determines how large the Configuration Server log can be before it is switched to a new log. The value of this setting will dictate what happens to the log that is rolled out, as described in this section.

- If THRESHOLD ≥ 0, the old log will be renamed and saved, and the Configuration Server REXX method, ZLOGSWCH, will run.
  The Configuration Server's log directory (`ConfigurationServer\log`) will have multiple log files whose names conform to the following operating system-specific conventions.

> **Note:** ISO is the International Standards Organization.
> Configuration Server ID is the value of the `edmprof` file's setting MGR_STARTUP.MGR_NAME.

Windows

- The *standard* log naming format is the Configuration Server log prefix (nvd), followed by the letter **r**, and the Configuration Server ID:
  `nvdmr001.log`
- The *log switch* format is the **r** replaced by an **s**, followed by the Configuration Server ID, and the ISO-formatted date and time (each preceded by an underscore) appended:
  `nvdms001_20050427_075835.log`

> **Note:** This THRESHOLD setting will result in multiple Configuration Server log files. To differentiate between them, check the date on which they were modified.
> It is recommended to use this setting because it results in saved, rather than deleted, log files.

- If THRESHOLD < 0, the old log will be overwritten (but not deleted), and the Configuration Server REXX method, ZLOGWRAP, will run.
  The Configuration Server log directory will have just one old log file whose name conforms to the following operating system-specific conventions.

> **Note:** ISO is the International Standards Organization.
> Configuration Server ID is the value of the edmprof file's setting MGR_STARTUP.MGR_NAME.

Windows

- The *standard* log naming format is the Configuration Server log prefix (nvd), followed by the letter **r**, and the Configuration Server ID:
  `nvdmr001.log`
- The *log wrap* (dump) format is the **r** replaced by a **d**:
  `nvdmd001.log`

At the next log switch, the previous log (from the last wrap) will be replaced/overlaid by the newly switched log.

ZLICUTIL

The Configuration Server REXX utility, ZLICUTIL, can be used with ZLOGSWCH and ZLOGWRAP for daily license reclamation. For more information, see License Reclamation 52.

License Reclamation

A Configuration Server license string supports a certain number of RCA agent devices. To keep only active, current devices in a license count, the Configuration Server enables the reclamation of RCA agent-device licenses. If an RCA agent device hasn't connected to the Configuration Server for a specified number of days, its license becomes inactive and can be reclaimed in the license count by the Configuration Server utility, ZLICUTIL.

Considerations

- By default, license reclamation occurs only at Configuration Server shutdown.
- To make license reclamation occur daily:
  - Specify values for SWITCH_TOD and THRESHOLD (see "SWITCH_TOD" on page 52 and "THRESHOLD" on page 52 or in "MGR_LOG" on page 49).
  - Modify the appropriate REXX (either ZLOGSWCH or ZLOGWRAP) so that a log-switching REXX method will launch the license utility, ZLICUTIL, when it gets called.
- Licenses that are aged out via ZLICUTIL are not immediately available. They are reclaimed only at the next midnight.

  > **Note:** The exception to this condition is that aged-out licenses will be immediately available when the Configuration Server is restarted.

The following instructions detail how to implement license reclamation on a Configuration Server.

To Reclaim Licenses Daily

1. Add the log switch setting, SWITCH_TOD, to the MGR_LOG section of the `edmprof` file, and specify a valid value (seeSWITCH_TOD in the table, "MGR_LOG" on page 49 such as:
   `[MGR_LOG]SWITCH_TOD = 23:45`
2. Specify a value for THRESHOLD or, as described .
3. Navigate to the directory in which the Configuration Server methods reside, such as:
   <InstallDir>`\ConfigurationServer\rexx\NOVADIGM`
   and copy either ZLOGSWCH or ZLOGWRAP (or both) up one level to the `rexx` directory.

   > **Caution:** During the Configuration Server installation, the REXX methods are, by default, installed to the platform-specific directories that are listed in step 3.
   > Before making changes to either ZLOGSWCH or ZLOGWRAP, be sure to copy it up one level to the rexx directory

4. Open the ZLOGSWCH (or ZLOGWRAP) file, and add the commands shown here:
   ```
   call edmget zcvt
   ADDRESS CMD “ZLICUTIL” zcvt.dbpath zcvt.rptpath zcvt.uuid
   zcvt.mgrid zcvt.syspath || license.nvd
   ```

   > **Note:** See the section, "Configuration Server Running as a Windows Service" on next page, for Windows-specific configuration information.

5. Save and close ZLOGSWCH (or ZLOGWRAP).
   From this point on, the license reclamation feature will be started asynchronously when the log switch is invoked.

Configuration Server Running as a Windows Service

This section describes the changes that are necessary if the Configuration Server is running as a service on a Windows machine.

Note: This information is relevant *only* if the Configuration Server is configured as a Windows service; if it is not, this information is not applicable.

The directory in which Windows always first looks for its “service” programs is `Windows\system32`. Therefore, in order for a program to be run as a service, it must be located here; if it is not, it won’t be found by Windows—or Windows has to be instructed to look elsewhere.

Even though the Configuration Server might be set up to run as a Windows service, it probably isn’t in this directory—rather, it is likely in the directory that was specified during the installation (the default of which is <*InstallDir*>`\ConfigurationServer`). Therefore, Windows has to be told where to find the program files. Instructions for doing this are in the following section.

Directing Windows to the Configuration Server Program Files

To resolve the issue of Windows not being able to find the Configuration Server files, make the following changes to the ZLOGSWCH (or ZLOGWRAP) REXX method.

1. Specify the *fully qualified path* of the directory in which ZLICUTIL resides so that Windows will know where to look. (See the example that follows.)
2. Add a fifth argument—the *fully qualified path* of the directory in which the file `license.nvd` resides.

Examples

Original ZLOGSWCH (or ZLOGWRAP) REXX method command:

```
ADDRESS CMD “START ZLICUTIL” zcvt.dbpath
“<InstallDir>\ConfigurationServer\log” “05c87c3e95194a9d9251fee5cbfddafb”
zcvt.mgrid
```

Revised ZLOGSWCH (or ZLOGWRAP) REXX method command:

```
ADDRESS CMD “START FULLY_QUALIFIED_BIN_PATH\ZLICUTIL.EXE” zcvt.dbpath
“<InstallDir>\ConfigurationServer\log” “05c87c3e95194a9d9251fee5cbfddafb”
zcvt.mgrid “FULLY_QUALIFIED_BIN_PATH\LICENSE.NVD”
```

MGR_MESSAGE_CONTROL

This section specifies which log messages are to be sent and to where, or if they are to be suppressed.

MGR_MESSAGE_CONTROL Settings

Setting	Description
nnnnn = (*Destination*)	(*Message_Number*) = (*Message_Destination*)

Note: The left side of the equals sign (=) specifies which messages are to be affected by the command. There is an "ALL" directive that can be used to affect all messages (0001-9999). Also, if there is no destination after the equals sign, the message has no associated destination, so it is suppressed.

The following table presents a list of the six destinations for log messages.

MGR_MESSAGE_CONTROL Log Message Destinations

Destination	Description
EVENTLOG	Write log messages to the Windows Event log. (*Windows only*)
LOG	Write log messages to the Configuration Server log.
REXX	Write log messages to a pre-determined REXX (named MGRLOG). Important: MGRLOG is not an existing REXX, and therefore, must be created to use this facility. **Note:** The MGRLOG REXX will receive the message number and the message text as its first and second input parameters. You can then process the information in whatever manner you chose.
SNMPTRAP	Write log messages as traps to the current SNMP Manager.
USERLOG	Write log messages to the user log. Note: You must first enable the user log feature in the MGR_USERLOG section of the `edmprof` file, by specifying ACTIVATE=YES.

Example

```
[MGR_MESSAGE_CONTROL]
```

`ALL`	`= SNMPTRAP`
`500`	`=`
`520-600`	`= LOG`
`225, 300`	`=`
`220-299, 400, 542-545, 803`	`= EVENTLOG,REXX,USERLOG`

In this example:

- The first line will send all messages to the SNMP Manager as traps.

  Note: Keep in mind that this means that no messages will be written to the Configuration Server log; all messages will go out as SNMP traps only, unless subsequent entries specifically override this.

- The second line will cause message 500 to be suppressed—that is, it will not be written to the Configuration Server log.
- The third line will cause messages 520 through 600 (inclusive) to be written to the Configuration Server log.
- The fourth line will suppress messages 225 and 300 only; all others will be written to the Configuration Server log.
- The fifth line will cause messages 220 through 299 (inclusive), 400, 542 through 545 (inclusive), and 803 to be written to the Windows Event log, the pre-determined REXX, and the user log.

MGR_MESSAGE_CONTROL Values

Setting	Value as Installed	Default Value
nnnnn = (*Destination*)	N/A	ALL=LOG

Performance and Usage Considerations

- SNMPTRAP should be used as a destination only if the address of the SNMP Manager has been configured in the SNMP section.
- Any line that does not contain an equals sign (=) is treated as a comment. In addition, lines that begin with an asterisk (*****), double forward slashes (//), or a slash (/) are treated as comments. Blanks and tabs can occur anywhere on the line, even ahead of the comment specifications.
- ALL = will suppress all messages.
- Any errors encountered in parsing a line cause the entire line to be ignored and an error message to be written to STDERR.

MGR_METHODS

This section specifies options for method execution.

MGR_METHODS Settings

Setting	Description
LOG_ LIMIT	Maximum number of messages that a method can issue to the Configuration Server log. When this limit is reached, a message will be written stating that the message limit has been reached and that all other messages from the method will be ignored.
TIMEOUT	The duration (in seconds) that a Configuration Server will wait for a response from a method that is running. After this interval, if no response is received, the Configuration Server will terminate the method.

Example

```
[MGR_METHODS]
```

`LOG_LIMIT`	`= 0`
`TIMEOUT`	`= 300`

MGR_METHODS Value

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
LOG_LIMIT	0	0 = No Limit	0	2,147,483,647
TIMEOUT	300 (seconds)	60 (seconds)	0	32000

Performance and Usage Considerations

- The number of messages generated by the execution of a method is also affected by the TASK_LOG_LIM setting in MGR_TASK_LIMIT.
- Tune system resource usage with the TIMEOUT setting. If system resources are critical, lower the TIMEOUT setting to free up unused processing cycles.
- When the TIMEOUT value is reached, the method is terminated and messages are written to the Configuration Server log.
- If the TIMEOUT=0, the method will continue in effect until its conclusion.

MGR_NOTIFY

This section specifies the defaults for the Configuration Server's RCA agent notification.

MGR_NOTIFY Settings

Setting	Description
ISSUE_WAKE_ON_LAN	Enables support for TCP Wake-On-LAN.
NFY_RETRY	Number of times to attempt re-notification after initial, unsuccessful attempt.
NFYT_TIMEOUT	Notify timeout (in seconds) for TCP/IP RCA agents.
RECOVERY_DOMAIN	The Domain that the Retry Manager will access to reinitiate notifies after the Configuration Server has prematurely shut down.
SUBNET_MASK	This value is used to convert a destination IP address into a subnet address for EDMWAKE. Note: Applicable only to version 4.3 CM Configuration Server, with Service Pack 2.

WAKE_ON_LAN_TTL	This value is the number of routers that the WOL broadcast is permitted to pass. The default is **99**. Note: Before establishing this value, consult your network administrator.

Example

```
[MGR_NOTIFY]
```

`NFYT_TIMEOUT`	`= 120`
`NFY_RETRY`	`= 5`
`SUBNET_MASK`	`= 255.255.0.0`
`ISSUE_WAKE_ON_LAN`	`= YES`
`RECOVERY_DOMAIN`	`= ALL`
`WAKE_ON_LAN_TTL`	`= 99`

Note: In some network addressing schemes, a class A IP address is used with a class B or class C subnet mask. This often results in problems, caused by parameters passed by a Configuration Server while trying to invoke EDMWAKE. If specified, the SUBNET_MASK parameter will cause the Notify Manager to use this mask from the `edmprof` file, rather than create the subnet mask according to the network type.
For example, if the notify for IP address, 192.168.5.5 failed, EDMWAKE will be issued for subnet address, 192.168.255.255.

MGR_NOTIFY Values

Setting	Value as Installed	Default Value	Value Range
ISSUE_WAKE_ON_LAN	N/A	NO	YES/NO
NFY_RETRY	N/A	0	32000
NFYT_TIMEOUT	120 seconds	0	32000
RECOVERY_DOMAIN	N/A	ALL	RETRY/ALL
SUBNET_MASK	N/A	N/A	N/A
WAKE_ON_LAN_TTL	99	99	99

Performance and Usage Considerations

- Establish MGR_NOTIFY settings based on network operations parameters.
- The MGR_NOTIFY settings should be coordinated with values in the MGR_RETRY and MGR_TASK_LIMIT sections.
- In order for the Wake-On-LAN Notify function to operate, and to enable the retrying of failed operations, zrtrymgr must be specified under MGR_ATTACH_LIST.

MGR_OBJECT_RESOLUTION

This section specifies the parameters to use during object resolution.

MGR_OBJECT_RESOLUTION Settings

Setting	Description
ALLOW_ DUPLICATE _INSTANCES	Allow or disallow duplicate instances to be used during object resolution. Values are YES and NO.
ALWAYS_ CALL _ZADMIN	Force object resolution to call the ZADMIN method to process the ZADMIN object. Values are YES and NO.
ZERRORM_ MAX _ERRORS	Maximum number of errors associated with a ZERRORM event.
ZERRORM_ MAX _WARNINGS	Maximum number of warnings associated with a ZERRORM event.

Example

```
[MGR_OBJECT_RESOLUTION]
```

`ALWAYS_CALL_ZADMIN`	`= YES`
`ZERRORM_MAX_WARNINGS`	`= 50`
`ZERRORM_MAX_ERRORS`	`= 50`
`ALLOW_DUPLICATE_INSTANCES`	`= YES`

MGR_OBJECT_RESOLUTION Values

Setting	Value as Installed	Default Value
ALWAYS_CALL_ZADMIN	YES	NO
ALLOW_DUPLICATE_INSTANCES	N/A	YES
ZERRORM_MAX_WARNINGS	N/A	50
ZERRORM_MAX_ERRORS	N/A	50

Performance and Usage Considerations

- ALLOW_DUPLICATE_INSTANCES=NO will cause the Configuration Server to eliminate duplicate services at resolution. This might result in the elimination of duplicate

SOFTWARE.FILE instances on the RCA agent. As the size of the SOFTWARE.FILE object expands, the resolution time will increase.

- It is recommended that you do not modify or remove ALWAYS_CALL_ZADMIN, as doing so might prohibit administrator type functions.

MGR_POLICY

This section specifies the IP address/name, port, and details of Policy Resolution process of the Radia Client Automation Policy Server (Policy Server), if it has been enabled.

Note: This section is present only if the Policy Server option is selected during the Configuration Server installation.

MGR_POLICY Settings

Setting	Description
HTTP_ HOST	The IP address of the Policy Server. If the Policy Server is co-located with the Configuration Server, specify `localhost`.
HTTP_ PORT	The port of the Policy Server. The default is **3466**.
RADISH_ DEBUG	Provides details of Policy Resolution process. If information messages on details of the Policy Resolution process are required, set RADISH_DEBUG=`1`. The default is **0**.

Example

```
[MGR_POLICY]
```

`HTTP_HOST`	`= localhost`
`HTTP_PORT`	`= 3466`
`RADISH_DEBUG`	`= 1`

MGR_POLICY Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
HTTP_HOST	None	N/A	N/A	N/A
HTTP_PORT	N/A	3466	N/A	N/A
RADISH_DEBUG	0	0	N/A	1

MGR_POOLS

Note: This section of the `edmprof` file is specific to Windows operating systems.

This section enables you to specify allocations (pools) of memory in different sizes, before Configuration Server startup. These allocations can be changed dynamically while the Configuration Server is running by using modify commands. In addition, you can establish percentage-of-waste tolerances (the amount of memory that is not used by a specific requirement) for each of the allocations.

You can establish pools of any size. The pool sizes specified, however, should be multiples of eight. If not, they will be rounded up to the next multiple of eight by the system. Likewise, you can specify any number of pools.

Note: The sizes and number of pools you establish, as well as the percentage of waste tolerated, should be based on the workload and operating requirements of your environment.

When the Configuration Server requires memory, the pools are searched from smallest to largest until a pool is found in which the memory requirement fits. When that pool is found, the first item on the free list is used to resolve the request—providing that the percent of space wasted is not above the value specified for that pool. If there is no item on the free list, then the memory allocation is redirected to the standard C heap. If the C heap is also exhausted, a host system native-storage request will be performed to satisfy the memory request.

The format for specific pool allocation is:

(Pool Size) = (number of elements for that pool size),
(specific percentage of waste tolerated),(expansion increments).

Note: The specific percentage of waste tolerated parameter is optional. The expansion increments default is **1**.

MGR_POOLS Settings

Setting	Description
WASTE_ TOLERATED	The general percentage of waste that will be tolerated to fit a specific requirement to an available pool. The default is **100** (%).
ALLOCATION_ SIZE _ERROR_ THRESHOLD	Deny memory allocations above this size.
ALLOCATION_ SIZE_ REPORTING_ THRESHOLD	Report memory allocations above this size.
XXXXXX	The allocation for the *XXXXXX* byte pool. Any reasonable number of these can be specified.

By default, the MGR_POOLS section establishes the pool sizes shown in the example below.

Example

```
[MGR_POOLS]
```

`WASTE_TOLERATED`	`= 100`
`ALLOCATION_SIZE_ERROR_THRESHOLD`	`= 4194304`
`ALLOCATION_SIZE_REPORTING_THRESHOLD`	`= 65536`
`168`	`= 150`
`296`	`= 50`
`552`	`= 20`
`1064`	`= 10`
`2100`	`= 5000`
`2700`	`= 100`
`4136`	`= 4`
`8232`	`= 2`
`12500`	`= 2`

MGR_POOLS Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
WASTE_ TOLERATED	100	100 (%)	0	100 (%)
ALLOCATION_SIZE _ERROR_ THRESHOLD	4 MB	4 MB	0 (disabled)	2 GB
ALLOCATION_ SIZE_ REPORTING_ THRESHOLD	64 KB	64 KB	0 (disabled)	2 GB
XXXXXX	N/A	(See "MGR_POOLS" on previous page)	0	32767

Performance and Usage Considerations

- Provisions should be made for pools that are the product of the MGR_CLASS section of the STARTUP member, as requests for these memory elements will automatically occur at the initiation of resolution (generally, the receipt of the ZMASTER object). Special attention should be given to the pools that will be identified by the product of the third and fourth values specified in MGR_CLASS, as in Y,N,3072,500 for the MGR_CLASS entry for SOFTWARE.FILE.

- The product of the third and fourth values of this is a value exactly equal to 1.5 MB, and one of these will be allocated at the initiation of each resolution. Additional storage is required for storage management of these queues, so a storage pool of elements size (3072) * (500) + 500 = 1536500 should be created. The number of elements to assign to this pool is dependent on the MGR_TASK_LIMIT values specified (for example, one element for each concurrent task), and the number of SOFTWARE.FILE instances resolved for each RCA agent.
- When using the Distributed Configuration Server, the default setting of ALLOCATION_SIZE_ERROR_THRESHOLD must be changed to prevent a destination CSDB being only partially updated. If the Distributed Configuration Server transfers a resource file that is larger than the default (4 MB), and the setting has not been changed, the 'commit' operation will halt, leaving the destination CSDB only partially updated. See the examples below for completing this:
  - If you use memory pooling, set ALLOCATION_SIZE_ERROR_THRESHOLD to zero, as below:
    `[MGR_POOLS]ALLOCATION_SIZE_ERROR_THRESHOLD = 0`
    Here, 0 will disable the memory allocation limit.
  - If the MGR_POOLS section does not exist in the `edmprof` file, there are no issues and no changes are required.

Pool Values

Size	64	400	632	2048	4120	6504	8200	12000
Number	100	100	100	60	60	30	15	20
Waste %	100	100	100	100	100	89	65	75
InUse	21	2	12	0	0	0	1	0
HighWM	21	15	12	1	1	0	1	4
Allocs	30	656	19	1	5	0	1	218
Empty	0	0	0	0	0	0	0	0
Waste	0	0	0	0	0	0	0	0

- **Size** is the size of the elements of the pool.
- **Number** is the number of elements currently in the pool.
- **Waste %** is the percentage of waste tolerated at storage element allocation time.
- **InUse** is the number of elements that were in use when the stats call was made.
- **HighWM** is the high watermark that shows the maximum number of elements that have been allocated at any one time since startup or a clear command.
- **Allocs** is the total number of elements that were allocated since startup or since the last clear command on that pool.

- **Empty** is the number of times we failed to get an element because there were no free elements available.
- **Waste** is the number of times we failed to get an element because the waste was higher than the tolerated waste percent.

In the example in "Performance and Usage Considerations" on page 63, the pools of 64, 400, 632, 2048, and 4120 do not have a specified waste tolerance. This is because these pool sizes are so small that the waste in them will not be an issue. There is a DISABLE command that looks like: `F jobname,DISABLE,2048`or just: `F jobname,D,2048`This would disable pool 2048, that is, it would prevent new allocations from using that pool. Allocations between 633 and 4120 bytes would then come from the 4120 pool, provided the tolerated waste percentage is met. A disabled pool will show up on the command with a “D” after the size. A disabled pool can be re-enabled with the ENABLE command, as follows: `F jobname,E,2048`The pool counts can be cleared with the CLEAR command: `F jobname,C,2048`

Note: The HighWM, Allocs, Empty, and Waste counters are cleared by the above command.

Lastly, there is an ADJUST command, which looks like: `F jobname,A,2048,80,95`This command would add 20 free elements to the 2048 pool (60 + 20 = 80) and would set its waste tolerated percentage to 95. The last parameter (percentage of waste tolerated) is optional. The “A” denoting the ADJUST subcommand can be replaced with ADJUST, ADJUS, ADJU, ADJ, or AD. The Enable, Disable, and Clear commands can also be specified in a longer form. If you use the ADJUST command to set the number of elements to 0, then the pool is completely deleted if all the elements can be freed. However, the pool count will not go to 0 as long as there are allocated elements. New pools can be added on the fly with the ADJUST command. Finally, storage trace (STORAGE=YES) can be used to produce a message in the log each time a storage allocation is made and freed. This can be used to tune the pool, and to understand the Configuration Server’s use of dynamic memory.

MGR_RESOLUTION_FILTERS

This section defines filtering rules for resolution, based on connection type.

Note: This section is not included in the `edmprof` file at installation—it must be manually added.

MGR_RESOLUTION_FILTERS Settings

Setting	Description
DOMAIN.CLASS	Specify the domain and class of the CSDB for which to define filtering rules for resolution.

Valid values for MGR_RESOLUTION_FILTERS are:

- RADIA – Allow Client Automation resolution only.
- ANY – Allow all resolutions. (This is the default.)

Example

```
[MGR_RESOLUTION_FILTERS]
```

SOFTWARE.ZSERVICE	= RADIA
POLICY.WORKGROUP	= RADIA
POLICY.*	= RADIA

MGR_RESOLUTION_FILTERS Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
DOMAIN.CLASS	N/A	N/A	N/A	N/A

Performance and Usage Considerations

- CLASS can be specified as a valid CSDB class, or with a wildcard (*****) to specify all classes in a domain.

MGR_RETRY

This section specifies how soon (in minutes) an RCA agent can attempt another connection to the Configuration Server, after being rejected due to an exceeded task limit or a disabled connection.

MGR_RETRY Settings

Setting	Description
BUSY_ RETRY	Number of minutes for an RCA agent to wait before reconnecting when the Configuration Server is at its task limit (TASK_LIMIT).
DISA_ RETRY	Number of minutes for an RCA agent to wait before reconnecting when the Configuration Server has logons halted.

Example

```
[MGR_RETRY]
```

BUSY_RETRY	= 1
DISA_RETRY	= 999

Note: It is recommended that the values of at least 1 for these settings to avoid unnecessarily tying up the Configuration Server.

MGR_RETRY Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
BUSY_ RETRY	7 minutes	0 (allow connect now)	0 (allow connect now)	999 (do not retry)
DISA_ RETRY	999 (do not retry)	999 (do not retry)	0 (allow connect now)	999 (do not retry)

Performance and Usage Considerations

- You can raise these values if processing resources are critical. Lowering these values will provide greater assurance that connections will be re-established if broken during RCA agent connects.
- The MGR_RETRY settings should be coordinated with values in the MGR_NOTIFY and MGR_ TASK_LIMIT sections.

MGR_RIM

This section specifies the IP address/name and port of the Radia Client Automation Inventory Manager (Inventory Manager) if it has been enabled.

Note: This section will be present only if the Inventory Manager option was selected during the Configuration Server installation.

MGR_RIM Settings

Setting	Description
HTTP_ HOST	The IP address of the Inventory Manager. If the Inventory Manager is co-located with the Configuration Server, specify `localhost`.
HTTP_ PORT	The port of the Inventory Manager. The default is **3466**.

Example

```
[MGR_RIM]
HTTP_HOST = localhost
HTTP_PORT = 3466
```

MGR_RIM Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
HTTP_HOST	None	N/A	N/A	N/A
HTTP_PORT	N/A	3466	N/A	N/A

MGR_RMP

This section specifies the IP address/name and port of the Radia Client Automation Portal (Portal), if it has been enabled.

Note: This section will be present only if the Portal option was selected during the Configuration Server installation.

MGR_RMP Settings

Setting	Description
HTTP_ HOST	The IP address of the Portal. If the Portal is co-located with the Configuration Server, specify `localhost`.
HTTP_ PORT	The port of the Portal. The default is **3466**.

Example

```
[MGR_RMP]
HTTP_HOST = localhost
HTTP_PORT = 3466
```

MGR_RMP Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
HTTP_HOST	None	N/A	N/A	N/A
HTTP_PORT	N/A	3466	N/A	N/A

MGR_ROM

Specifies information used for OS management.

Caution: Do not change any of these settings unless requested to do so by your Persistent Software Support representative

MGR_ROM Settings

Setting	Description
LDAP_ DIRECT_ ENABLED	Enable the use of OpenLDAP authentication for OS Management operations (enabled by default)

LDAP_ DIRECT_HOST	Host name or IP address of a server where OpenLDAP is active
LDAP_ DIRECT_PORT	Port number to use for OpenLDAP authentication
LDAP_ DIRECT_UID	User ID for OpenLDAP authentication
LDAP_ DIRECT_PASS	Password for OpenLDAP authentication
PORTAL_ HOST	Host name or IP address for the RCA Portal
PORTAL_ PORT	Port number for the RCA Portal
PORTAL_ ZONE	Zone name in the RCA Portal
DISPLAYNAME	Display name used in the RCA Portal for the device
PORTAL_UID	User ID of an RCA Portal user who can update a device or the ROM object
PORTAL_PASS	Password of an RCA Portal user who can update a device or the ROM object
PORTAL_USE_ SSL	Enable use of SSL to connect to the RCA Portal (handled by the UI)
FORCE_ZONE	Used for migration from an HPCA Classic installation to an HPCA Core and Satellites installation. This setting ensures that non-standard zone names (different from cn=hp,cn=radia) are handled without cancelling the OS installation process.

Example

```
[MGR_ROM]

LDAP_DIRECT_ENABLED=1

LDAP_DIRECT_HOST = localhost

LDAP_DIRECT_PORT = 3474

LDAP_DIRECT_UID = (contains AES encrypted user ID value)

LDAP_DIRECT_PASS = (contains AES encrypted password value)

PORTAL_HOST = localhost

PORTAL_PORT = 3466
```

```
PORTAL_ZONE = cn=hp, cn=radia

FORCE_ZONE = true

DISPLAYNAME = compname

PORTAL_UID = (contains AES encrypted user ID value)

PORTAL_PASS = (contains AES encrypted password value)

PORTAL_USE_SSL = 0
```

MGR_SMTP_MAIL

This section specifies SMTP-related parameters, including the ID of the Configuration Server and the TCP port number of the Configuration Server to be used.

MGR_SMTP_MAIL Settings

Setting	Description
DNS_ SERVER	The Domain Name Service (**DNS**) server that is used to query the mail server address. **Note:** Specify a value for this setting to ensure that mail is delivered.
MAIL_DIR	Directory to spool and queue outgoing mail from the SMTP send Manager.
MAIL_ TIMEOUT	Timeout interval (in seconds) for establishing communications with the mail server.
MAX_ TIME_IN_ SPOOL	The interval (in minutes) to wait before deleting undelivered mail in the spool. The default is **4320** minutes (3 days).
MGR_ MAIL_ID	The mail ID for the Configuration Server. This setting takes the form of a fully qualified address (for example, `ConfigServer@HP.com`) and has a maximum length of 255 bytes. The mail-receiving Configuration Server will reject mail that is addressed to any other user ID.
RETRY_ INTERVAL	The interval (in seconds) to wait before re-attempting to deliver mail in the spool. The default is **300** seconds (5 minutes).
SMTP_ PORT	The port on which the Configuration Server's SNMP Manager *send* and *receive* tasks (ZSMTSMGR and ZSMTRMGR, respectively) will listen for incoming mail.

Examples

Windows Example:

```
[MGR_SMTP_MAIL]
```

`DNS_SERVER`	`= 192.168.1.20`

MAIL_DIR	`= D:\MGR\MAIL`
`MAIL_TIMEOUT`	`= 60`
`MAX_TIME_IN_SPOOL`	`= 4320`
`MGR_MAIL_ID`	`= ConfigServer@HP.com`
`RETRY_INTERVAL`	`= 300`
`SMTP_PORT`	`= 25`

MGR_SMTP_MAIL Values

Setting	Value as Installed	Default Value
DNS_SERVER	N/A	NONE
MAIL_DIR	as specified during installation	*current_directory*
MAIL_TIMEOUT	N/A	60
MAX_TIME_IN_SPOOL	4320 minutes (3 days)	4320 minutes (3 days)
MGR_MAIL_ID	as specified during installation	*sending_address*
RETRY_INTERVAL	300 seconds (5 minutes)	300 seconds (5 minutes)
SMTP_PORT	as specified during installation	25

Performance and Usage Considerations

- If storage capacity is an issue, decrease the MAX_TIME_IN_SPOOL value. This will decrease the length of time messages are retained.
- Increase the RETRY_INTERVAL value to use fewer system processing resources.
- DNS_SERVER: For cross-platform compatibility, operating system network settings for DNS servers are not used by the Configuration Server mail delivery. Therefore, to ensure delivery of all e-mail (including license warning messages) that are sent by the Configuration Server, DNS_SERVER must be defined.

MGR_SNMP

This section contains SNMP-related parameters that include where SNMP traps are to be sent and how to control the behavior of the built-in SNMP agent.

MGR_SNMP Settings

Setting	Description
RUN_AS_EXTENSION	If YES, the Windows SNMP service is the primary SNMP agent, and HP SNMP transactions are processed by a HP SNMP extension DLL. If so, SNMP_PORT and SNMP_IP_ADDR are not used because SNMP port access is handled by the Windows SNMP service. The value of SNMP_COMMUNITY

	will be used for insertion into traps that are issued by the Configuration Server, but will not be used to authenticate GET and SET commands. If NO, the Configuration Server will act as the primary SNMP agent, and SNMP_ COMMUNITY should be specified, while SNMP_IP_ADDR and SNMP_PORT can be specified to override their defaults.
SNMP_ COMMUNITY	This is a password that incoming SNMP transactions must match. It should be set to a character string. The string will be used as the SNMP community name by the agent. The default is **public**. **Note:** This keyword is effective only if `RUN_AS_EXTENSION=NO`.
SNMP_IP_ ADDR	The TCP/IP address of the local network adapter card on which the agent is to receive SNMP transactions. The default is **0.0.0.0**, meaning any adapter on the machine can be used. Note: This keyword is effective only if `RUN_AS_ EXTENSION=NO` and there are several adapters on the machine, and a specific adapter is to receive SNMP transactions.
SNMP_ MANAGER_ IP _ADDR SNMP_ MANAGER_ IP _ADDR2 SNMP_ MANAGER_ IP _ADDR3	The SNMP Managers at the IP addresses specified here are authorized to issue GET and SET commands for variables supported by the agent. The SNMP Manager specified for SNMP_MANAGER_IP_ADDR is considered the primary SNMP Manager. This field is required because it specifies two things: 1) the receiving location of traps created by the Configuration Server, and 2) the address of the authorized source for SNMP commands.
SNMP_ MANAGER _PORT	This parameter is used to specify the remote TCP/IP port to which the Configuration Server sends its traps. The default is port **162**.
SNMP_PORT	This parameter is used to specify the TCP/IP port on which the agent receives SNMP transactions. The default is port **161**.
	Note: This keyword is effective only if `RUN_AS_EXTENSION=NO`.
SNMP_SET _ COMMUNITY	This parameter can be set to a character string. The agent will use this string as the SNMP community name when it is attempting to authorize SET commands. If this keyword is not specified, the community name given by SNMP_ COMMUNITY is used for SET commands.
	Note: This keyword is effective only if `RUN_AS_EXTENSION=NO`.
SNMP_ ZERROR	This parameter is used to specify the severity of ZERROR instances to send as SNMP traps. The trap is sent when the Configuration Server adds an error

	instance to its ZERRORM for an error whose severity is greater than or equal to the value specified by this parameter. The parameter can be set to a positive value between 0 and 99; the default is **12**.

Example

```
[MGR_SNMP]
```

`RUN_AS_EXTENSION`	`= NO`
`SNMP_COMMUNITY`	`= public`
`SNMP_IP_ADDR`	`= 0.0.0.0`
`SNMP_PORT`	`= 162`
`SNMP_MANAGER_IP_ADDR`	`= 183.235.246.32`
`SNMP_ZERROR_SEVERITY`	`= 12`

MGR_SNMP Values

Setting	Value as Installed	Default Value
RUN_AS_EXTENSION	YES	NO
SNMP_COMMUNITY	public	public
SNMP_IP_ADDR	0.0.0.0	0.0.0.0
SNMP_MANAGER_IP_ADDR	N/A	N/A
SNMP_MANAGER_PORT	N/A	162
SNMP_PORT	161	161
SNMP_SET_COMMUNITY	N/A	N/A
SNMP_ZERROR_SEVERITY	12	12

Performance and Usage Considerations

There are no performance or usage issues with any of the SNMP parameters. If you do not start zsnmpmgr, you are not starting the RCA agent, and are running one less task in the Configuration Server.

MGR_SSL

This section specifies the operational settings for an SSL Manager. For more information on configuring and using an SSL Manager, see, *Radia Client Automation Enterprise User Guide*.

MGR_SSL Settings

Setting	Description
CA_FILE	Sets the Certificate Authority's certificate. The CA certificate is usually stored in a file in Privacy Enhanced Mail (PEM) format. The SSL Manager needs a CA certificate to start up. If it's expired or corrupt it prevents the SSL Manager from starting up.
CERTIFICATE_FILE	Sets the Configuration Server or the server certificate. The certificate is usually stored in a file in PEM format. This value must be a valid and existing certificate file. The SSL Manager needs a certificate to start up. An expired or corrupt certificate will prevent the SSL Manager from starting up.
KEY_FILE	Sets the private key. The private key is usually stored in a file in PEM format. This value must be a valid and existing key file. The private key is usually stored in the same file as the server certificate, in which case, you don't have to specify any value for the key file.
KEY_PASSWORD	The password that is used to encrypt the private key, the one specified in the KEY_FILE keyword. This is usually needed if the private key is encoded. If the private key is not encoded, you don't need to specify this parameter. **Note:** This setting is not automatically added to the `edmprof` file by the installation; it must be manually added by an RCA administrator.
SSL_CIPHERS	Specifies the preferred OpenSSL ciphers to use for SSL connections between the Configuration Server and other RCA components. For example: `ALL:ADH:RC4+RSA:+HIGH:+MEDIUM:+EXP:+eNULL` If you do not specify SSL_CIPHERS, the default ciphers are used: `ALL:ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL` Refer to the OpenSSL web site additional information: http://openssl.org/docs/apps/ciphers.html
SSL_PORT	The port on which the SSL Manager will listen for RCA agent connects.
VERIFY_CLIENT	Specifies whether the Configuration Server should verify the RCA agent by requesting a certificate from it. If the RCA agent doesn't have a certificate, the connection will be dropped. **Note:** This setting is not automatically added to the `edmprof` file by the installation; it must be manually added by an RCA administrator.

Example

```
[MGR_SSL]
```

`CA_FILE`	`= w:\openssl\ms\cacert.pem`
`CERTIFICATE_FILE`	`= w:\openssl\ms\srvcert.pem`

`KEY_FILE`	`= w:\openssl\ms\srvprvk.pem`
`KEY_PASSWORD`	`= violin`
`SSL_PORT`	`= 3456`
`SSL_CIPHERS`	`= ALL:ADH:RC4+RSA:+HIGH:+MEDIUM:+EXP:+eNULL`
`VERIFY_CLIENT`	`= Y`

Performance and Usage Considerations

- In order for an SSL Manager to function, `CMD_LINE=(zsslmgr) RESTART=YES`must be specified in the MGR_ATTACH_LIST section.
- The settings are placeholders and, as such, are not used until an SSL add-on is installed.

MGR_STARTUP

This section specifies startup information for the Configuration Server.

Caution: After a successful installation:
DO NOT change MANAGER_TYPE, MEMORY_TYPE, MGR_ID, MGR_NAME, or TASK_TYPE, unless advised to do so by Persistent Technical Support.
DO NOT change or delete MGR_UUID unless instructed to do so by Persistent Technical Support.

MGR_STARTUP Settings

Setting	Description
ALLOW_DUPLICATE_IP_ADDRESS	NO will cause the Configuration Server to reject a second log on if one from the same IP address is already active. YES will allow multiple concurrent IP connections from the same RCA agent IP address.
BYTE_LEVEL_DIFF	Turns on byte-level differencing during resolution.
MANAGER_TYPE	Type of Configuration Server; valid values are DISTRIBUTED (the default), SERVER, and STANDALONE. For more information on this setting, see "MANAGER_TYPE Values" on page 78. **Note:** If the Configuration Server is going to participate in Distributed Configuration Server operations, its type must be DISTRIBUTED or SERVER. Important Note: Do not alter this setting unless instructed to do so by Persistent Technical Support.
MEMORY_	Reserved for future development. Important Note: Do not alter this setting unless

	instructed to do so by Persistent Technical Support.
MGR_ID	A three-byte, hexadecimal identifier for the Configuration Server log file. This ID is used in the CSDB to identify Configuration Servers in a Distributed Configuration Server environment, and is passed to the RCA agent as the ZOBJMID variable. Valid values are any combination of 001 to EFF. Important Note: Do not alter this setting unless instructed to do so by Persistent Technical Support.
MGR_ NAME	Configuration Server name. Important Note: Do not alter this setting unless instructed to do so by Persistent Technical Support.
MGR_UUID	The Universal Unique Identification Number (UUID) of the Configuration Server. This 32-byte ID is created automatically during the installation and is used exclusively for licensing. Important Note: Do not change or delete this value.
OBJECTID_ FORMAT	This setting determines which format will be used for generating object IDs. Specify **1** to use the established algorithm. Specify **2** to use the new algorithm. If the value is absent or invalid, the default of **1** is assumed and the established algorithm is used. **Note:** For more information on this setting, see the section, "OBJECTID_ FORMAT" on page 78.
SHOW_ VERINFO	Displays version information in the Configuration Server log at startup.
TASK_ TYPE	Reserved for future development. Important Note: Do not alter this setting unless instructed to do so by Persistent Technical Support.
TCP_PORT	Port on which to listen for TCP/IP RCA agent connects. The default is **3464**. **Note:** This setting will be overridden by specifying a TCP/IP port in the MGR_ATTACH_LIST section. See "MGR_ATTACH_LIST" on page 33 for more details.
VERBOSE	Sends messages to the screen (stderr) when the Configuration Server starts up, verifies the license, verifies the CSDB, loads cache, and readies IP Managers (TCP and SSL) for processing and when shutting down.

Example

```
[MGR_STARTUP]
```

`BYTE_LEVEL_DIFF`	`= NO`
`MANAGER_TYPE`	`= DISTRIBUTED`
`MGR_ID`	`= 001`

`MGR_NAME`	`= RCS`
`MGR_UUID`	`= ssed111454d6kgh4eh7g3md90f94d6k8`
`OBJECTID_FORMAT`	`= 2`
`SHOW_VERINFO`	`= YES`
`TASK_TYPE`	`= THREAD`
`TCP_PORT`	`= 3464`
`VERBOSE`	`= NO`

MGR_STARTUP Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
ALLOW_ DUPLICATE _IP_ADDRESS	YES	NO	N/A	N/A
BYTE_LEVEL_ DIFF	N/A	NO	N/A	N/A
MANAGER_TYPE	DISTRIBUTED	DISTRIBUTED	N/A	N/A
MEMORY_TYPE	SHARED	SHARED	N/A	N/A
MGR_ID	As specified	001	000	EFF (F00-FFF are reserved)
MGR_NAME	As specified	EDM	N/A	N/A
MGR_UUID	N/A	N/A	N/A	N/A
OBJECTID_ FORMAT	2	1	N/A	N/A
SHOW_VERINFO	N/A	YES	N/A	N/A
TASK_TYPE	THREAD	THREAD	N/A	N/A
TCP_PORT	3464	1029	N/A	N/A
VERBOSE	N/A	NO	N/A	N/A

Performance and Usage Considerations

- If MANAGER_TYPE is set to STANDALONE, the Configuration Server will not be eligible for Distributed Configuration Server functionality. (See the section, "MANAGER_TYPE Values" on next page.)

- MGR_ID must be a three-character, alphanumeric, hexadecimal string. (Valid values are 001 to EFF.)
- If running the Configuration Server as a Windows Service, VERBOSE will be disabled.

OBJECTID_FORMAT

Note: A CSDB object's ID is a unique, 12-character, hexadecimal identifier that is automatically generated and assigned when a CSDB object is created.

The time-based format of object ID generation eliminates the possibility of randomly generating a duplicate object ID because each object ID begins with the letter A, B, C, E, or F—differing from the old format, which used only the letter D as a prefix.

The format of CSDB object IDs is A*mmmXXXXXXXX*, where:

A = the new prefix letter (A, B, C, E, or F)

mmm = the Configuration Server ID (see MGR_ID in "MGR_STARTUP" on page 75.)

XXXXXXXX = the unique value in hexadecimal format

Note: Specifying the object ID generation format affects new object ID generation only; it has no effect on existing object IDs.
When the time-based generator counter wraps, this format will automatically start prefixing the object IDs with the next sequential letter.

MANAGER_TYPE Values

The values for the MANAGER_TYPE settings are described in more detail below.

- **DISTRIBUTED**: The CSDB is read- and write-enabled and can be updated by various means; this CSDB can participate in RCA-DCS operations in which it can function as Source, Destination, or both.
  This is the MANAGER_TYPE **external default**.
- **SERVER**: The CSDB is read-only; it cannot be updated; to run, it requires that OBJECTID_FORMAT=2 and object ID cache be disabled; this CSDB can participate in RCA-DCS operations, but only as Destination or middle-tier.
- **STANDALONE**: The CSDB is read- and write-enabled for the Administrator and Configuration Server methods from RCA agent connects; this CSDB cannot participate in DCS operations.
- This is the MANAGER_TYPE **internal default**.

  Note: If the value of the MANAGER_TYPE setting is anything other than DISTRIBUTED or SERVER (including no value and invalid values), it will assume the MANAGER_TYPE internal default of STANDALONE.

MANAGER_TYPE=SERVER

If MANAGER_TYPE=SERVER then, by default, two subsequent conditions will exist:

- the OBJECTID_FORMAT=2 algorithm will be used for object ID generation, and
- object ID caching will be disabled.

For more information about how this setting affects DCS operations, refer to the *Radia Client Automation Enterprise Configuration Server Database Reference Guide*.

MGR_TASK_LIMIT

An RCA administrator can use this section of the edmprof file to specify the various settings that are related to Configuration Server tasks.

MGR_TASK_LIMIT Settings

Setting	Description
TASK_HEAP_SIZE	This setting controls the heap size of a task (used in conjunction with MGR_POOLS). Note: This setting is not applicable to the current release of the Configuration Server.
TASKLIM	The maximum number of RCA agent tasks allowed to concurrently run on the Configuration Server.
TASK_LOG_LIM	The maximum number of lines, per RCA agent, that can be written to the Configuration Server log.
TASK_RESO_LIM	The maximum number of resolutions allowed per RCA agent.
TASK_STACK_SIZE	This setting defines how many variables can be used per task.

Example

```
[MGR_TASK_LIMIT]
```

`TASK_HEAP_SIZE`	`= 0`
`TASKLIM`	`= 20`
`TASK_LOG_LIM`	`= 0`
`TASK_RESO_LIM`	`= 64000`
`TASK_STACK_SIZE`	`= 64000`

MGR_TASK_LIMIT Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
TASK_HEAP_SIZE	0	0	0	4 GB
TASKLIM	50	0	0	32 KB
TASK_LOG_LIM	0	100000 Lines	0	N/A
TASK_RESO_LIM	64000	64000 Resolutions	1	64000
TASK_STACK_SIZE	64000	64000	16384	64000

Caution: *Do not* change the TASK_STACK_SIZE setting in this section unless advised to do so by a member of Persistent Technical Support.

Performance and Usage Considerations

- The MGR_TASK_LIMIT settings should be coordinated with values in the MGR_NOTIFY and MGR_RETRY sections.
- TASK_STACK_SIZE will affect the depth of resolution. Higher values will result in deeper resolution.

MGR_TIMEOUT

This section specifies how long the Configuration Server will wait for a request from a connected RCA agent before disconnecting that RCA agent due to inactivity (no requests/responses from the RCA agent).

MGR_TIMEOUT Settings

Setting	Description
ADMIN_TIMEOUT	Timeout (in seconds) for administrator functions.
SEND_THROTTLE	Timeout (in milliseconds) before each send.
TIMEOUT_COMM	Communications (receive) timeout (in seconds).

Example

```
[MGR_TIMEOUT]
ADMIN_TIMEOUT = 0
SEND_THROTTLE = 0
TIMEOUT_COMM = 1800
```

MGR_TIMEOUT Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
ADMIN_ TIMEOUT	0 (never time out)	0 (never time out)	0	32767
SEND_ THROTTLE	N/A	0	0	4 GB
TIMEOUT_COMM	1800 (seconds)	0 (never time out)	0	32767

Performance and Usage Considerations

- If processing resources are critical, increase these MGR_TIMEOUT values.
- The MGR_TIMEOUT settings should be coordinated with values in the MGR_RETRY section.
- The SEND_THROTTLE value can be overridden by bandwidth throttling variables sent up in an RCA agent object.
- TIMEOUT_COMM is the Configuration Server analog to the RCA agent ZTIMEO. If a connect is active and the Configuration Server has not received any data from a RCA agent for the TIMEOUT_COMM value, the Configuration Server will terminate the session.

MGR_TPINIT

This section specifies packet sizes to send to RCA agents.

MGR_TPINIT Settings

Setting	Description
BUFTCP	TCP buffer size used for send/receive.
GET_ REMOTE _HOST_ NAME	Controls the Configuration Server's attempt to obtain the *host_name* from the DNS server. The default is **NO**. **Note:** Do not set to YES if the Configuration Server is not in a dynamic TCP/IP environment (such as DNS and DHCP).
MAXREC	Maximum record size.

Caution: *Do not* change any settings in this section unless advised to do so by a member of Persistent Technical Support.

Example

```
[MGR_TPINIT]
```

`BUFTCP`	`= 12288`
`GET_REMOTE_HOST_NAME`	`= NO`
`MAXREC`	`= 6144`

MGR_TPINIT Values

Setting	Value as Installed	Default Value
BUFTCP	12288	12288
GET_REMOTE_HOST_NAME	NO	NO
MAXREC	6144	6144

Performance and Usage Considerations

- The buffer size reflected in the MGR_TPINIT settings should be the same for the Configuration Server and RCA agents.
- Any buffer size setting in the MGR_TPINIT section should only be changed in coordination with equivalent changes to RCA agents and after having been directed to do so by a member of Persistent Technical Support.
- Do not use GET_REMOTE_HOST_NAME if the Configuration Server is not in a dynamic TCP/IP environment. This will cause the Configuration Server to expend unnecessary processing time attempting to associate the remote host name.
  If GET_REMOTE_HOST_NAME=YES, the Configuration Server obtains the remote host name using standard library calls. The IPNAME is received on the Configuration Server (via DNS) and stored in ZMASTER.ZIPNAME, which is stored in the appropriate domain of the PROFILE File. The remote host name will appear in each associated line of the Configuration Server log in place of the IP address.

MGR_TRACE

This section contains a list of keywords (settings) that you can specify to control and influence diagnostic logging for the Configuration Server. All diagnostic output produced by TRACE settings is written to the active Configuration Server log. To enable a TRACE keyword, specify YES. To de-activate a TRACE keyword, specify NO.

TRACE keywords specified in this section are invoked at Configuration Server initialization, and remain in effect:

- until they are changed by altering the MGR_TRACE setting and restarting the Configuration Server.
- while the Configuration Server is running, using the Console.
- until a specified REXX overrides the setting.
- unless a ZCVT value overrides the setting.

The trace settings that are in effect at Configuration Server initialization are displayed at the beginning of the log.

Note: The value for each setting is evaluated in the order in which it is presented in the `edmprof` file. The results can be non-intuitive. For example:

- If ALL=YES is the first setting specified, and each following settings are specified as NO, the effect is to turn off *tracing*.
- If ALL=YES is the last setting specified, all tracing will be active.
- If ALL=NO is the last setting, all tracing is turned off.

MGR_TRACE Settings

Setting	Description
ADMIN	Traces ADMIN transaction flow.
ADMPROM	Not used.
ALL	Turns on all other traces.
ALLOC	Traces file allocations.
AUDIT	Traces audit file activity.
BUFF	Traces data buffers (without transformation).
CMPR	Traces data compression.
COMM	Traces data stream buffers.
COMMCBS	Traces communications control block (CCB) activity.
COMMDATA	Traces data communications.
COMMRPLS	Traces communications control blocks (CCBS).
CONFIG	Traces configuration file activities.
DATA	Traces data buffers to or from the RCA agent.
DES	This setting is no longer used.
DMA	Traces Distributed Configuration Server activity.
ENQDEQ	Traces serialization activity (enqueues/dequeues).
EXPL	Traces data transformation (explode).
FILE	Traces file I/O.
IMPL	Traces data transformation (implode).
LOOKASID	Traces cache activity for classes/instances.
METHOD	Traces Configuration Server method execution/return codes.

NOTIFY	Traces notify processing.
OBJCRC	Traces object CRC processing.
OBJRES	Traces object resolution (very detailed).
OBJRES1	Traces object resolution (medium detail).
OBJRESO	Traces high-level object resolution flow (light detail).
OBJXFER	Traces object transfer.
PASSWORD	Traces passwords.
POOLMISS	Traces memory pool allocation.
PROFILE	Traces profile database activity.
PROMOTE	Traces file promotion.
RESOURCE	Traces resource file activity.
REXX	Traces REXX environment.
REXXOFF	Suppresses all REXX activity.
STORAGE	Traces storage in conjunction with the MGR_LOG's STORAGE_INTERVAL setting.
STATS	Traces statistics.
SUBST	Traces variable substitution.
TCP	Traces TCP/IP activity.
TEST	Reserved.
VAR	Traces the variable references.
VARSTG	Traces variable processing storage usage.
VARSUB	Traces variable substitution activity.
YEAR2000	Traces a database's Year-2000 compliance.

Example

```
[MGR_TRACE]
```

`ADMIN`	`= NO`
`ADMPROM`	`= NO`

ALLOC	= NO
AUDIT	= NO
BUFF	= NO
CMPR	= NO
COMM	= NO
COMMCBS	= NO
COMMDATA	= NO
COMMRPLS	= NO
CONFIG	= NO
DATA	= NO
DMA	= NO
ENQDEQ	= NO
EXPL	= NO
FILE	= NO
IMPL	= NO
LOOKASID	= NO
METHOD	= NO
NOTIFY	= NO
OBJCRC	= NO
OBJRES	= NO
OBJRES0	= NO
OBJRES1	= NO
OBJXFER	= NO
PASSWORD	= NO
POOLMISS	= NO
PROFILE	= NO
PROMOTE	= NO
RESOURCE	= NO
REXX	= NO

REXXOFF	= NO
STATS	= YES
STORAGE	= NO
SUBST	= NO
TCP	= NO
TEST	= NO
VAR	= NO
VARSTG	= NO
YEAR2000	= NO
ALL	= YES

Performance and Usage Considerations

- Turning on trace flags creates a large number of Configuration Server log messages. This will degrade the performance of the Configuration Server due to the disk I/O load. However, this might be necessary at times for problem resolution. Ensure that the Configuration Server log is properly configured.
- Tracing can be clustered to troubleshoot a particular aspect of Configuration Server operations, while simultaneously preserving an appropriate level of logging activity. For example, turning on flags, such as OBJCRC, OBJRES, OBJRES1, OBJRES0, and OBJXFER, can help identify problems that might be occurring during object resolution. Likewise, CPIC, DATA, and TCP focus on communications activities. (Note that some of these traces pertain to specific protocols.) Special purpose trace flags include DMA, NOTIFY, REXX, and METHOD.
- STORAGE=YES can be used to produce a message in the log each time a storage allocation is made and freed. This can be used to tune the pool and to understand the Configuration Server's use of dynamic memory.
- ALL=YES does not affect the REXXOFF trace settings.

MGR_USERLOG

This section specifies the logging directory and logging options for the user logging facility.

MGR_USERLOG Settings

Setting	Description
ACTIVATE	Activate user log at Configuration Server startup. Values are YES and NO.
DIRECTORY	Fully qualified directory path where the user log is written.
FLUSH_SIZE	The number of bytes between automatic flushes of operating system buffers for Configuration Server log file.

MESSAGE_ WIDTH	The maximum width (in bytes) of the messages in the user log.
PIPE_SIZE	The maximum amount (in bytes) of log messages that can be queued by the Configuration Server logging facility while the log file is busy. When this value is reached, any task that issues a log message will freeze until the pipe starts emptying.
THRESHOLD THRESHHOLD (both spellings accepted)	Maximum number of messages that will be written to a log before automatically switching to the next log. When the limit is reached, new log files are created. Specify a negative number to overwrite the log file when the limit is reached.

Windows Example

```
[MGR_USERLOG]
```

`ACTIVATE`	`= NO`
`DIRECTORY`	`= C:\Program Files\Hewlett-Packard\HPCA\ConfigurationServer\log`
`FLUSH_SIZE`	`= 256`
`MESSAGE_ WIDTH`	`= 256`
`PIPE_SIZE`	`= 1000000`
`THRESHOLD`	`= 5000000`

MGR_USERLOG Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
ACTIVATE	NO	NO	N/A	N/A
DIRECTORY	`/edmmgr/log`	*current_directory*	N/A	N/A
FLUSH_SIZE	256 bytes	100000 bytes	1	N/A
MESSAGE_WIDTH	256 bytes	90	80	N/A
PIPE_SIZE	1000000 bytes	65535 bytes	1	N/A
THRESHHOLD	-5000000 bytes	5000 bytes	1	N/A

Performance and Usage Considerations

- Increasing the FLUSH_SIZE will enhance performance, but will delay messages flushed to the log file.
- Increase MESSAGE_WIDTH if log messages are being truncated.

- When modifying parameters in this section as they relate to memory or disk use, be sure that the maximum amount of memory or storage space is available.

User Log Naming Conventions

The user log names conform to the following operating system specific conventions.

Note: ISO is the International Standards Organization.
Configuration Server ID is the value of the `edmprof` file's setting MGR_STARTUP.MGR_NAME.

Windows

- The *standard* log naming format is the user log prefix (nvd), followed the user designation (ur), and the Configuration Server ID:
  nvdur001.log
- The *log switch* format is similar to the standard but an **us** (rather than ur) is added to the prefix, and the ISO-formatted date and time (each preceded by an underscore) appended:
  `nvdus001_20050427_083357.log`
- The *log wrap* (dump) format is similar to the standard but sees the **r** being replaced by a **d**:
  `nvdud001.log`

OBJECT_SIZES

Note: The OBJECT_SIZES section must be manually added to the `edmprof` file. Additionally, this section must be added to the `edmprof` file in order for the Configuration Server self-tuning tool to operate properly.

This section accommodates specifying the number of heaps and the heap size for CSDB objects that are being created on the Configuration Server as in-storage CSDB objects.

- These values affect only the Configuration Server self-tuning tool; they have no effect on other Configuration Server processing.
- The format for specifying these values is:
  OBJECT_NAME = *heap_size*,*number_of_heaps*

For example:

```
ZCONTROL = 512,100000
```

In this example, whenever an object named ZCONTROL is created in the Configuration Server as an in-storage object, its initial allocation will be for 100,000 heaps of 512 bytes each.

```
ZERROR = 400,1000
```

In this example, whenever an object named ZERROR is created in the Configuration Server as an in-storage object, its initial allocation will be for 1000 heaps of 400 bytes each.

Note: For a detailed description of how these settings affect the Configuration Server self-tuning tool, see "Configuration Server Self-Tuning Tool" on page 98.

Example

```
[OBJECT_SIZES]

FILE = 1536,2000
RELEASE = 420,1536
WBEMAUDT = 6000,100
MSIFEATS = 512,100
ZOBJSTAT = 512,270
PRODUCT = 1024,200
ZREQDATA = 200,100
ASERVICE = 3000,60
ZSERVICE = 4000,60
ZREQNEWI = 150,15
MSIPROPS = 1024,30
ZERROR = 512,10
UMFLTCRI = 1536,23
WBEM = 1024,20
APPEVENT = 1024,5
SAPSTATS = 1024,10
SAP = 1024,10
CLISTATS = 512,5
UMFLTRUL = 600,4
DESKTOP = 2000,3
MSI = 2000,5
REGISTRY = 1536,5
PATH = 1024,3
ZCONFIG = 4000,1
ZMASTER = 3000,1
TIMER = 1536,2
```

RCS_TUNING_CONTROL

Note: The RCS_TUNING_CONTROL section must be manually added to the `edmprof` file. Additionally, this section must be added to the `edmprof` file in order for the Configuration Server self-tuning tool to operate properly.

This section provides a mechanism to override the default values that are specified in the Configuration Server self-tuning tool and these are described in the following table. (For a look at the Configuration Server self-tuning tool, see "Configuration Server Self-Tuning Tool" on page 98.)

RCS_TUNING_CONTROL Settings

Setting	Description
MAXIMUM_	A four-digit integer value that specifies the maximum number of MBs that are to be allocated to CONTENT CACHE.

	Note: This setting provides protection when the size of a class (or classes) would exceed the virtual storage of the process space. In the case that one class exceeds this value, that class is forced to be cached on first reference (=Y,N).
MINIMUM_ MEG_ CACHE	A four-digit integer value that specifies the minimum number of MBs that are to be allocated for CONTENT CACHE.
MAXIMUM_ SHARED _MEMORY_ SEGMENTS	A two-digit integer value that specifies the *maximum_number_of_segments* times the *largest_shared_memory_size*. **Note:** This value must be less than 15 because the value cannot exceed the size of the process space (generally, 2 GB [2*1024*1024*1024]).
MAXIMUM_ SHARED _MEMORY_ SIZE	A four-digit integer value that specifies the maximum number of MBs that are be allocated to any single CONTENT CACHE segment. **Note:** If the aggregate size of CONTENT CACHE exceeds this value, it should be allocated as multiple segments—each less than this value in size.
MAXIMUM_ CLASS_ INSTANCES	A ten-digit integer value that specifies the maximum number of instance elements that can exist in a class that might be cached at startup. **Note:** Any class with more instances will be marked (=Y,N) as “cache on first reference.”
MINIMUM_ INSTANCES	A five-digit integer value that specifies the minimum number of instance elements for which ICACHE and CONTENT CACHE are intended to be sized.
UPDATE_ RCS_ STARTUP	A YES-NO toggle for updating the startup of the Configuration Server. The default (**NO**) leaves the `edmprof` file unchanged and creates the file `EDMPROF_TUNED.DAT_`.

Example

```
[RCS_TUNING_CONTROL]

MAXIMUM_MEG_CACHE = 1500
MINIMUM_MEG_CACHE = 50
MAXIMUM_SHARED_MEMORY_SEGMENTS = 6
MAXIMUM_SHARED_MEMORY_SIZE = 32
MAXIMUM_CLASS_INSTANCES = 100000
MINIMUM_INSTANCES = 10000
UPDATE_RCS_STARTUP = YES
```

RCS_TUNING_CONTROL Values

Setting	Value as Installed	Default Value	Minimum Value	Maximum Value
MAXIMUM_MEG _CACHE	None	1500	None	None
MINIMUM_MEG _CACHE	None	200	None	None
MAXIMUM_SHARED _MEMORY_ SEGMENTS	None	6	None	None
MAXIMUM_SHARED _MEMORY_SIZE	None	384	None	None
MAXIMUM_CLASS _INSTANCES	None	100000	None	None
MINIMUM_ INSTANCES	None	50000	None	None
UPDATE_RCS_ STARTUP	None	NO	None	None

SECTION_DELIMITERS

This section is not a true section of the Configuration Server `edmprof` file. It is used only to specify the symbols that will be used to delimit section names in the `edmprof` file.

Caution: If used, SECTION_DELIMITERS must be the very first entry and the first non-blank line in the `edmprof` file. If it is not, the Configuration Server will not be able to read-in the license string and will not start up.

SECTION_DELIMITERS Settings

Setting	Description
SECTION_ DELIMITERS	The character set that is to be used to enclose the section headings of the `edmprof` file. The format is SECTION_DELIMITERS = *xy*, where *x* and *y* are the left and right delimiters. The options are: **[]**, **()**, **<>**, and **{}**.

Examples

```
SECTION_DELIMITERS = <>

<MGR_LICENSE>LICENSE_STRING = FFCDAB

SECTION_DELIMITERS = []

[MGR_LICENSE]LICENSE_STRING = FFCDAB
```

SECTION_DELIMITERS Values

Setting	Value as Installed	Default Value
SECTION_DELIMITERS	N/A	[ ]

Chapter 3

Managing Configuration Server Processing

Configuration Server Operations

Configuration Server operations has three basic phases:

- Startup
- Processing Requests
- Shutdown

Configuration Server startup is initiated by icon, command line, console, or control panel, depending on the platform and installation. In the Startup phase, the Configuration Server initializes ZTOPTASK. Using the Configuration Server `edmprof` file to provide the working parameters for configuring the Configuration Server, ZTOPTASK then starts the Task Manager. After the various tasks specified in MGR_ATTACH_LIST are enabled, the Configuration Server is ready to process requests.

A request can be sent to the Configuration Server as a system command, an RCA agent connect, an administrative transaction, or a Distributed Configuration Server command. During the Processing Requests phase, the Configuration Server performs the requests (tasks) that are submitted to it. There are four types of tasks.

- **System**
  These tasks pertain to Configuration Server functions.
- **Client**
  These tasks pertain to RCA agent requests.
- **Admin**
  These tasks pertain to RCA Admin CSDB Editor and Publisher operations.
- **Distributed Configuration Server**
  These tasks pertain to Distributed Configuration Server functions.

The type of task is shown in each line of the Configuration Server log, as shown in the following example.

```
NVD1069I 13:49:52 [208.244.225.166 /16B] Radia ClientMax size of local
memory allocated : 230805

NVD8115I 13:50:30 [ztoptask/17B] System TaskREXX Method
<D:\DEV\MGR\REXX\ZSHUTDWN> with parms <%s> started at <13:50:30:574>
```

Like startup, Configuration Server shutdown can be initiated in a number of ways. In the Shutdown phase, the Configuration Server performs some basic housekeeping, then essentially reverses the startup flow. The Configuration Server is now down, enabling you to run a backup, use the database utilities, apply maintenance, or perform other operating system tasks.

Customizing Configuration Server Processing

The values in your Configuration Server `edmprof` file enable you to customize its overall configuration. There are two main ways of customizing the flow of Configuration Server processing:

- Configuration Server REXX programs and
- Configuration Server methods.

The primary difference is that REXX programs are preconfigured, while the methods can be inserted anywhere in the processing flow.

This chapter describes the Configuration Server REXX programs and Configuration Server methods. An overview of the methods is presented in "Overview" on page 123. Also, "Configuration Server Methods" on page 202 details each method, providing an example of its use, a description, its associated parameters, and possible return codes.

Configuration Server REXX Programs

REXX Directories

The Configuration Server installation creates two REXX-related directories,

- *<InstallDir>*`\ConfigurationServer\rexx`
- *<InstallDir>*`\ConfigurationServer\rexx\NOVADIGM`

The `rexx` directory is empty, and the `rexx/NOVADIGM` (`rexx\NOVADIGM`) directory contains all the HP-related REXX programs.

> **Note:** The `ConfigurationServer/rexx` (`ConfigurationServer\rexx`) directory can be renamed to further distinguish it from the HP REXX directory.

Customizing the HP REXX Programs

The HP REXX programs can be customized to adapt to, and enhance, various computing environments. To customize any of these REXX programs, copy them from the `rexx\NOVADIGM` (`rexx/NOVADIGM`) directory to the `rexx` directory, then modify them as needed.

> **Caution:** Do not make any changes to the REXX programs in the `rexx\NOVADIGM` (`rexx/NOVADIGM`) directory.
> Doing so will adversely affect the performance of the CSDB.

There are two reasons that HP REXX programs have to be copied to the `rexx` directory before being modified:

- If a REXX is customized and left in the `\rexx\NOVADIGM` directory, the customizations could be lost (overwritten) if a database update is applied, thereby affecting the behavior and execution of the database operations.
- During processing, the database reads the `\rexx` directory first. Therefore, place any customized REXXs in that directory.

Event Points

There are eight event points at which the Configuration Server issues calls to ten major REXX programs. The table below lists these REXX programs and the points at which they are called.

Configuration Server REXX Programs

REXX Name	When Called (Event Point)
ZSTARTUP	Configuration Server Startup
ZPCACHE	Configuration Server Startup
ZINIT	Configuration Server Startup
ZTASKSTA	Task Start
ZTASKEND	Task End
ZNFYxSTA	Notify Start
ZNFYxEND	Notify End
ZLOGSWCH	Log Switch
ZLOGWRAP	Log Wrap
ZSHUTDOWN	Configuration Server Shutdown

REXX Programs

ZSTARTUP

This is called just before the Configuration Server is enabled to accept and process RCA agent connections. ZSTARTUP does not pass any parameters to the REXX, nor does it accept any information from this REXX. It can be used to perform user-defined specific processing before enabling connections to be initiated.

ZPCACHE

This can be called after cache is loaded during Configuration Server startup.

ZINIT

This is called during Configuration Server startup. It runs REXXs and tests for certain conditions. If the conditions are not met (return code = 16), Configuration Server startup will be halted.

Caution: This is not configurable. Consult Persistent Technical Support before using this REXX.

ZTASKSTA

This is called when each connection is first accepted. It is passed a single parameter that contains the protocol level identifier for the RCA agent.

ZTASKEND

This is called when each connection has ended, while storage and objects associated with the connection are still available. It is passed a single parameter that contains multiple subparameters that can be parsed and that are position dependent.

- Connect termination reason.
- User ID of the current connection.
- Total number of object instances, of any type, processed during this connection.
- Total number of object instances, of any type, resulting from resolution.
- The maximum depth of transient objects processed in any single instance resolution.
- Count of communications protocol sends and receives originating from this connection.
- Total number of bytes transmitted from the Configuration Server to the RCA agent during this connection (non-compressed count).
- Total number of files transferred from the Configuration Server to the RCA agent during this connection.

ZNFYxSTA

This is called at the beginning of Notify processing for each RCA agent that is being notified. The x defines the type of Notify (T for TCP/IP and D for Dial-up). The set of parameters passed includes:

- UINF=<value1>
  user info passed via EDMMPUSH method. If no information is provided, COMMON will be set as a value.
- RETRS=<value2>
  number of retries for this destination.
- STATUS=<value3>
  RETRY, SUCCESS, FAILURE.

- MSG=<value4>
  message describing the result of notification.

ZNFYxSTA can generate a return code that controls the execution of the Notify, RC value 1956 (RC_SKIP_NOTIFY). If this is set, it will cause the Notify request to be written for retry without execution of current notification.

ZNFYxEND

This is called at the end of Notify processing for each RCA agent being notified. The x defines the type of Notify (T for TCP/IP and D for Dial-up). The set of parameters passed includes:

UINF=<value1>
user info passed via EDMMPUSH method. If no information is provided, COMMON will be set as a value.

RETRS=<value2>
number of retries for this destination.

STATUS=<value3>
RETRY, SUCCESS, FAILURE.

MSG=<value4>
message describing the result of notification.

ZNFYxEND can generate a return code that controls the execution of the Notify, RC value 1955 (RC_NEVER_RETRY). If this is set, it will prevent the Notify request from being rescheduled for retry.

ZLOGSWCH

This can be called when log switch occurs (a new log file is created) to insert a user-defined command (for example, zip the old log file and save it).

ZLOGWRAP

This can be called when log wrap occurs (the log file is reused) to insert a user-defined command (for example, zip the old log file and save it).

ZSHUTDWN

This is called just before the Configuration Server shuts down.

Configuration Server Self-Tuning Tool

The Configuration Server self-tuning tool enables Configuration Server instances to automate the tuning of the `edmprof` file’s MGR_CACHE and MGR_CLASS sections.

Note: Two `edmprof` file sections, "OBJECT_SIZES" on page 88 and "RCS_TUNING_CONTROL" on page 89, are needed in order for this tool to function properly.

This tool does not dynamically alter the MGR_CACHE and MGR_CLASS sections while the Configuration Server is active; rather, while the Configuration Server is shutdown, it examines the database and creates an updated dataset that can be compared to the existing (master) `edmprof` file, and possibly used to automatically replace it.

The self-tuning tool can be configured to rename the master `edmprof` file and replace it with the newly created one, while this is not the default behavior (which is to overwrite the master `edmprof` file). If the non-default (rename-and-replace) option is selected, the only active `edmprof` file settings that should differ between the two `edmprof` files are those in the MGR_CACHE and MGR_CLASS sections (because these settings are generated at shutdown and are based on the size of the database and the number of instances).

The MGR_CLASS Section

If the master `edmprof` file does not have a MGR_CLASS section, the self-tuning tool attempts to content cache all the database instances. It does this by calculating the amount of space that will be necessary to support content caching all the instances, and by generating the caching directives for each class that is to be set for caching in its entirety at Configuration Server initiation and/or cache refresh.

If the MGR_CLASS section in the master `edmprof` file has caching directives for individual classes, they will be preserved. However, if these caching directives are intended for internally created objects, or are received by the Configuration Server from a session partner, they will not be preserved in the automatically generated values because the new MGR_CLASS section is based on the CSDB.

OBJECT_SIZES

The OBJECT_SIZES section will be preserved (as it was in the master `edmprof` file) in the new `edmprof` file, and will result in additional MGR_CLASS section entries, in the form of:

```
“NONDB.” OBJECT_NAME = N,N,heap_size,number_of_heaps
```

These entries will be listed first in the updated MGR_CLASS section.

They are followed by the MGR_CLASS entries from the master `edmprof` file, which are listed based on the Radia Client Automation processing preference in which priority is placed on processing the ZSERVICE and PACKAGE instances to facilitate the initial catalog resolution, followed by component instances, and then non-component instances.

Revise and Overwrite

The default behavior for the Configuration Server self-tuning tool is to create a new `edmprof` file in the directory that contains the master (or original) `edmprof` file. That format for naming the file is:

`EDMPROF_TUNED.DAT_REVISED_`*YYYYMMDD_HH_MM_SS_uuuuuu*

Where:

- `EDMPROF_TUNED.DAT_REVISED_` is a literal value,
- *YYYYMMDD* is the year, month, and date of the revise action, and
- *HH_MM_SS_uuuuuu* is the local hour, minute, second, and microsecond at which the new file is established.

Only one file matching the filter EDMPROF_TUNED.DAT_REVISED_* will be retained in the directory—that file will be the most recent one that was created by this tool.

Rename and Replace

If the non-default behavior of replacing the master `edmprof` file with the newly created one is selected, the `edmprof` file that existed when the Configuration Server was started will be renamed to:

`EDMPROF.DAT_REPLACED_`*YYYYMMDD_HH_MM_SS_uuuuuu*

Where:

- *YYYYMMDD* is the year, month, and date of the replace action, and
- *HH_MM_SS_uuuuuu* is the local hour, minute, second, and microsecond at which the file is replaced.

Up to 15 files matching the filter EDMPROF.DAT_REPLACED_* will be retained in the directory that contains the `edmprof` file.

Note: The 15 files are the 14 most recent datasets, and the oldest `edmprof` file (so that an original `edmprof` file is retained for reference).

All of these files will be in the directory in which the original `edmprof` file was found. Therefore, the account under which the Configuration Server is executing must have update and create authority for that directory.

The MGR_CACHE Section

In addition to the updated MGR_CLASS section, a new MGR_CACHE section gets created and replaces the section in the input `edmprof` file.

Note: While processing the original input `edmprof` file, the existing MGR_CLASS and MGR_CACHE sections are ignored, while all other sections are copied in their entirety. The newly created MGR_CACHE and MGR_CLASS sections will be appended to the end of the copied input `edmprof` file.
If, in the original `edmprof` file, there are comments preceding the MGR_CLASS and/or MGR_CACHE sections, then, in the newly created `edmprof` file, the comments might appear to be dislocated. However, since these are comments, their position does not effect the Configuration Server startup.

Reporting Files

A reporting file (`DB_EDMPROF_REPORT.TXT`) will be created in the Configuration Server log directory. It documents the MGR_CACHE and MGR_CLASS settings and the fact that the newly created values represent the actual amount of required storage. Only the most recent copy of each file will be retained, and each execution of the self-tuning tool will delete prior copies.

The self-tuning tool will attempt to allocate sufficient ICACHE and CONTENT CACHE to accommodate a database expansion of 30%. However, the results might change due to default sizes for minimum number of database instances, maximum amount of virtual storage to commit to content cache, and the effect of very large classes.

- A file named `EDMPROF_SECTION.DAT` is also created. It contains a replica of the MGR_CLASS and MGR_CACHE sections that were merged into the master `edmprof` file and replaced the existing MGR_CACHE and MGR_CLASS sections when the self-tuning tool started.
- A file (`LIST_PREFIX.CSV`) is created, but currently not used. When implemented, this will contain a snapshot of database domain and class information, including the size and number of instances in each class at shutdown.

To implement the Configuration Server self-tuning tool:

1. Install the new ZSHUTDWN REXX method into the `rexx` directory.

   Note: If a previous, customized version of ZSHUTDWN REXX exists in the rexx directory, be sure to copy any customizations from it to the new one. Then delete the previous one, or move it down to the `rexx\NOVADIGM` directory.

2. Shutdown the Configuration Server.
   (After the Configuration Server has shutdown, in the location that housed the `edmprof` file when the Configuration Server was started, there should be either: an updated `edmprof` file along with the original (but now renamed) `edmprof` file, or the revised `edmprof` file that overwrote the original.)
3. Restart the Configuration Server.

HP REXX Functions

Ten HP REXX functions perform specific actions. These actions encompass the getting and setting of objects and variables; object resolution; retrieving object names and properties; and UTF-8 and local code page strings.

Knowledge of REXX operators and word lists is assumed.

The REXX extensions are:

- EDMGET (objects)
- EDMGETV
- EDMSET (objects)
- EDMSETV
- EDMRESO
- NvdCurrentObjects
- NvdObjectInfo
- NvdObjectInfoEX
- NvdL2U
- NvdU2L

EDMGET

(*object,heapnumber*)

- *object* is the Configuration Server in-storage object that is to be processed.
  Object names are usually specified in uppercase; EDMGET will uppercase the value that is specified for *object*.
- *heapnumber* is the heap—of the above-referenced object—that is to be processed.
  If *heapnumber* is not specified, it defaults to zero (**0**), meaning the current heap. The *heapnumber* can range from one (**1**) to *the number heaps in the object*.

  Note: On the Configuration Server, the numbering of heaps starts at **1**, whereas on an RCA agent the numbering of heaps starts at **0**.

EDMGET is used to retrieve the contents of a CSDB object, one heap at a time, into a REXX's variable pool. In addition to returning a return code, EDMGET will set the REXX variables to contain the object's heap and variable count, the attribute names that are saved in the object, and their "as-is" and "resolved" ("indirect") values. The clause

```
rcode = EDMGET("zmaster")
```

will copy the contents of the first heap of the ZMASTER object into REXX variables. The REXX variable names that are created are in the following form.

```
object.attribute
```

```
object
```

```
objectvars
```

```
object.resolved.attribute
```

`object1` ... `objectn`

For example, to check the value of the ZUSER attribute in the ZMASTER object, use the REXX name

```
zmaster.zuser.
```

Note: For additional information on using REXX stem variables that are in this form, refer to the section on *Symbols* in the *Radia Client Automation Enterprise Application Manager and Configuration Server REXX Programming Guide*.

For example, the clause

`Say zmaster.zuserid` /* might return Fred */

- If the value of `zmaster.zuserid` was `&(object.attribute)`, its syntax is known as substitution, which is a form of value indirection.
- If the value of `zmaster.zuserid` was `&(SESSION.USERID)`, the USERID attribute in the SESSION object can be checked for the value. However, the SESSION object does not have to be opened to check the value because it is saved in the REXX variable `zmaster.resolved.zuser.`

If the value of `zmaster.zuserid` was not indirected, the value of `zmaster.resolved.zuserid` is the same.

In addition to the attributes in the object, two other REXX variables are created. The first has the same name as the object that is being processed; the second is the *objectname* with `vars` appended to the end of it.

Therefore, if the object that is being processed is ZMASTER, then:

`say zmaster` /* is the number of heaps in the object */

`say zmastervars` /* is the count of attributes in the object */

The variables <*object*>*1* ... <*object*>*n* contain the name of the attribute that is in the object. For example,

`say zmaster1` /* might be ZUSERID */

To get all the attributes in an object, code the following.

```
object = "zmaster"

heaps = value(object)

vars = value(object"vars")

vnames = ""

do vv = 1 to vars

vnames = vnames value(object|| vv)

end vv
```

If *heapnumber* is not specified, it defaults to zero (0), meaning the current heap. The heapnumber can range from one (1) to *the number heaps in the object*.

Note: On the Configuration Server, the numbering of heaps starts at 1, whereas on an RCA agent the numbering of heaps starts at 0.

The return value is:

- 0 if the object was copied to REXX storage
- Ø if it failed; the Configuration Server log will indicate the exact error.

EDMGET has a high cost to process a CSDB object, because each call to EDMGET will return all of the attributes in the object, thus creating all of the REXX variables that are noted above. If the intent (in the REXX code) is to get the value of some of the attributes that are defined in the object, then use EDMGETV, which is discussed in the next section

EDMGETV

(*object,attribute,heapnumber,resolve*)

- *object* is the CSDB object that is to be processed.
  The value that is specified will be automatically uppercased. Valid syntax for object names is 1

to 8 bytes; invalid names will generate a REXX syntax error.

- *attribute* is the attribute—of the above-referenced object—that is to be read.
  The value that is specified will be automatically uppercased. Valid syntax for attribute names is 1 to 8 bytes; invalid names will generate a REXX syntax error.
- *heapnumber* is the heap—of the above-referenced object—that is to be processed.
  The *heapnumber* can range from one (**1**) to *`the number heaps in the object`*.
  - If *heapnumber* is not specified, it defaults to zero (**0**), meaning the current heap.
  - If the value that is specified is out of range, a REXX syntax error is generated.
- `resolve` is a flag that can accept the values of **0**, **1**, **Y**, and **N**.
  - If `resolve` is not specified, it defaults to **0** (**N**), and the value that is returned is the contents of the object and attribute that are specified in arguments 1 and 2.
  - If **1** (**Y**) and the contents of the object and attribute that are specified in arguments 1 and 2 are in the form of `&(object.attribute)`, the value that is returned is an “indirect” value.
    If the contents are not in the form of `&(object.attribute)`, the contents are returned “as-is.”

    Note: For additional information on REXX operators, refer to the *Radia Client Automation Enterprise Application Manager and Configuration Server REXX Programming Guide*.

EDMGETV will return the null string (`""`) if either the object or the attribute does not exist.

Use the REXX “strictly equal” operator, consecutive equals signs (==), to check for the null string (`""`).

Note: For additional information on signal instruction, refer to the *Radia Client Automation Enterprise Application Manager and Configuration Server REXX Programming Guide*.

For example:

```
thisuser = edmgetv("zmaster", "zuserid") /* such as FRED */

this_data = edmgetv("zmaster", "mydata")

if "" == this_data

then say "The attribute mydata does not exist"

else say "The attribute mydata is" this_data
```

EDMSET

(*object,heapnumber*)

EDMSET is used to migrate the value of REXX variables into the specified object at the specified heap. Attributes that are not defined in the object are added; existing attributes are updated. To add a new heap to an existing object, set the value of *heapnumber* to one more than the current number of heaps in the object (that is, *the current number heaps in the object* + 1). The variables that are migrated to the object are in the following form.

object.attribute

The process scans all of the currently allocated (REXX) variables, looking for names that start with the value that is specified for *object*, such as **zmaster**. When a match is found, it is checked to determine whether the second part of the argument (*attribute*) conforms to an attribute name (1 to 8 bytes in length).

If so, it is checked to determine whether it fits the size restrictions (no more than 255 bytes) of an attribute.

If it passes this check, the REXX variable is migrated to the object.

- If either of these format checks fails, the REXX variable is skipped.
  The return value is:
  - 0 if no errors were encountered during the migration.
  - Ø if errors were generated; the Configuration Server log will indicate the exact error.

As noted with EDMGET, EDMSET is a costly process to save one or two attributes. For this processing, it would be better to use EDMSETV, which is discussed in the next section.

EDMSETV

(*object,attribute,value,heapnumber*)

- *object* is the CSDB object that is to be processed.
  The value that is specified must be from 1 to 8 bytes in length; this value will be automatically uppercased.
- *attribute* is the attribute—of the above-referenced object—that is to be written.
  The value that is specified must be from 1 to 8 bytes in length; this value will be automatically uppercased.
- *value* is the text to be saved to the attribute.
- *heapnumber* is the heap—of the above-referenced object—that is to be processed.
  If *heapnumber* is not specified, it defaults to zero (0), meaning the current heap. The *heapnumber* can range from one (1) to *the number heaps in the object*.
  If the specified arguments do not conform to the syntax, a REXX syntax error is generated with the error text being written to the Configuration Server log.

If EDMSETV completes (without a syntax error), a return value of:

- 0 indicates that the value was saved in the specified object/attribute.
- Ø indicates that it failed; the Configuration Server log will indicate the exact error.

EDMRESO

EDMRESO runs a “secondary” resolution on the instance name that is specified by

```
rcode = EDMRESO(f.d.c.i).
```

Note: The `f.d.c.i` indicates the CSDB tree structure, FILE.DOMAIN.CLASS.INSTANCE.

EDMRESO status is indicated by return codes of:

- 0 indicates that the process started
- Ø indicates that the process did not start; check the Configuration Server log for details.

Configuration Server Database Object and non-ASCII Object Names

For REXX variables to be created by EDMGET, their names need to consist of characters that are valid REXX variable names. (See the *Radia Client Automation Enterprise Application Manager and Configuration Server REXX Programming Guide*) Such is not the case for CSDB object names that consist of multi-byte UTF-8 sequences.

Objects can still be processed via EDMGETV and EDMSETV because all of the arguments to these functions are (REXX) strings, as in:

say EDMGETV("zmaster", "zos")

All object processing can be done with EDMGETV and EDMSETV, but the numbers of heaps and attributes, which are set by the EDMGET call, would not be known.

Additional Functions

Three additional REXX functions retrieve object names and properties.

- "NvdCurrentObjects" below
  returns information pertaining to the current Configuration Server session.
- "NvdObjectInfo" on next page
  returns information pertaining to heap counts and attributes.
- "NvdObjectInfoEX" on next page
  is similar to NvdObjectInfo but includes attribute-property information.

These functions are valid only on Windows operating systems.

NvdCurrentObjects

`Objectlist = NvdCurrentObjects()` /* no arguments */

NvdCurrentObjects returns a list of words that contains the names of the objects that exist in the current Configuration Server session. So,

```
AllObjects = NvdCurrentObjects()
```

returns a list of all object names that are allocated in the current Configuration Server-RCA agent connection. Use this information to check the availability of, for example, ZMASTER, as in:

```
Objects = NvdCurrentObjects()

If wordpos("ZMASTER", Objects) > 0  /* Note that object names are case sensitive */

then say "zmaster exists"

else say "Can’t find ZMASTER"
```

NvdObjectInfo

`Infolist = NvdObjectInfo(`*object*`)`

object is the CSDB object that is to be processed.

The value that is specified will be automatically uppercased. The name that is specified has to be from 1 to 8 bytes; invalid names will generate a REXX syntax error.

Returns

NvdObjectInfo returns a list of words in which

- the first word is the heap count for the specified object,
- the second word is the total size of all the attributes in the specified object, and
- the rest of the list contains the names of the attributes that are in the specified object.

> **Note:** Encrypted attributes are not returned.

So, the above-mentioned information can be determined by using the following REXX code.

```
Object = "zmaster"

Parse value NvdObjectInfo(Object) With Heaps TotalSize AttributeNames

AttributeCount = words(AttributeNames)
```

If there are any errors, only the heap count is returned, and is so as a negative number.

```
if heaps > 0

then nop

else exit 16
```

To get the attribute names:

```
Do nn = 1 to AttributeCount

say word(AttributeNames, nn)

End nn
```

NvdObjectInfoEX

`Infolist = NvdObjectInfoEX(`*object*`)`

- *object* is the CSDB object that is to be processed.
  The value that is specified will be automatically uppercased. The name that is specified has to be from 1 to 8 bytes; invalid names will generate a REXX syntax error.

Returns

NvdObjectInfoEX returns “property” information for each of the attributes that is returned—including encrypted attributes.

`ObjectInfo = NvdObjectInfoEX(`*objectname*`)`

The property information is returned in the following form, with the last four values returned as numerals.

attributename:size:mflags:cflags:vtype

- *size*
  “size” is a decimal numeric value from 1 to 255
- *mflags*
  “manager flags” is a decimal numeric value from 0 to 255.
- *cflags*
  “client flags” is a decimal numeric value from 0 to 255.
- *vtype*
  “attribute type” is a single ASCII character that describes the attribute “type.” Valid values are **C** (connection), **V** (variable), and **M** (method).
  This is a decimal numeric value, so **V** would be specified as **86**.

To access this information, use the following REXX code.

```
parse value nvdobjectinfoex("zmaster") with heaps totalsize attributes

do aa = 1 to words(attributes)

parse value word(attributes, aa) with attribute ":", maxlen ":" mflags
":" cflags ":" vtype .

wstr = d2c(vtype)

if datatype(wstr, 'm')

then vtype = wstr

else vtype = d2x(vtype)

mflags = x2b( d2x(mflags, 2))

cflags = x2b( d2x(mflags, 2))

say attribute mflags cflags vtype

end aa
```

Lastly, two utility functions to convert between local code page (LCP) and UTF-8 strings.

NvdL2U

`UTF8Value = NvdL2U(`*LCPVALUE*`)`

NvdU2L

```
LCPValue = NvdU2L(UTF8Value)
```

The REXX functions **stream**, **linein**, **lineout**, **charin**, **charout**, and **lines** reference external files. The filename and path that are specified must be in LCP. These functions can be used to convert the value. For example, to read a file that has been saved in the Configuration Server's method path, use:

```
wstr = edmgetv("ZCVT","MTHPATH")

if right(wstr, 1) = rxxospathseparator()

then nop

else wstr = wstr || rxxospathseparator()

mydata = nvdu2l(wstr || "mydata.file")

info = stream(mydata, 'c', 'query exists')
```

In this case, *mydata* will contain LCP characters that are the value needed for the REXX stream function.

ZCVT and ZTCBG

These two objects assist you in determining current values for CSDB tasks and connects, whether they are running. ZCVT performs a global function—determining values of tasks and connects for the entire CSDB, whereas ZTCBG determines the values that are relevant to the specified task or connect.

The ZCVT and ZTCBG objects (`objname`) work with the EDMGET, EDMGETV, EDMSET, and EDMSETV REXX extensions, as described in the previous section.

"ZCVT Table of Variables" below and "ZTCBG Table of Variables" on page 119 present the variables associated with ZCVT and ZTCBG, respectively.

ZCVT Table of Variables

ZCVT Variables

Variable Name	Variable Type	Description
VERMAJ	USHORT	Configuration Server major version number
VERMIN	USHORT	Configuration Server minor version number
VERREVNO	USHORT	Configuration Server revision number
VERREVLE	UCHAR	Configuration Server revision letter

VERBLDNO	ULONG	Configuration Server build number
OSNAME	STR	Configuration Server's operating system
TOLOGONS	LONG	Total number of logons to the Configuration Server
TRCOMMCB	FLAG	COMM Trace value
TRCOMDAT	FLAG	COMM trace value
TRDSCOMP	FLAG	DSCOMP trace value
TRTEST	FLAG	TEST trace value
TRDYNALO	FLAG	ALLOC trace value
TRVARSTG	FLAG	VARSTG trace value
TRDBCB	FLAG	ODBCBS trace value
TRDBDATA	FLAG	ODBDATA trace value
TRAUDIT	FLAG	AUDIT trace value
TRPROFIL	FLAG	PROFLE trace value
TRRESRCE	FLAG	RESOURCE trace value
TRPROMOT	FLAG	PROMOTE trace value
TRCONFIG	FLAG	CONFIG trace value
TRMETHOD	FLAG	METHOD trace value
TRCPIC	FLAG	CPIC trace value
TRADMIN	FLAG	ADMIN trace value
TRRESLV0	FLAG	OBJRES0 trace value
TRBINDFL	FLAG	DATA trace value
TRDAXFRM	FLAG	TRAN trace value
TR3270BU	FLAG	BUFF trace value
TRCOMM	FLAG	COMM trace value
TRFILPRO	FLAG	FILE trace value
TROBJXFR	FLAG	OBJXFER trace value
TROBJCRC	FLAG	OBJCRC trace value
TRREXX	FLAG	REXX trace value
TRVARS	FLAG	VAR trace value

TRSUBST	FLAG	SUBST trace value
TRDESENC	FLAG	DES trace value
TRCOMPR	FLAG	CMPR trace value
TROBJRES	FLAG	OBJRES trace value
TRIMPLOD	FLAG	IMPL trace value
TREXPLOD	FLAG	EXPL trace value
TRLASIDE	FLAG	LOOKASID trace value
TRENQUE	FLAG	ENQDEQ trace value
TRSTATS	FLAG	STATS trace value
TRRESLV1	FLAG	OBJRES1 trace value
TRTCPIP	FLAG	TCP trace value
TRADMPRM	FLAG	ADMPROM trace value
TRNOTIFY	FLAG	NOTIFY trace value
TRSESBLK	FLAG	SESSBLK trace value
TRSTORAG	FLAG	Storage trace value
TRY2K	FLAG	YEAR2000 trace value
TRDMA	FLAG	DMA trace value
TRVSAPI	FLAG	VSAM trace value
TRVSCB	FLAG	VSAMRPL trace value
TRVSDATA	FLAG	VSAMDATA trace value
TRREXOFF	FLAG	REXXOFF trace value
SHTINDIC	UCHAR	Shutdown indicator
SHTLGMGR	UCHAR	Log Configuration Server shutdown indicator
REXALLOC	LONG	REXX allocate count
TOPTSKID	LONG	ZTOPTASK ID
LOGMGRID	LONG	ZLOGMGR ID
ULGMGRID	LONG	ZULOGMGR ID
CLKMGRID	LONG	ZCLKMGR ID
TSKMGRID	LONG	ZTASKMGR ID

MGRTYPE	UCHAR	MANAGER_TYPE value
TASKTYPE	UCHAR	TASK_TYPE value
MEMTYPE	UCHAR	MEMORY_TYPE value
DBLUDATE	STR	Date of last CSDB update
DBLUTIME	STR	Time of last CSDB update
DBLCKCNT	USHORT	CSDB lock count
DBSTATUS	UCHAR	CSDB status
BYTELEVD	UCHAR	Byte level differencing on value
DNYDUPIP	FLAG	ALLOW_DUPLICATE_IP_ADDRESS value
TRUSTEDP	FLAG	Authorized trusted partner value
TSOSRVCE	FLAG	TSO service value
LICERROR	FLAG	License violation error value
ACALLADM	FLAG	ALWAYS_CALL_ZADMIN value
OKDUPINS	FLAG	ALLOW_DUPLICATE_INSTANCES value
QUITASKS	FLAG	Quiesce task
QUITRANS	FLAG	Quiesce transaction
LALLTCP	FLAG	TCP/IP logons allowed value
REXDISAB	FLAG	REXX off
MODNAMLO	FLAG	SHOW_MODULE value
MODVERLO	FLAG	SHOW_VERINFO value
TCPRHNAM	FLAG	GET_REMOTE_HOST_NAME value
HISTORY	UCHAR	History file value
ACCESSA	UCHAR	MGR_ACCESS ADMIN value
ACCESSC	UCHAR	MGR_ACCESS CONSOLE value
ZTIME	STR	Time in HH:MM
ZTIME24	STR	Time in HH:MM on 24-hour clock
ZAMPM	STR	AM or PM value
ZDATE	STR	MM/DD/YYYY date form
ZDATEDMY	STR	DD/MM/YYY date form

ZMONTH	STR	Short value for month (e.g., Jan)
ZMONTHLNG	STR	Long value for month (e.g., January)
ZDATEJUL	STR	Julian date
ZDATEYMD	STR	YYYY/MM/DD form for sorting
ZDAT2YMD	STR	YYYY/MM/DD form for sorting – full MM, DD value
MGRID	STR	Configuration Server ID (MGR_ID)
MGRNAME	STR	Configuration Server name (MGR_NAME)
DOMAIN	STR	Configuration Server domain name (DOMAIN)
STDOMANR	STR	RCA agent start domain (START_DOMAIN)
STCLASSR	STR	RCA agent start class (START_CLASS)
STDOMANA	STR	ATM start domain (START_DOMAIN)
STCLASSA	STR	ATM start class (START_CLASS)
DBASE	STR	CSDB used at startup (DBASE)
AGENT	STR	Agent ACB (AGENT)
TCPPORT	STR	TCP PORT value (TCP_PORT)
TCPUSRID	STR	USERID of TCP/IP address space
TORESO	LONG	Number of object resolutions
DEEPRESO	LONG	Deepest object resolution
TOOBJI	LONG	Number of inbound objects
TOOBJO	LONG	Number of outbound objects
ZERMXWRN	SHORT	Max number of warnings heaps
ZERMXERR	SHORT	Max number of error heaps
TOCOMP	LONG	Number of times compression done
TOCOMPI	LONG	Compression total bytes in
TOCOMPO	LONG	Compression total bytes out
TODCMPI	LONG	Decompression total bytes in
TODCMPO	LONG	Decompression total bytes out
TODCMP	LONG	Number to times decompression done
COMPSEED	LONG	Compression seed

TODBGETS	LONG	Number of GET operations
TODBPUTS	LONG	Number of PUT operations
TODBADDS	LONG	Number of ADD operations
TODBDELE	LONG	Number of DELETE operations
TOFILEIO	LONG	Global file i/o counts
TOFALLOC	LONG	Global file allocations
TSKLIMAX	SHORT	Max TASKLIM
TSKLIMIT	SHORT	TASKLIM (TASKLIM)
TSKLHARD	SHORT	HARD TASKLIM (TASKLIM_HARD) calculated as ALSLOTS + 5 + TSKLIMIT + (TSKLIMIT/3)
TSKLSOFT	SHORT	SOFT TASKLIM calculated as TSKLIMIT + ALSLOTS
TSKLDLTA	SHORT	TASKLIM Delta calculated as (TSKLSOFT - TSKLIMIT) + 1
MAXRESCL	LONG	Max number of resource per RCA agent
MAXRESAL	LONG	Max number of resource all RCA agents
TOMETBIN	LONG	Number of methods run – ASM and C
TOMETREX	LONG	Number of methods run – REXX
TOXNRCVD	LONG	Number of transaction received
TOXNRJCT	LONG	Number of transaction rejected
PLSTATUS	PTR	Pointer to disable Pools heap
PLGLBHEP	FLAG	Global/local pools in used flag
PLCONTIG	FLAG	Pools are allocated contiguous flag
PLEXPSIZ	ULONG	Pool expansion size
PLCSCRED	ULONG	Current storage credit
PLPSGUAR	ULONG	Percent Storage Guard
PLCSCUSH	ULONG	Current storage cushion
PLMSCUSH	ULONG	Minimum storage cushion
PLPOLHWM	ULONG	Largest polar bytes used
TSKPRIV	ULONG	Task private size
TSKSTSIZ	ULONG	Task stack size
TSKSTHWM	ULONG	Task stack size overuse HWM

TSKHPSIZ	ULONG	Task heap size
TSKHPHWM	ULONG	Task heap size overuse HWM
TIMEOCOM	LONG	TIMEOUT_COMM
TIMEONCM	LONG	TIMEOUT_NCOMM
TIMEOADM	LONG	ADMIN_TIMEOUT
TIMEODMA	LONG	DMA_TIMEOUT
RETRYBUS	LONG	BUSY_RETRY
RETRYDIS	LONG	DISABLE_RETRY
DSCOMPI	LONG	Data stream compression total bytes in
DSCOMPO	LONG	Data stream compression total bytes out
DSCOMPT	LONG	Number of times data stream compression done
SNDTHRTL	ULONG	SEND_THROTTLE
TIMEONFT	SHORT	NFYT_TIMEOUT
TIMEONFS	SHORT	NFYS_TIMEOUT
TIMEONFD	SHORT	NFYD_TIMEOUT
OBJNMASK	STR	Object name mask for traces
VARNMASK	STR	Variable name mask for trace
LOGLNTSK	ULONG	TASK_LOG_LIM
MAXRSTSK	ULONG	TASK_RESO_LIM
MAXREC	LONG	MAXREC
BUFTCP	LONG	BUFTCP
LOGDIR	STR	DIRECTORY
LOGTHRES	LONG	THRESHOLD
LOGLNCNT	LONG	Log file line count
LOGSTINT	LONG	STORAGE_INTERVAL
LOGFSIZE	LONG	FLUSH_SIZE
LOGMWIDT	USHORT	MESSAGE_WIDTH
LOGMPREF	STR	MESSAGE_PREFIX
LOGMDATE	UCHAR	MESSAGE_DATE

LOGMLDEL	UCHAR	MESSAGE_DELIMITER
LOGMRDEL	UCHAR	MESSAGE_DELIMITER
LOGPSIZE	LONG	PIPE_SIZE
LOGFLUSH	FLAG	Log flush
LOGSWITC	FLAG	Log switch
LOGELOFF	FLAG	DISABLE_NT_EVENT_LOGGING
LOGSLOFF	FLAG	DISABLE_SNMP_TRAP_LOGGING
LOGSFREQ	STR	SWITCH_TOD
LOGFFREQ	STR	FLUSH_INTERVAL
LOGBPIPE	LONG	Log bytes in pipe
ULGDIR	STR	USERLOG DIRECTORY
ULGTHRES	LONG	USERLOG THRESHOLD
ULGLNCNT	LONG	USERLOG LINECOUNT
ULGFSIZE	LONG	USERLOG FLUSH_SIZE
ULGMWIDT	USHORT	USERLOG MESSAGE_WIDTH
ULGACTIV	FLAG	USERLOG ACTIVATE
ULGPSIZE	LONG	USERLOG PIPE_SIZE
ULGFLUSH	FLAG	USERLOG flush
ULGSWITC	FLAG	USERLOG switch
SIMTSKPC	LONG	Simulation TASKS_PER_CONNECT
STATPATH	STR	Stats path
STATINTV	LONG	Stats interval
MTHPATH	STR	METHOD_PATH
MTHMLIMI	LONG	LOG_LIMIT
MTHTIMEO	LONG	TIMEOUT
DBPATH	STR	DBPATH
DBPPRIM	STR	PRIMARY DB path
DBPSECO	STR	SECONDARY DB path
DBPPROF	STR	PROFILE DB path

DBPRESO	STR	RESOURCE DB path
DBPHIST	STR	HISTORY DB path
DBPNOTI	STR	NOTIFY DB path
REXXPATH	STR	REXX_PATH
SYSPATH	STR	System path
DMASPATH	STR	DMA_STAGE_PATH
MGRSETFI	STR	PROFILE path
EXPTPATH	STR	EXPORT_PATH
USRPATH1	STR	USER_PATH1
USRPATH2	STR	USER_PATH2
USRPATH3	STR	USER_PATH3
USRPATH4	STR	USER_PATH4
USRPATH5	STR	USER_PATH5
USZTCBGS	SHORT	Number of used ZTCBGs
CACSEGS	USHORT	CACHE_SEGMENTS
CACSIZE	ULONG	CACHE_SIZE
CACMAXE	ULONG	Max cache entries
CACSTATS	USHORT	CACHE_STATS
CACFULL	USHORT	CACHE full
CACCLOSE	USHORT	CACHE closed
ICASIZE	ULONG	ICACHE_SIZE
ICACLOSE	UCHAR	ICACHE closed
SMMAILDR	STR	MAIL_DIR
SMMGRMID	STR	MGR_MAIL_ID
SMLOCHST	STR	Local host NAME
SMDNSSRV	STR	DNS_SERVER
SMSMTPRT	USHORT	SMTP_PORT

SMSPLCNT	ULONG	Spooled mail count
SMTIMEO	USHORT	MAIL_TIMEOUT
SMMAXSPL	USHORT	MAX_TIME_IN_SPOOL
SMRETRYI	USHORT	RETRY_INTERVAL
SNPORT	USHORT	SNMP_PORT
SNLOGPRT	USHORT	SNMP_LOGGER_PORT
SNMGRPRT	USHORT	SNMP_MANAGER_PORT
SNRUNEXT	FLAG	RUN_AS_EXTENSION
SNIPADD	STR	SNMP_IP_ADDR
SNMIPAD	STR	SNMP_MANAGER_IP_ADDR
SNMIPAD2	STR	SNMP_MANAGER_IP_ADDR2
SNMIPAD3	STR	SNMP_MANAGER_IP_ADDR3
SNCMNT	STR	SNMP_COMMUNITY
SNSTCMNT	STR	SNMP_SET_COMMUNITY
SNZERSEV	SHORT	SNMP_ZERROR_SEVERITY
ALSLOTS	LONG	ATTACH_LIST_SLOTS
ALVINTVL	LONG	VERIFY_INTERVAL
ALRLIMIT	LONG	RESTART_LIMIT
DMSECMTH	STR	SECURITY_METHOD
DIAINTVL	ULONG	DIAGNOSTIC_INTERVAL
DIADBYTE	ULONG	DIAGNOSTIC_MIN_DB_BYTES
DIALBYTE	ULONG	DIAGNOSTIC_MIN_LOG_BYTES
DBVERIFY	STR	VERIFY_DEPTH
DBAUTOFIX	FLAG	DB_AUTOFIX
METHDLLS	FLAG	Run methods as DLLS

ACTSKMON	ULONG	Number of monitors
ACTSKCON	ULONG	Number of consoles
ERREMAIL	STR	UserEmailErrorsTo
DBEEMAIL	FLAG	DBERROR e-mail
DBESHTDN	FLAG	DBERROR SHUTDOWN
DBESNMP	FLAG	DBERROR SNMP
POLCYSVR	STR	Status of the Policy Server (enabled/disabled)
RIM	STR	Status of the Inventory Manager (enabled/disabled)
RMP	STR	Status of the Portal (enabled/disabled)

ZTCBG Table of Variables

ZTCBG Variables

Variable Name	Variable Type	Description
AUDFLAG	FLAG	Audit trace flag
TSKNAME	STR	Name of this task or RCA agent
TSKSTTIM	STR	Time when task started
TSKSTDAT	STR	Date when task started
TSKLSTCO	STR	Time of last communication transaction
FREEMAIN	FLAG	Return storage on last free call
SHORTSTO	FLAG	Retry due to Short on storage
ZTERMINI	FLAG	Terminal task initiated
STOCRCUR	ULONG	Current storage credit
STOCRHEP	ULONG	Heap credit
STOCRSTK	ULONG	Stack credit
STOCRPVT	UONG	Private credit
STOCRPOO	ULONG	Zpools credit
STOCRCUS	ULONG	Credit cushion
MTHNAME	STR	Child name
MTHLIBNA	STR	Method library name
MTHLIBHA	ULONG	Method library handle

MTHLIBEN	ULONG	Method library entry point
MTHTHPRM	PTR	Method thread parameter pointer
TASKID	ULONG	Unique ID of this task
TASKPAR	ULONG	Unique ID of the parent task
USERID	STR	Task’s user ID
TASKTYPE	STR	TASK TYPE
DEEPRESO	LONG	DEEPSET OBJECT RESOLUTION
OBJSRESO	LONG	NUMBER OF OBJECTS RECEIVED
OBJSRECV	LONG	NUMBER OF OBJECTS RECEIVED
OBJSSENT	LONG	NUMBER OF OBJECTS SENT
TRCOMDAT	FLAG	COMM TRACE FLAG
TRDSCOMP	FLAG	DSCOMP TRACE FLAG
TRTEST	FLAG	TEST TRACE FLAG
TRDYNALO	FLAG	ALLOC TRACE FLAG
TRVARSTG	FLAG	VARSTG TRACE FLAG
TRAUDIT	FLAG	AUDIT TRACE FLAG
TRPROFIL	FLAG	PROFILE TRACE FLAG
TRRESRCE	FLAG	RESOURCE TRACE FLAG
TRPROMOT	FLAG	PROMOTE TRACE FLAG
TRCONFIG	FLAG	CONFIG TRACE FLAG
TRMETHOD	FLAG	METHOD TRACE FLAG
TRCPIC	FLAG	CPIC TRACE FLAG
TRADMIN	FLAG	ADMIN TRACE FLAG
TRRESLVL	FLAG	OBJRES0 TRACE FLAG
TRBINDFL	FLAG	DATA TRACE FLAG
TRDAXFRM	FLAG	TRAN TRACE FLAG
TR3270BU	FLAG	BUFF TRACE FLAG
TRCOMM	FLAG	COMM TRACE FLAG
TRFILPRO	FLAG	FILE TRACE FLAG

TROBJXFR	FLAG	OBJXFER TRACE FLAG
TROBJCRC	FLAG	OBJCRC TRACE FLAG
TRREXX	FLAG	REXX TRACE FLAG
TRVARS	FLAG	VAR TRACE FLAG
TRSUBST	FLAG	SUBST TRACE FLAG
TRDESENC	FLAG	DES TRACE FLAG
TRCOMP	FLAG	CMPR TRACE FLAG
TROBJRES	FLAG	OBJRES TRACE FLAG
TRIMPLOD	FLAG	IMPL TRACE FLAG
TREXPLOD	FLAG	EXPL TRACE FLAG
TRLASIDE	FLAG	LOOKASID TRACE FLAG
TRENQUE	FLAG	ENQDEQ TRACE FLAG
TRSTATS	FLAG	STATS FLAG
TRRESOL1	FLAG	OJBRES1 TRACE FLAG
TRTCPIP	FLAG	TCP TRACE FLAG
TRADMPRM	FLAG	ADMPROM TRACE FLAG
TRNOTIFY	FLAG	NOTIFY TRACE FLAG
TRSESBLK	FLAG	SESSBLK TRACE FLAG
TRSTORAG	FLAG	STORAGE TRACE FLAG
TRY2K	FLAG	YEAR2000 TRACE FLAG
TRDMA	FLAG	DMA TRACE FLAG
TRVSAPI	FLAG	VSAM TRACE FLAG
TRVSCB	FLAG	VSAMRPLS TRACE FLAG
TRVSDATA	FLAG	VSAMDATA TRACE FLAG
TRREXOFF	FLAG	REXXOFF TRACE FLAG
STBBSENT	FLAG	BB SENT FLAG
STNOSNAP	FLAG	NO SNAP FLAG
STCOWAIT	FLAG	WAIT FOR COMM OP FLAG
STPWDVER	FLAG	EDATS SF ORDERS OK FLAG

STTIMOUT	FLAG	TIMEOUT IN PROGRESS FLAG
STFORTER	FLAG	FORCED TREMINATION FLAG
STPARSES	FLAG	PAR SESS PARTNER FLAG
STSESEST	FLAG	SESSION ESTABLISHED FLAG
STSESTER	FLAG	SESSION TERMINATED FLAG
STSESLST	FLAG	SESSION LOST FLAG
STTSKABN	FLAG	TASK ABENDED FLAG
STSESSND	FLAG	SEND FLAG
STNOPDS	FLAG	NO PARSE R/S DATA STREAM FLAG
STTSKINA	FLAG	TASK BEING INACTIVATED FLAG
STTIMSND	FLAG	TIME SENT FLAG
STNODSCO	FLAG	NO DS COMPRESSION FLAG
STABTRES	FLAG	ABORT OBJECT RESOLUTION FLAG
STABTLEG	FLAG	ABORT OBJECT RESOLUTION FLAG
STSTRLOG	FLAG	IN LOGGER FLAG
STEOT	FLAG	EOT FLAG
STNOSUB	FLAG	NO SUBSTITUTION FLAG
STDRAINS	FLAG	DRAIN FLAG
STCONSOL	FLAG	CONSOLE IS RUNNING
STHRDLCK	FLAG	SYSTEM HARDLOCKED FLAG
STSSRESO	FLAG	SINGEL SERVICE RESOLUTION FLAG
STUSEMET	FLAG	USER METHOD RUNNING FLAG
STMSGLIM	FLAG	LOG MSG LIMIT REACHED FLAG
STMETMES	FLAG	METHOD MSG LIMIT REACHED FLAG
STREXMET	FLAG	REXX METHOD RUNNING FLAG
TOCOMP	LONG	NUMBER OF TIMES COMPRESSION DONE
TOCOMPI	LONG	COMPRESSION TOTAL BYTES IN
TOCOMPO	LONG	COMPRESSION TOTAL BYTES OUT
TODCOMPI	LONG	DECOMPRESSION TOTAL BYTES IN

TODCOMPO	LONG	DECOMPRESSION TOTAL BYTES OUT
TODCOMP	LONG	NUMBER OF TIME DECOMPRESSION DONE
TODBGETS	LONG	NUMBER OF GETS
TODBPUTS	LONG	NUMBER OF PUTS
TODBADDS	LONG	NUMBER OF ADDS
TODBDELE	LONG	NUMBER OF DELETES
TOFILEIO	LONG	FILE I/O COUNT
TOFALLOC	LONG	FILE ALLOCATION COUNT
TOMTHBIN	LONG	NUMBER OF METHODS RUNA – ASM AND C
TOMTHREX	LONG	NUMBER OF METHOS RUN – REXX
REMIPNAM	STR	IP NAME OF REMOTE CLIENT

Configuration Server Methods

Overview

A method is a program or procedure that can be packaged and exchanged as an object, specifically as an instance of the METHOD Class. By connecting an instance of this class to another class instance, an RCA administrator can specify where and when that program/procedure will run. An RCA administrator can also run a method from a REXX script, thereby enabling the execution of methods outside of the object resolution process. The following is an example of the format that is used to execute a method in this way.

```
ADDRESS EDMLINK ZOBJCMPR 'ZTEST'
```

EDMLINK is a method that enables an RCA administrator to process other methods. It returns the return code of the invoked method. The format for EDMLINK is:

`ADDRESS EDMLINK` *methodname* `'`*Parameter associated with Method*`'`

Configuration Server methods allow an RCA administrator to manipulate in-storage objects and database entities (database components) at the system (Configuration Server) level as opposed to the RCA agent or workstation objects.

- *Configuration Server Database components* are the entities (files, domains, classes, instances, and variables) that reside in the CSDB.
- *In-storage objects* are used or created during object resolution.

Note: Appendix A, Configuration Server Methods describes each method with parameters, examples, and return codes.

The Affects of Configuration Server Methods

The table "Methods affecting in-storage objects or CSDB entities" lists the Configuration Server methods that affect in-storage objects and database entities.

Note: The following table "Methods affecting in-storage objects or CSDB entities" lists only those Configuration Server methods that affect in-storage objects and database entities. Configuration Server methods that affect neither are not listed in this table.

Methods affecting in-storage objects or CSDB entities

Method	Affects
EDMMAILQ	In-storage objects
EDMMCACH	In-storage objects
EDMMDB	CSDB entities
EDMMRPRO	CSDB entities
ZDCLASS	CSDB entities
ZDELINS	CSDB entities
ZDELOBJS	In-storage objects
ZDELPROF	CSDB entities
ZEXIST	CSDB entities
ZGETPROF	CSDB entities
ZOBJCMPR	In-storage objects
ZOBJCOPY	In-storage objects
ZOBJDELI	In-storage objects
ZOBJDELV	In-storage objects
ZOBJSORT	In-storage objects
ZPROMANY	CSDB entities
ZPUTHIST	In-storage objects
ZPUTPROF	In-storage objects
ZSIMRESO	In-storage objects
ZTOUCH	CSDB entities
ZVARDEL	In-storage objects

ZVARGBL	In-storage objects
ZVARLOG	In-storage objects
ZXREF	In-storage objects

Methods are often used in conjunction with one another to achieve a purpose. For example, you can use the ZDELOBJS method to delete an in-storage object, and then execute ZOBJCOPY to copy an object, giving it the original object name.

Methods must be connected to other class instances at an appropriate point to achieve a desired result. For example, you do not want to delete an instance before it is used in object resolution, or create an instance if it will be immediately overwritten.

The default file and domain used by some methods can be specified in the DBASE and DOMAIN values of the MGR_STARTUP setting of the Configuration Server `edmprof` file.

Method Naming Standards

The standard that is used to name the methods is dissectible, enabling you to ascertain the method’s use. All method names are structured as detailed in "Method Naming Standards" above.

Configuration Server Methods Naming Standard

Symbol	Definition
EDM	Identifies the method as an HP method.
M	Identifies the method as a Configuration Server method.
DOBJ	Represents an abbreviation of the function for which the method can be used, in this case, Delete Object.

“Must Run” Methods

When you configure methods to run during the resolution process, you expect specific outcomes. If one method is intended to work in conjunction with another, or have a direct effect on the correct outcome of the resolution, the entire resolution process might depend on, first the existence of, then the successful launching of, this method.

You can designate a method as “must run,” which means that, before continuing with the resolution process, the Configuration Server will determine if the method exists and can be run. If it is not found or cannot be launched, the return code for the method will be set to 16 (Abort Resolution). The resolution will then be halted.

If you do not designate a method as “must run,” the Configuration Server does not recognize it as being essential to the outcome of the resolution and will continue processing based on the resulting return code. The only indications that the method was not processed are the return codes (as shown in messages in the log) and the result of the resolution path.

To configure a method as “must run,” insert the ZMUSTRUN variable in the METHOD instance and set the value to YES. (The default value for ZMUSTRUN is **NO**.) You can also establish a specific message to be returned by inserting the MSGONERR variable in the ZMETHOD instance.

Note: If no value is specified for MSGONERR, the following message will appear: `“CONFIGURATION UNCHANGED! UNABLE TO DETERMINE NEW CONFIGURATION.”`

Chapter 4

EDM Access Method Services (EDMAMS)

Caution: It is recommended to shut down the Configuration Server to ensure that the CSDB contents are not changing during the execution of the EDMAMS utilities.
However, ZEDMAMS can be run as a method, in which case the Configuration Server must be running.
In addition, it is recommended to back up the CSDB before running any of the EDMAMS utilities that will update it.

Overview

EDMAMS is a core set of utilities that can be used to create, delete, copy, change, and list objects in the Configuration Server Database. The functionality of these utilities is invoked using one of the EDMAMS verbs, which are called by the module, ZEDMAMS. The ZEDMAMS module is located in the directory in which the Configuration Server was installed.

- On a Windows system, the default is:
  <InstallDir>`\ConfigurationServer\bin`

Terminology

"Terminology" above lists terms that will be encountered in this chapter.

EDMAMS Verb Terminology

Term	Definition
Verb	The action to be performed on the database object. For example: `DELETE_INSTANCE`, `EXPORT_CLASS`, and `SYNC_CLASS`.
Keyword	The (required and optional) predefined database location or object designators, such as: `FILE`, `DOMAIN`, `FROMDOMA`, and `TOINST`. Also, a predefined instruction to be considered during the action. For example: `KEEPDATE`, `PREVIEW`, and `HEADER`.
Value	The user-specified database location or object on which the verb will act, such as: `POLICY`, `USER`, `RAD*`, and `EXPC.DAT`. Also, a user-specified consideration that accompanies the action. For example: Retaining an object's creation date and time (`KEEPDATE=YES`), Applying a comment to a copied object (`COMMENT=`*copied_object*), and Replacing all existing data in the target object (`REPLACE=YES`)
Unmated Instance	An instance that does not have a resource.
Orphaned Resource	A resource that does not have a parent instance.

Component Orphans	A component instance that does not have a parent (for example, a package instance).

Invoking the EDMAMS Verbs

For a list of the EDMAMS verbs that are available with the version of the CSDB running in an environment, on the command line, type:

```
ZEDMAMS VERB=
```

Note: The ZEDMAMS module is subject to continuous development. Due to this ongoing development, some of the verbs detailed in this chapter might not be applicable to the CSDB version that is running in your environment.
Subsequent printings of this guide will document enhancements to the EDMAMS functionality.

To display a verb's (required and optional) keywords, type:

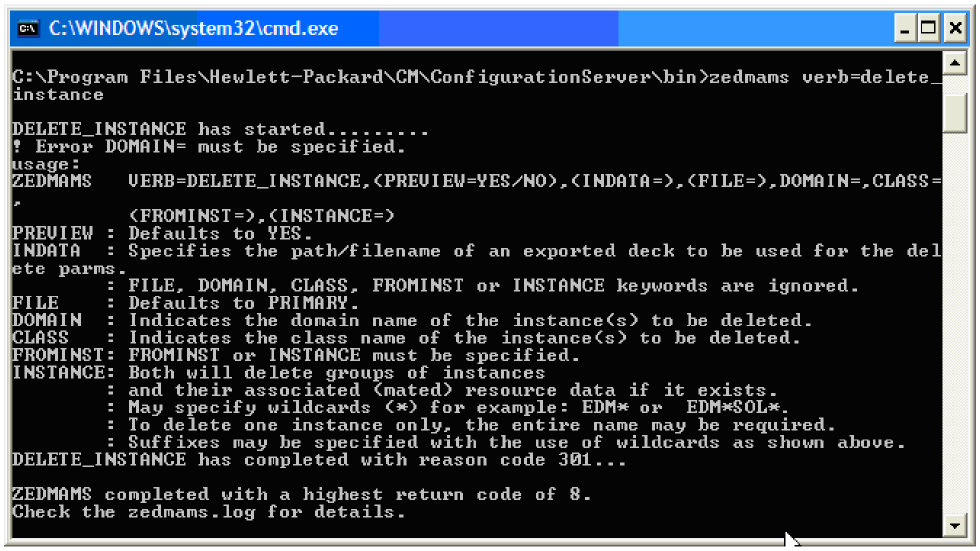
ZEDMAMS VERB=*VERB_NAME*

For example, typing,

```
ZEDMAMS VERB=DELETE_INSTANCE
```

on the command line, yields the following:

Keywords and conditions for the DELETE_INSTANCE verb



```
C:\Program Files\Hewlett-Packard\CM\ConfigurationServer\bin>zedmams verb=delete_
instance

DELETE_INSTANCE has started.........
! Error DOMAIN= must be specified.
usage:
ZEDMAMS    VERB=DELETE_INSTANCE,(PREVIEW=YES/NO),(INDATA=),(FILE=),DOMAIN=,CLASS=
,
           (FROMINST=),(INSTANCE=)
PREVIEW : Defaults to YES.
INDATA  : Specifies the path/filename of an exported deck to be used for the del
ete parms.
        : FILE, DOMAIN, CLASS, FROMINST or INSTANCE keywords are ignored.
FILE    : Defaults to PRIMARY.
DOMAIN  : Indicates the domain name of the instance(s) to be deleted.
CLASS   : Indicates the class name of the instance(s) to be deleted.
FROMINST: FROMINST or INSTANCE must be specified.
INSTANCE: Both will delete groups of instances
        : and their associated (mated) resource data if it exists.
        : May specify wildcards (*) for example: EDM* or  EDM*SOL*.
        : To delete one instance only, the entire name may be required.
        : Suffixes may be specified with the use of wildcards as shown above.
DELETE_INSTANCE has completed with reason code 301...

ZEDMAMS completed with a highest return code of 8.
Check the zedmams.log for details.
```

"Keywords and conditions for the DELETE_INSTANCE verb" above shows the results of invoking the verb, DELETE_INSTANCE. The keywords are listed, as well as an explanation of each, with examples and applicable defaults.

Note: In "Keywords and conditions for the DELETE_INSTANCE verb" above, some of the keywords are in parentheses; these keywords are optional.

For more information on using the verbs, see the sections, "Using the EDMAMS Verbs" below and "Usage Considerations" below.

Using the EDMAMS Verbs

All of the EDMAMS verbs are specified in the following format:

`ZEDMAMS VERB=`*VERB_NAME*`,`*KEYWORD*`=`*VALUE*`,`*KEYWORD*`=`*VALUE*

For example,

```
ZEDMAMS VERB=DELETE_INSTANCE,CLASS=USER,INSTANCE=SALES
```

Note: There are no rules governing the order in which the keyword-value combinations are specified.

Usage Considerations

- Optional keywords are enclosed in parentheses **( )**.
- Verbs, keywords, and data can be typed in UPPERCASE, lowercase, and a coMbinATioN. However, the value of string keywords, such as FROMDATA=, TODATA=, and STRING= are case-sensitive, where indicated.
- A comma (without a space) must follow each keyword-value combination, with the exception of the last keyword-value combination, which must be followed by a space.
- An asterisk (*) is not required for those values that recognize partial specification.
- For all of the EDMAMS verbs, the default value of FILE is **PRIMARY**.
- The EDMAMS verbs act on one database object per execution, unless otherwise noted. For example, to change the names of the instances, **East_Sales** and **North_Sales** to **US_Sales**, the verb RENAME_INSTANCE would have to be run once for each of these instances.
- Output is written to `zedmams.log`, unless a different log is specified for `LOGFILE=`. Error conditions are written to STDERR.

Input Files

Another way to run the EDMAMS utilities is to contain the commands in an input file—a file that has been edited by a text editor. This file can then be run to successively execute several EDMAMS verb functions.

- In an input file, the keywords can be specified on one or more lines for a single function.
- Like the command line method, a comma must follow each keyword-value combination, with the last keyword-value combination followed by a space.

EXPORT Verbs

Input files are very useful when using the exporting EDMAMS verbs (EXPORT_CLASS, EXPORT_INSTANCE, and EXPORT_RESOURCE) because they enable multiple domain-class combinations to be exported during a single export function.

The section, "Internationalization Considerations for Exporting/Importing Database Decks" on next page contains important information regarding the use of the EXPORT verbs.

Multiple Verbs

More than one verb can be specified in an input file. For example, the COPY_DOMAIN verb can be followed by the verbs, DELETE_CLASS and LIST_CONS_VARS. Any combination of verbs can be included in one run, as long as the data sets being accessed do not conflict. If an asterisk (*) is placed in the first column, it is considered a comment and no action is taken.

To execute commands that are contained in an input file, enter the following on the command line:

```
ZEDMAMS ZFILE "DRIVE:\FILE_PATH\FILE_NAME"
```

Note: ZEDMAMS is the executable. ZFILE is the second argument and must be in uppercase.

Example of multiple VERBS executed from one file:

```
VERB=COPY_DOMAIN,FROMDOMA=POLICY,TODOMAIN=SYSTEMA,REPLACE=NO

*

VERB=DELETE_CLASS,DOMAIN=SYSTEMA,CLASS=TESTCLAS

*

VERB=LIST_CONS_VARS,DOMAIN=SOFTWARE,CLASS=FILE,INSTANCE=*
```

Wildcards

EDMAMS supports two types of wildcards: *implicit* and *explicit*.

- **Implicit wildcards**
  are available for the COPY_INSTANCE, DELETE_INSTANCE, and LIST_INSTANCE verbs. Specify any portion of the value to select all occurrences that contain that portion of the value. Implicit wildcards do not require an asterisk, and are expressed as follows:
  KEYWORD=<wildcard_string>.
  For example, specify FROMINST=RAD to include all the fields that contain RAD as any part of the string.

- **Explicit wildcards**
  are available for the CHANGE_INS_FIELD, COPY_NEW_SUFFIX, COPY_RESOURCE, LIST_CONS_VARS, LIST_INST_DATA, LIST_ZRSC_FIELDS, and SEARCH_INSTANCES verbs. Explicit wildcards require an asterisk, and are expressed as follows:
  KEYWORD=<wildcard_string> *.
  For example, specify FROMINST=RAD* to select all the instances that contain RAD as the first part of the string.

LOGFILE

This keyword can be used with all of the EDMAMS verbs. Specify the fully qualified path to, and name of, the log file to which the information is to be reported. The default log file name is `ZEDMAMS.LOG`.

If this keyword is not specified, or specified without a value, the default location on windows is assumed the `bin` folder in which the `zedmams.exe` utility is running.

To send the information to a location\file other than the default, specify:

`LOGFILE=`*FILE_PATH*`\`*FILE_NAME*

Note: It is recommended that if LOGFILE is specified as a path other than the default, that its existence be verified. If the directory does not exist, the log creation process will fail, and the command will not execute.

Internationalization Considerations for Exporting/Importing Database Decks

This section details the conditions under which database decks can be exported/imported using the ZEDMAMS EXPORT and IMPORT verbs. "Codepage and Locale Defaults" on next page, in the section, "Codepage and Locale Defaults" on next page, is a supplement to this information; it lists the defaults of the new keywords for the EXPORT and IMPORT verbs.

EXPORT_CLASS, EXPORT_INSTANCE, and EXPORT_RESOURCE

Source Database Format	Acceptable Target Database Formats
UTF-8	UTF-8 and Basic ASCII
Other Locale or Codepage	UTF-8
Basic ASCII	UTF-8 and Basic ASCII
Extended ASCII	UTF-8 and Extended ASCII

EXPORT Verb Considerations

- If the deck is being exported from a database with a *locale* or *codepage* that differs from the database into which it is being imported, the appropriate keyword (FROM_LOCALE or FROM_CODEPAGE) must be specified.
  If the source database's *locale* or *codepage* is the same as the database into which it is being imported, neither keyword needs to be specified.
- If the deck is being exported from a database with a *locale* or *codepage* that differs from the database into which it is being imported, the appropriate keyword (TO_LOCALE or TO_CODEPAGE) must be specified.
  If the source database's *locale* or *codepage* is the same as the database into which it is being imported, neither keyword needs to be specified.

- If TO_LOCALE=LEGACY is specified, XPR headers will be automatically converted to EBCDIC.
- If TO_LOCALE=UTF8 is specified, an internationalized ZEDMAMS UTF-8 export deck is produced.

IMPORT_CLASS, IMPORT_INSTANCE, and IMPORT_RESOURCE

Target Database Format	Source Database Format Must Be
UTF-8	UTF-8 or Basic ASCII or Extended ASCII
Other Locale or Codepage	UTF-8
Basic ASCII	Basic ASCII
Extended ASCII	Extended ASCII

IMPORT Verb Considerations

- If the deck is being imported into a *locale* or *codepage* that differs from the database from which it was exported, the appropriate keyword (TO_LOCALE or TO_CODEPAGE) must be specified. If the target *locale* or *codepage* is the same as the database from which the deck was exported, neither keyword needs to be specified.
- If the deck is being imported into a *locale* or *codepage* that differs from the database from which it was exported, the appropriate keyword (FROM_LOCALE or FROM_CODEPAGE) must be specified.
  If the target *locale* or *codepage* is the same as the database from which the deck was exported, neither keyword needs to be specified.
- Translated data must fit the field size limits.
- Legacy XPR headers in EBCDIC are automatically converted to ASCII.

Codepage and Locale Defaults

The following table lists the defaults for the IMPORT and EXPORT *codepage* and *locale* keywords.

Codepage and Locale Defaults

Keywords	Default
FROM_ CODEPAGE	If the deck is encoded in UTF8, the default is **UTF8 codepage 65001**; otherwise, the default is the current system codepage.
TO_ CODEPAGE	If the database is encoded in UTF8, the default is **UTF8 codepage 65001**; otherwise, the default is the current system codepage.
FROM_ LOCALE	If the deck is encoded in UTF8, the default is **Locale country_region.UTF8**; otherwise, the default is the current process locale.
TO_ LOCALE	If the database is encoded in UTF8, the default is **Locale country_region.UTF8**; otherwise, the default is the current process locale.

Note: If the command line options are omitted, ZEDMAMS will open the import decks, look for the indicator and, if not found, will assume locale code page; it will then examine the database and look for the indicator and, if found, will treat as UTF8, but if not found, will treat as locale code page.

Specifying the ZEDMAMS Utility

"Specifying the ZEDMAMS Utility" above lists and describes the verbs for the ZEDMAMS utility.

ZEDMAMS Verbs

Verb	Description
ADD_FIELD	Adds a variable at the end of a template, including an automatic README file.
BUILD_ PATCH	Builds a PATCH file in a CSDB, or a file from two Configuration Server Databases.
BUILD_ STAGING_ POINT	Creates a staging point, as needed, on the hard drive, and enables the resources (represented as an export deck) to be converted into a tree of files located in a stager style directory.
CHANGE_ FIELDNAME	Changes variable names in class templates.
CHANGE_ FLD_VALUE	Changes a template’s variable length, type, Configuration Server, and RCA agent flags.
CHANGE_ INST_DATA	Globally changes data in instance records, by class.
CHANGE_ INS_FIELD	Changes one field in each instance of a class and verifies connects.
CHECK_ RESOURCES	Verifies the size of resources against ZRSCSIZE and ZCMPSIZE.
CLONE_ INSTANCE	Clones an instance with a four-digit suffix (max. 2000).
COPY_ CLASS	Copies a class, its instances, and, if it exists, its resource data.
COPY_DATA	Copies only resource data from one *domain.class.instance* to another.
COPY_ DOMAIN	Copies domains.
COPY_FIELD	Copies attribute data to a new attribute or to an existing attribute.
COPY_ INSTANCE	Copies an instance or a range of instances and associated resource records.

COPY_NEW_ SUFFIX	Copies and renames the suffixes for the FILE class and its mated instances.
COPY_ RESOURCE	See COPY_INSTANCE.
CREATE_ INSTANCES	Writes and populates instances from data in an edited text file.
DELETE_ CLASS	Deletes a class and its instances.
DELETE_ COMP_ ORPHS	Deletes component orphans from the PACKAGE class.
DELETE_ DOMAIN	Deletes one, or a range of, domains.
DELETE_ FIELD	Deletes an attribute from a template.
DELETE_ INSTANCE	Deletes/displays a range of instances within a class and, if it exists, its resource data.
DELETE_ ORPHANS	Deletes unmated resource records (orphans) from the RESOURCE file.
EDIT_ CLASS_ PREFIX	Updates the fields in the first 60 bytes of the prefix.
EXPORT_ CLASS	Exports classes to an output file or data set for import to another file or data set.
EXPORT_ INSTANCE	Exports instances to an output file or data set for import to another file or data set for reporting.
EXPORT_ RESOURCE	Exports resource data to an output file or data set for import to another file or data set.
IMPORT_ CLASS	Imports classes from an exported output file or data set to a PRIMARY file specified in the `edmprof` file.
IMPORT_ INSTANCE	Imports instances from an exported output file or data set to a PRIMARY file specified in the `edmprof` file.
IMPORT_ RESOURCE	Imports resource data from an exported output file or data set to a RESOURCE file specified in the `edmprof` file.
LIST_ CLASSES	Displays a list of classes with the class names, object IDs, ZOBJDATE, ZOBJTIME, persistence flag, sequence sensitive flag, Distributed Configuration Server flag, database type, and count totals. Note: If DOMAIN=* is specified, all classes in the domain will be listed.

LIST_CONS_ VARS	Displays connect and variable field (types C and V) values from instance records and stores output in `zedmams.log`.
LIST_ DOMAINS	Display a list of domains in the log of a specified file.
LIST_FLAGS	Lists Configuration Server and RCA agent flags in selected class record.
LIST_INST_ DATA	Lists instance data including variable names to `zedmams.log`.
LIST_ INSTANCE	Lists instance names by domain, class, and instance (prefix).
LIST_ PACKAGE	List PACKAGE class instances and all mated components.
LIST_PREFIX	Lists the Distributed Configuration Servers prefix by file, domain, and class.
LIST_ RESOURCES	Lists resource prefix information (promote date, instance, and so forth).
LIST_ZRSC_ FIELDS	Lists field values of fields beginning with ZRSC.
MATCH_ RESOURCES	Matches resource records against data-bearing instances for the specified class.
PACKAGE_ UNMATES	List all PACKAGE class instances without mated components.
REFRESH_ DMA	Recounts the instances and classes in a file and updates the Distributed Configuration Server prefix.
RENAME_ INSTANCE	Renames or displays instances by full name or prefix.
SEARCH_ INSTANCES	Searches selected instances by domain and class for specified string
SORT_ OBJECT_ID	Sorts object IDs in ascending or descending order by file or domain.
SYNC_ CLASS	Synchronizes an existing class with a newly formatted class, and re-organizes all (existing class) instances according to the mapping in the newly formatted class. All existing class attributes that have a match in the new template will adopt the characteristics of the new template attribute, whereas any existing class attributes that do not have a match in the new template will be deleted.
UPDATE_ INSTANCES	Updates the existing instance fields from data in an edited text file.
UPDATE_ MGRIDS	Updates the Configuration Server ID and name by file, domain, and class.

VERIFY_ CLASS	Verifies class templates for gaps, overlaps, and other anomalies.
ZRSOURCE_ UNMATES	Matches data-bearing instances with the appropriate resources for the specified class.

The sections that follow describe each ZEDMAMS verb.

Note: In the **Syntax** for each verb, the default values are presented in **bold**.

ADD_FIELD

This verb adds a variable (attribute) to the end of a class template and unconditionally includes a README attribute.

- The README attribute is 35 bytes long.
- The LENGTH of the specified attribute cannot be greater than 255 bytes, or less than one byte.
- The Configuration Server and RCA agent flags (MFLAGS and CFLAGS) are optional and will default according to the attribute TYPE. The default flags are presented when requesting a usage display.
- The FLDNAME can be any one- to eight-byte name and can be changed with the CHANGE_ FIELDNAME function.
- INDEX adds the field ‘before’ the *n*th (INDEX=n) occurrence of the same attribute name. For example, on an _ALWAYS_ connection, if 10 _ALWAYS_ connections are present, an INDEX =1 adds the attribute to the beginning, an INDEX = 5 adds the attribute at the 5th spot, and an INDEX >= 11 adds the attribute as the last _ALWAYS_. When an attribute is added to the end, it is placed below the last attribute of the same type. However, if no INDEX is specified, the attribute is added at the end of all attributes.

Syntax:	`(FILE=,)DOMAIN=,CLASS=,FLDNAME=,LENGTH=,TYPE=(,MFLAGS=) (,CFLAGS=)(,KEEPDATE=YES/ NO)(,README=)(,DEFAULT=)(,INDEX=n)`
Example:	Add an **M** type attribute, NEWFIELD, to the specified class, giving it a length of 165 bytes, assigning default Configuration Server and RCA agent flags, and updating the object date and time: **`DOMAIN=SOFTWARE,CLASS=USER,FLDNAME=NEWFIELD,LENGTH=165, TYPE=M,INDEX=5`**
Tip:	Run LIST_FLAGS against the class before and after running this to see the old and new template.

BUILD_PATCH

The functionality of this verb has been replaced with the **Service Optimization** feature of the RCA Admin CSDB Editor. For more information, refer to the *Radia Client Automation Enterprise Administrator User Guide*.

BUILD_STAGING_POINT

This verb enables resources (represented as an export deck) to be converted into a tree of files located in a stager style directory. It creates, on the hard drive, a staging point at a location designated by OUTFILE, then the data can be transferred to a CD using standard CD-writing software. This verb checks for the staging point and, if it does not exist, creates it.

- Specify PREVIEW=YES to see the expected results of running the verb with specific parameters.
- INFILE is the fully qualified path and filename of an exported resource (XPR) file.
- OUTFILE is the destination directory location of the staging files.
  OUTFILE defaults to the *location_of_edmprof\staging_point*. Therefore, if `edmprof` is located in `C:\HP\configserver\bin`, the default staging point is `C:\HP\configserver\bin\`*staging_point*.
  If PREVIEW=YES, OUTFILE is ignored.
- XPI is an exported instance deck that is used to timestamp the staged resources.
  This setting's value must be specified as the fully qualified path and filename of an exported instance deck.
  If XPI is not specified, or if the specified deck does not contain instance data for the staged resource, the timestamp will be determined using the date and time from the input deck resource header.
- REPLACE=YES replaces the existing same-named file at the specified staging point.
- MULTICAST=YES specifies that the OUTFILE will contain the name of a directory where resource data will be stored in File.Domain.Class.Instance format, and each file will contain an embedded 60-byte prefix, identical to the prefix saved in the RESOURCE file.

Syntax:	`(PREVIEW=YES/NO,)INFILE=(,OUTFILE=)(,XPI=)(,REPLACE=YES/NO)(,MULTICAST=YES/NO)`
Example:	Accept the default staging point on the hard drive as the destination for files from `C:\NovaFiles\NewCD`: `INFILE=C:\NovaFiles\NewCD`
Tip:	To see what would be the result of running the verb (without actually doing so), specify PREVIEW=YES.

Resource Naming

BUILD_STAGING_POINT will remove the 60-byte prefix from each resource that is delivered to the staging point, and rename it according to the following format:

1. The first byte of the object ID is added to the **000** string, and used as a *branch root*.
2. Bytes 2 – 9 will be used as a *name*, and followed by a period.
3. The last three bytes will be used as an *extension*.
   For example, a resource with an object ID of
   `DABC12345678`
   will be renamed:
   `000D\ABC12345.678.`

In accordance with step 1, the first byte (**D**) was added to the string **000**. The back slash (****) was automatically inserted to make **000D** a branch root. A period was placed in front of the final three bytes (**678**), making them the extension, as described in step 3. The bytes in between the branch root and the extension (**ABC12345**) become the name.

CHANGE_FIELDNAME

This verb changes the specified class template field name (FROMNAME) to a new field name (TONAME).

- If there are multiple occurrences of a field name (such as, README), all of them will be changed.
  However, README names should not be changed; only user-created variable field names should be changed.

Syntax:	`(FILE=,)DOMAIN=,CLASS=,FROMNAME=,TONAME=(,KEEPDATE=YES/ NO)`
Example:	Change the template field name in the specified class from GROUPID to GROUP: **`DOMAIN=SOFTWARE,CLASS=USER,FROMNAME=GROUPID,TONAME=GROUP`**
Tip:	Run LIST_INST_DATA to display template field names with INSTANCE=_BASE.

CHANGE_FLD_VALUE

This verb changes a variable's length, type, and Configuration Server and RCA agent flags by FLDNAME.

- The LENGTH must be in the range of 1 to 255 bytes.
- The Configuration Server (Manager) and RCA agent flags (MFLAGS and CFLAGS) are optional, and will retain their current settings if omitted.
- The keyword DESC changes the DESCRIPTION field in the template for a maximum of 20 bytes. The DESCRIPTION field can be changed either alone or in addition to other fields.
  If blanks are included in the text, it must be enclosed in quotation marks (" ").
- The keyword TYPE is required except when DESC is the only field being changed. The values are:
  - For connect types – the characters **C**, **A**, **I**, **R**, and **O**.
  - For method types – the characters **M**, **T**, **H**, and **D**.
  - For variable types – the characters **V**, **U**, and **W**.
- INDEX changes the nth occurrence of variable FLDNAME. When omitted, the first variable with a matching fieldname will be changed.
- KEEPDATE=YES will prevent the OBJDATE and OBJTIME from being updated.
- README is the 1- to 35-byte description of the changed field that was placed in the _BASE_ INSTANCE_. If blanks are included in the text, it must be enclosed in quotation marks (" ").
- DEFAULT the length of the default data must be less than or equal to the length specified by LENGTH.

This value populates FLDNAME in the _BASE_INSTANCE_. If blanks are included in the text, it must be enclosed in quotation marks (“ ”).

Syntax:	`(FILE=,)DOMAIN=,CLASS=(,DESC=,)FLDNAME=(,LENGTH=,) TYPE=(,MFLAGS=)(,CFLAGS=)(,INDEX=)(,KEEPDATE=YES/ NO)(,README=)(,DEFAULT=)`
Exam-ple:	In the template specified, change the TYPE value of the third attribute (named EDMSETUP) from V to C. (The length, and Configuration Server and RCA agent flags retain their current value.): `DOMAIN=SOF-TWARE,CLASS=ZSERVICE,FLDNAME=EDMSETUP,TYPE=C,INDEX=3`
Tip:	Run LIST_FLAGS against the class before and after running this to see the old and new template.

CHANGE_INST_DATA

This verb changes instance data.

- A match occurs only when the value of FROMDATA begins the instance field and is not embedded, or is not part of other data in the field.
- If PREVIEW=YES, only the instances to be changed will be displayed.
- If FIELD is specified, this verb will change all instances whose FROMDATA criteria is equal to the data portion of the heap specified by FIELD.
- FROMDATA is the data, in the heap, that is to be replaced.
  All instances that meet the criteria of FIELD and FROMDATA will be changed with TODATA.
  If FROMDATA=BLANKS, the change criteria will be all blank bytes.
- Omitting TODATA will set the field to blanks.
- FROMDATA and TODATA can be entered in any case. Keep in mind that the comparison made against FROMDATA is based on the case entered.
- If either FROMDATA or TODATA contain embedded spaces, the string must be enclosed in quotation marks.
- If KEEPDATE=NO, a new ZOBJDATE and ZOBJTIME are generated.

Syntax:	`(FILE=,-)DOMAIN=,CLASS=(,FIELD=,)FROMDATA=(,TODATA=)(,PREVIEW= YES/NO)(,KEEPDATE=YES/NO)`
Exam-ple:	Change all instance data in the specified class from *JohnPublic* to *John Q. Doe*: `DOMAIN=SOFTWARE,CLASS=USER,FROMDATA=JohnPublic,TODATA=“John Q. Doe”`
Tips:	Run LIST_INST_DATA to get a display of instance data and the class field names to which the data belongs. Run first with PREVIEW=YES to display the current variable data values.

CHANGE_INS_FIELD

This verb changes the instance data of the specified field in specific instances within the same domain.

- INDEX is the relative number (1– 99) of multiple same named fields (such as, EDMSETUP or README).
- The keyword PREFIX:
  - Can be specified wildcards (*****). For example, DIFF* and DIFF*SOL*.
  - If specified with just an asterisk (*****), all instances in the specified class are changed.
  - To change only one instance, the entire name must be specified.
- The keyword FIELD is the name of the attribute to be changed.
- VERIFY=YES verifies that a connection value in TODATA exists. If the verify fails, the program ends.
- KEEPDATE=YES will prevent the OBJDATE and OBJTIME from being updated.
- TODATA can be entered in any case; the data is placed in the field as entered. If TODATA contains embedded spaces, the string must be enclosed in quotation marks.
  Specify only CLASS.INSTANCE, not the domain.
- PREVIEW=YES displays instance names and the current data before the change and PREVIEW=NO displays instance names and the current data after the change.

Syntax:	`(FILE=,)DOMAIN=,CLASS=(,PREVIEW=`
	`YES``/NO,)PREFIX=,FIELD=,TODATA=(,INDEX=)(,VERIFY=YES/`
	`NO)(,KEEPDATE=YES/`
	`NO)`

Example:	Change the third EDMSETUP field (INDEX=3) in the specified instances to ZLOCMGR.RAD_RESOURCE_FILE. Verify the existence of the connection, and update the OBJDATE and OBJTIME:
	`DOMAIN=SOFTWARE,CLASS=USER,PREFIX=TSO,FIELD=EDMSETUP,`
	`TODATA=ZLOCMGR.RAD_RESOURCE_FILE,VERIFY=YES,INDEX=3`

Tip:	This verb enables wildcards for the PREFIX parameter. Therefore, you can specify PREFIX=RAD* to see all the prefixes that contain RAD as the first part of the string.

CHECK_RESOURCES

This verb verifies the actual size of the resource data against ZRSCSIZE or ZCMPSIZE.

The entire PRIMARY file is checked for the presence of the variable field names ZRSCSIZE and ZCMPSIZE. If these fields exist, and if ZCMPSIZE contains a value, it is compared against the actual size. If ZCMPSIZE is zero or blank, the value in ZRSCSIZE is used. If there is a mismatch, the name of the resource and the appropriate sizes are listed to the log and a return code of 8 is passed.

- If LISTALL=YES, all objects checked are listed, and objects with resources are listed with their respective sizes.
- If UPDATE=YES, and if the appropriate of either ZRSCSIZE or ZCMPSIZE does not contain the correct value, the field will be updated with the correct value. In addition, the 8-byte, printable hex field in the instance prefix will be updated if in error.
- CRC is a toggle to enable/disable the Cyclical Redundancy Check.
  If CRC=YES, a CRC is calculated for each resource and if UPDATE=YES, incorrect CRCs are updated to their correct values in the OBJRCRC field.
- DOMAIN is a 1- to 32-byte (Domain) name that is to have its resources checked.
- CLASS is a 1- to 8-byte (class) name that is to have its resources checked.
  If specified, DOMAIN must be specified.
- INSTANCE is a 1- to 32-byte (Instance) name that is to have its resources checked.
  If specified, DOMAIN and CLASS must be specified.

Syntax:	`LISTALL=YES/ NO(,UPDATE=YES/NO)(,CRC=YES/NO,)DOMAIN=,CLASS=,INSTANCE=`
Example:	List all the resources that have a mismatch: `CHECK_RESOURCES,LISTALL=YES`
Tip:	N/A

CLONE_INSTANCE

This verb clones the instance a specified number (nnnn) of times and each new instance name is suffixed with one to four digits: 0000 through nnnn-1.

The cloned instance name, plus its suffix, cannot exceed 32 bytes: 1 to 28 digits for the instance name + 1 to 4 digits for the suffix. Therefore, instance name length = 32, the length of the suffix.

- A new OBJID, OBJDATE, and OBJTIME are generated for the cloned objects.
- COUNT is the number of clones to spawn, and can range from 1 to 2000.

Syntax:	`DOMAIN=,CLASS=,INSTANCE=,COUNT=nnnn`
Example:	Clone 100 instances in the specified class, naming the cloned instances DIFF80 through DIFF899 type: **`DOMAIN=SOFTWARE,CLASS=USER,INSTANCE=DIFF8,COUNT=100`**
Tip:	Run LIST_INSTANCE to display the instance names before and after the cloning.

COPY_CLASS

This verb copies a class template, its component instances, and resource data from one domain to another.

Note: While this verb is supported, it is recommended to use the verbs EXPORT_CLASS and IMPORT_CLASS to copy a class from the CSDB.

As of version 4.4 of the CSDB, this verb will copy the component instances and resource data also.

- TODB specifies the path to a destination file other than the one in the `edmprof` file. If omitted, it defaults to the DBPATH specified in the `edmprof` file.
- If TOCLASS is omitted, the destination class will be assumed to be the same as the FROMCLAS. In this case, TODOMAIN and FROMDOMA must be different.
- A new object ID, ODJDATE, and OBJTIME are generated for the copied objects.
- If TODOMAIN does not exist, the ZBASE class and base instance will be copied from the source domain (FROMDOMA) to create a valid domain.
- If REPLACE=YES, all existing data is replaced.

Syntax:	`(TODB=)(,FILE=,)FROMDOMA=,FROMCLAS=,TODOMAIN=(,TOCLASS=)(,REPLACE=YES/NO)`
Example:	Copy the specified class template and only the base instance from SOFTWARE to SYSTEM: `FROMDOMA=SOFTWARE,FROMCLAS=USER,TODOMAIN=SYSTEM`
Tip:	Run once with REPLACE=NO to determine the preexistence of the class in the destination domain.

COPY_DOMAIN

This verb copies only the class template and _BASE_INSTANCE_ of a domain within a database, and optionally, to a different destination database. To copy the entire contents of a domain, see the section, "Copying a Domain and its Contents" below.

- A new object ID, OBJDATE, and OBJTIME are generated for the copied objects.
- If REPLACE=YES, all existing data is replaced.
- TODB specifies the path to a destination file other than the one in the `edmprof` file. If omitted, it defaults to the DBPATH specified in the `edmprof` file.

Syntax:	`(FILE=,)FROMDOMA=,TODOMAIN=(,REPLACE=YES/NO)(,TODB=)`
Example:	Copy the class template and _BASE_INSTANCE_ of the SOFTWARE Domain to the SYSTEM domain: `FROMDOMA=SOFTWARE,TODOMAIN=SYSTEM`
Tip:	Run once with REPLACE=NO to determine the preexistence of the destination domain.

Copying a Domain and its Contents

To copy a domain, run this verb with FROMDOMA and TODOMAIN specified, then use the export and import verbs as detailed below.

To copy a domain with contents:

1. Run this verb with FROMDOMA and TODOMAIN specified.
   This will copy just the _BASE_INSTANCE_ and class template.
2. Verify that the new domain has been created by checking the ZEDMAMS log's return code or physically querying the CSDB using an explorer tool.
3. Use the EDMAMS verbs, "EXPORT_INSTANCE" on page 135 and "EXPORT_RESOURCE" on page 135 with **DOMAIN=**<*old_domain*> (the value of FROMDOMA).
4. Import the domain using the EDMAMS verb, "IMPORT_INSTANCE" on page 135. Be sure to specify:
   - the exported instance (**FILE=**),
   - the resource files (**XPR=**),
   - **MAP_DOMAIN=***old_domain/new_domain*.

COPY_FIELD

This verb copies an attribute (FROMFLD) and its instance data to a new attribute (TOFLD). The length, type, and flags of the new attribute will be inherited from the existing attribute.

- INDEX is the nth occurrence of a FROMFLD multiple-named attribute.
- PREVIEW will display the value of the FROMFLD attribute before changing an existing TOFLD attribute.
- REPLACE=YES will overlay existing data in the TOFLD attribute with the values in the FROMFLD attribute.
- If KEEPDATE=NO, a new zobjdate and zobjtime are generated.

Syntax:	`(FILE=,)DOMAIN=,CLASS=,FROMFLD=,TOFLD=(,INDEX=)(,PREVIEW= YES/NO)(,REPLACE=YES/NO)(,KEEPDATE=YES/ NO)`
Example:	Copy an attribute named OLDFLD (and the associated data in the specified class) to a new attribute named NEWFLD: **`DOMAIN=SOFTWARE,CLASS=USER,FROMFLD=OLDFLD,TOFLD=NEWFLD`**
Tip:	Run LIST_INST_DATA against the class to see the contents of each attribute.

COPY_INSTANCE (COPY_RESOURCE)

These verbs copy a range of specified instances and resource data from one domain-class pair to another, in the PRIMARY and RESOURCE files, and optionally, to a different destination database. The function assumes that the destination class name is the same as the source, and that the templates are identical.

Note: While this verb is supported, it is recommended to use the verbs EXPORT_INSTANCE and IMPORT_INSTANCE to copy an instance from the CSDB.
This verb can be specified as either COPY_INSTANCE or COPY_RESOURCE, as these verbs have been combined and perform identical functions.

As of version 4.4 of the Configuration Server Database, this verb will copy the component instances and resource data also.

- TODB specifies the path to a destination file other than the one in the `edmprof` file. If omitted, it defaults to the DBPATH specified in the `edmprof` file.
- The value of FROMINST can be partially specified. For example, you can specify FROMINST=DIFF to specify all the instances that contain DIFF as any part of the string.
- A new object ID is generated for the copied objects. A new OBJDATE and OBJTIME will be generated unless KEEPDATE=YES.
- PREVIEW=YES will display a list of instance names that will be copied from the source class. PREVIEW=NO will display instances that have been copied.
- If REPLACE=NO and an existing instance is found, the function will stop, indicating the existing instance name to STDERR and the log, and listing in the log any instances that have been copied.
- The existing instance name will be written to STDERR, and instances that have been copied will be listed in the log.
- If PREVIEW=NO and the destination domain does not contain FROMCLAS, the source class (FROMCLAS) and its BASE_INSTANCE will be copied to the destination domain (TODOMAIN).
- NEWINST renames an instance (if TODB is specified).
  If wildcards are used for FROMINST, NEWINST is not allowed.

Syntax:	`(PREVIEW=YES`
	`/NO)(,TODB=)(,-`
	`FILE=,)FROMDOMA=,FROMCLASS=,TODOMAIN=,FROMINST=`
	`(,NEWINST=)(,REPLACE=YES/`
	
	`NO)(,KEEPDATE=YES/NO)`

Exam-ple:	Copy instances and resource data from the SOFTWARE Domain to the SYSTEM domain,
	with the from-instance wildcard specified as CICS*ICO*:
	`FROMDOMA=SOFTWARE,FROMCLAS=ZSERVICE,TODOMAIN=SYSTEM,`
	`FROMINST=CICS*ICO*`

Tips:	Run once with PREVIEW=YES to get a list of instances to be copied. Also, run LIST_RESOURCES against the source domain to display a list of existing resources and instance names that can be copied, and then against the destination domain to see what was copied. Destination domain instance names that match the new instance name will be deleted.

COPY_NEW_SUFFIX

This verb copies the specified instances and resource data from one domain to another in the PRIMARY and RESOURCE files; and adds a new suffix to the destination instance. This verb allows wildcards for FROMINST.

- The value of FROMINST can be partially specified. For example, you can specify FROMINST=DIFF to select all the instances that contain DIFF as the first part of the string.
- If the length of NEWSUFF is longer than that of OLDSUFF, the new instance name must not exceed 32 bytes. If it does, a message will be placed in the log and the instance will not be copied.
- PREVIEW=YES will display the old instance name and the new instance name and the total number of instances to be copied.
- If the class does not exist in the TODOMAIN, the class and BASE_INSTANCE will be copied from the source domain (FROMDOMA).
- Instance names must match the prefix (FROMINST) and the suffix (OLDSUFF) to be copied and renamed.

Syntax:	`(PREVIEW=YES /NO,) F- ROMDOMA=,FROMCLASS=,FROMINST=,TODOMAIN=,OLDSUFF=,NEWSUFF=`
Exam- ple:	From the SOFTWARE Domain, copy and re-suffix instances and resource data prefixed with TSO and suffixed with ICO, to the SYSTEM domain with the new suffix, TSO_(*)_NEW_ICO; and displaying only those that would be copied: **`FROM- DOMA=SOF- TWARE,FROMCLAS=ZSERVICE,TODOMAIN=SYSTEM,FROMINST=TSO, OLDSUFF=ICO,NEWSUFF=NEW_ICO`**
Tips:	Run once with PREVIEW=YES to verify that the new instance names will not exceed 32 bytes, and that the instances required will be copied. Also, run LIST_ RESOURCES against the source domain to display a list of existing resources and instance names that can be copied and then against the destination domain to see what was copied. Destination domain instances that match the new instance name will be deleted.

CREATE_INSTANCES

This verb creates Instances in the specified CSDB Domain from data in an edited text file (INFILE). The INFILE must conform to the following specifications.

- The first line must contain a new Instance name starting in column 1 for a maximum of 32 bytes.
- The next line contains the variable field name, beginning in column 2, for a maximum of eight bytes; then a blank in column 10; followed by a two-byte index value (01-99) in columns 11 and 12; followed by the data to be populated beginning in column 13.
- Existing Instance names will be bypassed and the function will stop unless a /* (in columns 1 and 2) line follows the last data line of an Instance. In the example below, the function will continue to the next input Instance.

Col- umn Numbe- r	1	2	3	4	5	6	7	8	9	1- 0	1- 1	1- 2	1- 3	1- 4	1- 5	1- 6	1- 7	1- 8	1- 9	2- 0	2- 1	2- 2	2- 3	2- 4

Line 1	U	S	E	R	_	N	A	M	E															
Line 2		N	A	M	E		J	.	A	.	D	E	V	E	L	O	P	E	R					
Line 3		E	D	M	S	E	T	U	P		D	O	M	A	I	N	.	C	L	A	S	S		
Line 4		Z	P	R	I	O	R	I	T		9	9	9											
Line 5	U	S	E	R	_	N	A	M	E		L	.	Z	I	M	M	E	R						
Line 6		N	A	M	E																			
Line 7		E	D	M	S	E	T	U	P		U	S	E	R	.	U	S	E	R	3				
Line 8		Z	T	R	A	C	E		N	N	N													

Syntax:	`INFILE=,DOMAIN=,CLASS=`
Example:	Create the instances in the specified text file in CLASS=USER: `INFILE=MYINPUT.TXT,DOMAIN=POLICY,CLASS=USER`
Tip:	N/A

DELETE_CLASS

This EDMAMS verb deletes a class template and all of its instances.

As of version 4.4 of the Configuration Server Database, this verb will delete the component instances and resource data also.

- If PACKAGE=YES, all component and component data is deleted.

Syntax:	`(FILE=)(,PREVIEW=YES/NO)(,PACKAGE=YES/NO,)DOMAIN=,CLASS=`
Example:	Delete the class, USERTEST, from the SOFTWARE Domain: `DOMAIN=SOFTWARE,CLASS=USERTEST`
Tip:	N/A

DELETE_COMP_ORPHS

This verb deletes component orphans from the PACKAGE class. Orphans are defined as RESOURCE file data that have no mated instances in the associated PRIMARY file PACKAGE class.

- CLASS defaults to all classes of the specified domain.

Syntax:	`(FILE=)(,PREVIEW=YES/NO,)DOMAIN=(,CLASS=)`
Example:	Delete orphans from the TSTRESLT Class of the SOFTWARE Domain: `DOMAIN=SOFTWARE,CLASS=TSTRESLT,PREVIEW=NO`
Tip:	N/A

DELETE_DOMAIN

This verb deletes a domain or an alphabetical range of domains (class templates and instances) from the file that is specified.

As of version 4.4 of the Configuration Server Database, this verb will delete the component instances and resource data also.

- If TODOMAIN is omitted, only FROMDOMA will be deleted.
  If TODOMAIN is specified, the range is inclusive.

Syntax:	`(FILE=)(,PREVIEW=YES/NO,)FROMDOMA=(,TODOMAIN=)`
Example:	From the PRIMARY file, delete only SOFTWARE: `FILE=PRIMARY,FROMDOMA=SOFTWARE` From the PROFILE File, delete USERA through USERF: `FILE=PROFILE,FROMDOMA=USERA,TODOMAIN=USERF`
Tip:	Run with PREVIEW=YES first.

Caution: Use caution when specifying a range of domains. Be certain of the range specified, because there is no confirmation request.

DELETE_FIELD

This verb deletes an attribute from a template along with its README field, and reorganizes the class accordingly.

- FILE will default to **PRIMARY** if it is omitted or if no value is specified.
- DOMAIN specifies the name of the CSDB Domain.
- CLASS specifies the name of the CSDB Class.
- FIELD must refer to an attribute, not a README (description) field.
- INDEX refers to multiple occurrences of the same attribute name. For example, if there were three occurrences of EDMSETUP, the third would be INDEX=3. Index deletes the *n*th occurrence of a multiple named variable.
  Values for INDEX are the numbers 1 through 99.
  The default is **1**.

Syntax:	`(FILE=,)DOMAIN=,CLASS=,FIELD=(,INDEX=)`
Example:	Delete the attribute USERATTR from the USER Class: `DOMAIN=POLICY,CLASS=USER,FIELD=USERATTR`
Tip:	Use with discretion because deletion of an attribute causes a restructuring and rewrite of all the instances in the class.

DELETE_INSTANCE

This verb deletes an Instance or an alphabetical range of Instances within a specified Class of the CSDB.

The keywords, TOINST and SUFFIX, were deleted from this verb's functionality with version 4.3 of the Configuration Server Database.

As of version 4.4 of the Configuration Server Database, this verb will delete the component instances and resource data also.

- FROMINST and INSTANCE perform the same function; either one must be used. Both will delete groups of instances and any existing, associated resource data. To delete only one instance, the entire name must be specified.
- FROMINST and INSTANCE can specify wildcards (*). For example, DIFF* and DIFF*SOL*.
- PREVIEW=YES displays a list of instances that would be deleted with PREVIEW=NO.
- INDATA specifies the fully qualified path to, and name of, a file that contains the delete parameters. This file can be either an exported instance deck (.XPI), or a manually created file that contains the FILE, DOMAIN, CLASS, and INSTANCE values.
  If using INDATA, the values that are specified for FILE, DOMAIN, CLASS, FROMINST, and INSTANCE on the command line are ignored because these values are extracted from the INDATA file.
  When not using INDATA, either FROMINST or INSTANCE *must* be specified. Both can delete groups of instances.
  FROMINST (instead of INSTANCE) can be specified inside the INDATA file to define a group of instances to be deleted.

Syntax:	`(PREVIEW=YES`
	`/NO)-`
	`(,INDATA=)(,FILE=,)DOMAIN=,CLASS=(,FROMINST=)(,INSTANCE=)`

Exam-ple:	Delete all USER Class instances, beginning with the instance prefix name of `OLD_USER:DOMAIN=POLICY,CLASS=USER,FROMINST=OLD_USER`
Tip:	Run once with PREVIEW=YES to see which instances will be deleted. Also, run LIST_INSTANCES against the class to display a list of instance names that might be deleted.

Caution: Use caution when specifying a range of domains. Be certain of the range specified, because there is no confirmation request.

DELETE_ORPHANS

This verb will delete all orphans in all domains. Orphans are defined as RESOURCE file data that have no mated instance in the associated PRIMARY file.

- TRACE=YES provides additional diagnostic (tracing) confirmation in the log.

Syntax:	`(PREVIEW=YES/NO)(,TRACE=YES/NO)`
Example:	Delete all orphans: `PREVIEW=NO`
Tip:	N/A

DELETE_RESOURCE

This verb deletes the specified resource from the RESOURCE File of the CSDB.

The keyword, DELETE_RESOURCE, was deleted from the Configuration Server as of version 4.2. Its functionality is handled by the verb, "DELETE_INSTANCE" on page 135.)

EDIT_CLASS_PREFIX

This verb enables you to edit the first 60 bytes of the template's prefix.

- FIELD is the name of the class prefix field, such as the priority of the class, CLASSPRI.
- VALUE will not be read if FIELD is not specified.
  Refer to the **Values** column in "EDIT_CLASS_PREFIX" above for the valid values for each FIELD type.
- If KEEPDATE=NO, a new OBJDATE and OBJTIME are generated.

Changeable Fields

Field	Values
CLASTYPE	**P** (POLICY_CLASS_TYPE) **C** (CONFIGURATION_CLASS_TYPE) **T** (COMPONENT_CLASS_TYPE) **B** (CLASS_TYPE_BLANK)
CLASSPRI	**5** (PATH), **10** (METACLAS), **50** (FILE), **50** (ZSERVICE), **60** (DESKTOP), **70** (MACALIAS),**70** (REGISTRY), **50** (DEFAULT_PRIORITY), **B** (Blank)
DBTYPE	**U** (UNICODE) **A** (ASCII) Note: A third value (**E**) is not applicable to the current release of the Configuration Server.
OBJDM	**D** (DISTRIBUTED_MANAGER)
OBJTYPE	**S** (SINGLE_DIMENSIONAL_OBJECT) **M** (MULTIPLE_DIMENSIONAL_OBJECT)
SEQ_ SENS	**S** (SEQUENCE_SENSITIVE) **I** (SEQUENCE_INSENSITIVE) Note: Specifies whether to process variables in the order in which they occur in the class template (**S**), or in order of attribute type (**I**)—that is, VARs then CONNs, and so on.
OBJNAME	Any text up to 20 bytes in length. If there are embedded spaces in the text, enclose the text with quotation marks (“ ”), as in the example.

Syntax:	`DOMAIN=,CLASS=,FIELD=,VALUE=(,PREVIEW=YES/NO)(,KEEPDATE=YES/NO)`
Example:	**`DOMAIN=POLICY,CLASS=USER,FIELD=OBJNAME,VALUE="User names",PREVIEW=NO`**
Tip:	N/A

EXPORT_CLASS

This verb enables the exporting of class template data from a file or data set for importing to another file or data set.

- If PREVIEW=YES, OUTPUT is ignored.
- OUTPUT is the name of the destination output file (with extension) where the exported data is to reside.
- INPUT references a predefined input file, which enables multiple FILE.DOMAIN.CLASS combinations to be specified.
  The input file must be specified in the following format.
  `FILE=`*file_name*`,DOMAIN=`*domain_name*`,CLASS=`*class_name*`,INSTANCE=`*instance_name*
- HEADER=YES produces an output header file.
- COMMENT=YES adds a comment to the optional output file header.
- If the deck is being exported from a database with a *locale* or *codepage* that differs from the database into which it is being imported, the appropriate keyword (FROM_LOCALE or FROM_CODEPAGE) must be specified.
  If the source database's *locale* or *codepage* is the same as the database into which it is being imported, neither keyword needs to be specified.
- If the deck is being exported from a database with a *locale* or *codepage* that differs from the database into which it is being imported, the appropriate keyword (TO_LOCALE or TO_CODEPAGE) must be specified.
  If the source database's *locale* or *codepage* is the same as the database into which it is being imported, neither keyword needs to be specified.
  - The values for the CODEPAGE keywords must be integers, such as 1252, 65001, and 936.
  - The LOCALE keywords' values will accept the following values only: **LEGACY**, **UTF8**, and **UTF-8**.
    - **LEGACY** is an alias for the local machine's code page, such as CP_ACP, 1252.
    - **UTF8** and **UTF-8** are aliases for a UTF-8 code page, such as CP_UTF8, 65001.
- If TO_LOCALE=
  - LEGACY: XPR headers will be automatically converted to EBCDIC.
  - UTF8: an internationalized ZEDMAMS UTF-8 export deck is produced.

Note: The keywords FROM_LOCALE, FROM_CODEPAGE, TO_LOCALE, and TO_CODEPAGE will not show up on the command line when the syntax/usage is queried; however, they are functional and, when specified, will work as designed.

Syntax:	`FILE=(,DOMAIN=)(,CLASS=)(,PREVIEW= YES/NO,)OUTPUT=(,COMMENT=)(,INPUT=)(,HEADER=YES/NO)`
Example:	Export all the classes (templates) in the PRIMARY file to a file named **EXPC.DAT**: `FILE=PRIMARY,PREVIEW=NO,OUTPUT=C:\EXPC.DAT`
Tip:	N/A

EXPORT_INSTANCE

This verb enables the exporting of instances to an output file or data set for importing to another file or data set or for reporting purposes.

- If PREVIEW=YES, OUTPUT is ignored.
- OUTPUT is the name of the destination output file (with extension) where the exported data is to reside.
- KEEP specifies a text file that contains a list of the instance attributes to be retained in the resulting output file. These names are case-sensitive.
- DROP specifies the instance attributes that are not to be retained in the output file.
- If ORDER=YES, the resulting file will be ordered by attribute name.
- COMMENT=YES adds a comment to the optional output file header.
- Specify REPORT=YES to export instances to third-party vendor software.
- CSVL=YES produces a Comma Separated Variable listing for all attribute values. Currently, the output file you specify with CSVL=YES is created in a subdirectory of the current working directory.
- If BASE=YES, the values of the _BASE_INSTANCE_ will be inherited.
- INPUT references a predefined input file, which enables multiple FILE.DOMAIN.CLASS.INSTANCE combinations to be specified.
  The input file must be specified in the following format.
  `FILE=`*file_name*`,DOMAIN=`*domain_name*`,CLASS=`*class_name*`,INSTANCE=`
  instance_name
- HEADER=YES produces an output header file.
- If SKIP_ERRORS=YES and database or consistency errors are encountered in the PROFILE File, a “bad object” event (return code=4) will be recorded in the log, but processing will continue. If SKIP_ERRORS=NO (the default) and errors are encountered, the exporting will stop.
- PHEX=YES outputs the data portion of variables in printable hex format.

- If the deck is being exported from a database with a *locale* or *codepage* that differs from the database into which it is being imported, the appropriate keyword (FROM_LOCALE or FROM_CODEPAGE) must be specified.
  If the source database's *locale* or *codepage* is the same as the database into which it is being imported, neither keyword needs to be specified.
- If the deck is being exported from a database with a *locale* or *codepage* that differs from the database into which it is being imported, the appropriate keyword (TO_LOCALE or TO_CODEPAGE) must be specified.
  If the source database's *locale* or *codepage* is the same as the database into which it is being imported, neither keyword needs to be specified.
  - The values for the CODEPAGE keywords must be integers, such as 1252, 65001, and 936.
  - The LOCALE keywords' values will accept the following values only: **LEGACY**, **UTF8**, and **UTF-8**.
    - **LEGACY** is an alias for the local machine's code page, such as CP_ACP, 1252.
    - **UTF8** and **UTF-8** are aliases for a UTF-8 code page, such as CP_UTF8, 65001.
- If TO_LOCALE=
  - LEGACY: XPR headers will be automatically converted to EBCDIC.
  - UTF8: an internationalized ZEDMAMS UTF-8 export deck is produced.

Note: The keywords FROM_LOCALE, FROM_CODEPAGE, TO_LOCALE, and TO_CODEPAGE will not show up on the command line when the syntax/usage is queried; however, they are functional and, when specified, will work as designed.

Syntax:	`FILE=(,DOMAIN=)(,CLASS=)(,INSTANCE=)(,FROMINST=)(,TOINST=)(,FROMDATE=)(,PREVIEW=YES/NO)(,TODATE=)(,FROMTIME=)(,TOTIME=,)OUTPUT=(,KEEP=YES/NO)(,DROP=)(,ORDER=)(,COMMENT=)(,CSVL=YES/NO)(,PHEX=YES/NO)(,BASE=YES/NO)(,INPUT=)(,HEADER=YES/NO)(,REPORT=YES/NO)(,SKIP_ERRORS=YES/NO)`
Example:	Export all the instances in the PRIMARY file to a file named EXPI.DAT: `FILE=PRIMARY,PREVIEW=NO,OUTPUT=C:\EXPI.DAT`
Tip:	N/A

EXPORT_RESOURCE

This verb enables the exporting of resource data to an output file or data set for importing to another file or data set.

- If PREVIEW=YES, OUTPUT is ignored.
- OUTPUT is the name of the destination output file (with extension) where the exported data is to reside.

- Specify COMMENT=YES to add a comment to the optional output file header.
- INPUT references a predefined input file, which enables multiple FILE.DOMAIN.CLASS.INSTANCE combinations to be specified.
  The input file must be specified in the following format.
  `FILE=`*file_name*`,DOMAIN=`*domain_name*`,CLASS=`*class_name*`,INSTANCE=` *instance_name*
- HEADER=YES produces an output header file.
- If the deck is being exported from a database with a *locale* or *codepage* that differs from the database into which it is being imported, the appropriate keyword (FROM_LOCALE or FROM_ CODEPAGE) must be specified.
  If the source database's *locale* or *codepage* is the same as the database into which it is being imported, neither keyword needs to be specified.
- If the deck is being exported from a database with a *locale* or *codepage* that differs from the database into which it is being imported, the appropriate keyword (TO_LOCALE or TO_ CODEPAGE) must be specified.
  If the source database's *locale* or *codepage* is the same as the database into which it is being imported, neither keyword needs to be specified.
  - The values for the CODEPAGE keywords must be integers, such as 1252, 65001, and 936.
  - The LOCALE keywords' values will accept the following values only: **LEGACY**, **UTF8**, and **UTF-8**.
    - **LEGACY** is an alias for the local machine's code page, such as CP_ACP, 1252.
    - **UTF8** and **UTF-8** are aliases for a UTF-8 code page, such as CP_UTF8, 65001.
- If TO_LOCALE=
  - LEGACY: XPR headers will be automatically converted to EBCDIC.
  - UTF8: an internationalized ZEDMAMS UTF-8 export deck is produced.

Note: The keywords FROM_LOCALE, FROM_CODEPAGE, TO_LOCALE, and TO_ CODEPAGE will not show up on the command line when the syntax/usage is queried; however, they are functional and, when specified, will work as designed.

Syntax:	`FILE=(,DOMAIN=)(,CLASS=)(,INSTANCE=)(,FROMINST=)(,TOINST=) (,PREVIEW= YES/NO)(,FROMDATE=)(,TODATE=)(,FROMTIME=)(,TOTIME=,) OUTPUT=(,COMMENT=YES/ NO)(,INPUT=)(,HEADER=YES/ NO)`
Example:	Export all resources in the RESOURCE file to a file named EXPR.DAT: `FILE=PRIMARY,PREVIEW=NO,OUTPUT=C:\EXPR.DAT`
Tip:	N/A

IMPORT_CLASS

This verb enables you to import template data from an exported data set or output file to a PRIMARY File specified in the `edmprof` file.

- FILE is the name of the file or data set that contains the import class data..
- TIME=NEW – generates a new OBJDATE, OBJTIME, and OBJID.
  TIME=OLD – retains the original OBJDATE, OBJTIME, and OBJID.
  TIME=MOD – generates a new OBJDATE and OBJTIME, but retains the original OBJID.
- TODOMA is the domain with which matching source domains are replaced.
- If the FROMDOMA domain exists in the destination database, specify TIME=NEW to avoid duplicate object IDs.
  If FROMDOMA is specified, TODOMA must also be specified.
- If REPLACE=NO, the class template will not be replaced.
- If the deck is being imported into a *locale* or *codepage* that differs from the database from which it was exported, the appropriate keyword (TO_LOCALE or TO_CODEPAGE) must be specified.
  If the target *locale* or *codepage* is the same as the database from which the deck was exported, neither keyword needs to be specified.
- If the deck is being imported into a *locale* or *codepage* that differs from the database from which it was exported, the appropriate keyword (FROM_LOCALE or FROM_CODEPAGE) must be specified.
  If the target *locale* or *codepage* is the same as the database from which the deck was exported, neither keyword needs to be specified.
  - The values for the CODEPAGE keywords must be integers, such as 1252, 65001, and 936.
  - The LOCALE keywords' values will accept the following values only: **LEGACY**, **UTF8**, and **UTF-8**.
    - **LEGACY** is an alias for the local machine's code page, such as CP_ACP, 1252.
    - **UTF8** and **UTF-8** are aliases for a UTF-8 code page, such as CP_UTF8, 65001.
- Translated data must fit the field size limits.
- Legacy XPR headers in EBCDIC are automatically converted to ASCII.

Note: The keywords FROM_LOCALE, FROM_CODEPAGE, TO_LOCALE, and TO_CODEPAGE will not show up on the command line when the syntax/usage is queried; however, they are functional and, when specified, will work as designed.

Syntax:	`FILE=(,PREVIEW=YES/NO)(,TIME=OLD/NEW/MOD)(,REPLACE=YES/NO)(,FROMDOMA=)(,TODOMA=)`
Example:	Import all the classes in the file specified by FILE to a PRIMARY file specified in the `edmprof` file: `FILE=`*input_file*`,PREVIEW=NO`
Tip:	N/A

IMPORT_INSTANCE

The IMPORT_INSTANCE verb enables an administrator to import instance and resource data from an exported data set or output file (an *import deck*) to a location in the CSDB. The import deck will be imported to the CSDB PRIMARY File that is specified for the DBPATH setting in the MGR_ DIRECTORIES section of the `edmprof` file, such as:

`[MGR_DIRECTORIES]DBPATH =`
<InstallDir>`\ConfigurationServer\DB`

The entire process consists of the following phases.

Verify the Import Deck
This verification checks the internal integrity (such as, size, referential integrity, and validity of data) of the incoming deck.

Preview the Import Deck
This phase is a comprehensive *analysis* that scans the database and the entire import deck, and reports the results, detailing differences that are relevant to this import session. This analysis checks for duplicate object IDs (OIDs), determines if fixes are possible, and determines if a new deck is required. The entire import deck is analyzed before this phase completes.

Note: For the import deck, an *all-or-nothing* rule applies. Therefore, if an error condition is realized for one instance of the import deck, the entire deck is invalid.

When OIDs in the deck are the same as OIDs in the database, this verb’s behavior is dictated by the keywords REPLACE and CONTINUE. These keywords are discussed in detail.

Import the Instances and Resources
The instances and resources from the deck will be imported only if the integrity check (performed during the **Preview the Import Deck** phase) is free of errors.

The IMPORT_INSTANCE information is presented as follows:

- "Verb History" below describes when the keywords were introduced or discontinued.
- "Syntax" on next page details all the keywords that are associated with this verb. This includes their expected behavior, and their interactions with, and dependencies on, one another.
- "Retired Syntax" on page 160 covers the keywords that have been retired from use in this version, and includes information about which new keywords have replaced them.
- "Usage Considerations" on page 161 addresses some of the more noticeable and critical effects that might result from using this verb, as well as some of the **new features.**
- "Examples" on page 162 presents a few examples of how to express the keywords.

Verb History

This verb was introduced with version 4.3 of the CM Configuration Server to manage its database.

- The keywords, VERIFY and LOGFILE, were added to this verb’s functionality in version 4.4 of the CM Configuration Server.

- In version 4.5.2 of the CM Configuration Server:
  - The keywords, FROMDOMA, TODOMA, TIME, CHGCONS, FORCE, Y2K, and IMPORT_RESOURCE were removed from this verb's functionality—but continue to be supported. See the section, "Retired Syntax" on page 160.
  - The keywords, XPR, DUPLICATES, FORCE, CONTINUE, NEW, AUTOFIX, MAP_DOMAIN, and COMMIT_CHANGES were added to provide enhanced behavior management in the event of duplicate object IDs, multiple domains, and import decks from older systems and databases.

Syntax

This section details the syntax (**keywords** and **values**) that is associated with this verb, including the most efficient and effective ways to use it.

```
FILE=(,PREVIEW=
YES/NO)(,DUPLICATES=STOP/MANAGE)(,XPR=)(,NEW=)(,REPLACE=YES/
NO)(,VERIFY=YES/NO)(,AUTOFIX=YES/NO)(,MAP_DOMAIN=)(,CONTINUE=YES/
NO)(,COMMIT_CHANGES=YES/NO)(,LOGFILE=)
```

Note: The keywords FROM_LOCALE, FROM_CODEPAGE, TO_LOCALE, and TO_CODEPAGE will not show up on the command line when the syntax/usage is queried; however, they are functional and, when specified, will work as designed.

- If the deck is being imported into a *locale* or *codepage* that differs from the database from which it was exported, the appropriate keyword (TO_LOCALE or TO_CODEPAGE) must be specified. If the target *locale* or *codepage* is the same as the database from which the deck was exported, neither keyword needs to be specified.
- If the deck is being imported into a *locale* or *codepage* that differs from the database from which it was exported, the appropriate keyword (FROM_LOCALE or FROM_CODEPAGE) must be specified.
  If the target *locale* or *codepage* is the same as the database from which the deck was exported, neither keyword needs to be specified.
  - The values for the CODEPAGE keywords must be integers, such as 1252, 65001, and 936.
  - The LOCALE keywords' values will accept the following values only: **LEGACY**, **UTF8**, and **UTF-8**.
    - **LEGACY** is an alias for the local machine's code page, such as CP_ACP, 1252.
    - **UTF8** and **UTF-8** are aliases for a UTF-8 code page, such as CP_UTF8, 65001.
- Translated data must fit the field size limits.
- Legacy XPR headers in EBCDIC are automatically converted to ASCII.

Note: FILE is the only keyword that must be specified on the command line in order for this verb to execute.The other keywords are optional (as denoted by their inclusion in parentheses); if they are not specified, their defaults will be assumed and they will effect the processing of this verb.

- PREVIEW creates a preview listing of the input file contents, the expected results, and its ability to be imported. The results are written to the log file. The default is **YES**.

  > **Note:** To better understand the functionality of this keyword, think of it as asking, “Preview *only*?”
  > If PREVIEW=NO, the processing will run and the import deck data *can* be written to the database—depending on the other keywords and the results of the comparison.

  If PREVIEW=YES (the default), the only result is a log file being generated—no action is taken on the database.
  If PREVIEW=NO, the processing will run and, provided there are no errors, the import deck can be written to the database.

- FILE is the fully qualified path and filename of the file that contains the collection of instances for the import deck. This file is commonly suffixed with the extension **XPI**, as shown in the examples starting

  > **Note:** If a fully qualified path is not specified (as shown below), the location of the import file is assumed to be that from which ZEDMAMS is running.
  > ZEDMAMS VERB=IMPORT_INSTANCE,FILE=DB_001.XPI,PREVIEW=YES

- DUPLICATES enables an administrator to indicate the action to be taken when duplicate OIDs of instances are encountered (in the import deck and the database). The default is **STOP**.
  DUPLICATES=STOP (the default) will result in this operation stopping. The return code **8** will be reported.
  DUPLICATES=MANAGE will result in a new deck being created to avoid the re-use of a previously allocated OID. If the instance is data bearing, it can be corrected only if the resource is available, in which case, XPR must be specified.

  > **Note:** A new deck could be created for any of the following reasons: duplicate OIDs; domain change; OBJRCRC is NULL; duplication, re-mapping, or OID difference between source decks and target database.

- XPR is the fully qualified path and filename of the resource deck.

- REPLACE dictates the behavior of the process (as defined in the following conditions) when identical instances are discovered in the database and the deck. The default is **NO**.
  If REPLACE=YES, the data in the import deck can be written to the database.
  If REPLACE=NO (the default), the database and import deck will be queried for differences, and the following logic will apply.
  - If no differences are found, processing will continue with the next instance in the import deck.
  - If differences are found, the value of CONTINUE is checked.
    - If CONTINUE=NO, each instance with a difference will be ignored and processing will continue with the next instance in the import deck, but the process will not proceed to the next phase. A return code of **8** will be returned.
    - If CONTINUE=YES, each instance with a difference will be ignored and processing will continue with the next instance in the import deck.

- CONTINUE dictates the behavior of the process when matching records are discovered. The default is **NO**.

If CONTINUE=YES:

- For any class attribute found in the target class template, the import will continue as long as it doesn't result in the truncating of any significant (non-blank) data. In this case, the process will continue and an error will be reported.
- For any class attribute found in the target class template, but the import cannot import the data without truncating significant (non-blank) data, the process will continue, an error will be reported, and the import will fail.
- For any class attribute not found in the target class template, the field will be dropped—even if it contains significant (non-blank) instance data. (A warning or error message will be issued, indicating this occurrence.)

> **Note:** Fields and data that are dropped will be documented in the log file.

If CONTINUE=NO (the default) *and…*

- the class attribute is not defined in the target class template, the import will fail.

- VERIFY compares the date (ZOBJDATE) and time (ZOBJTIME) of incoming files with those of the database files, if specified as YES. The default is **NO**.
  If VERIFY=YES and…
  - the dates and times do not match (**rc=8**), a warning message is issued, and processing will continue with the next instance in the import deck.
  - the dates and times match and XPR was specified (with a valid value), the VERIFY_ IMPORT verb will run to check the integrity of the decks.
  - the dates and times match, but XPR has not been specified (or its value is invalid), verification is not possible.

  The results are reported to ZEDMAMS.LOG (the default), unless a different log has been specified for LOGFILE.

  > **Note:** If the dates and times match, the ZEDMAMS.LOG will report a successful verify; if not, the ZEDMAMS.LOG will report a failed verify. These verifications are done on a per-instance basis and reported at the end of the process.
  > If VERIFY=YES, PREVIEW=YES and REPLACE=YES are assumed—but nothing is imported to the database.

  If VERIFY=NO, no verification is done.

- NEW is the fully qualified path and filename prefix of the new decks that will be created. Optionally, just the filename prefix can be specified, in which case the new decks will be created in the current directory (by default, the `bin` directory).

  > **Note:** This keyword is applicable only if DUPLICATES=MANAGE.

  - If either the filename prefix or the fully qualified path have embedded blanks, the entire string must be enclosed in quotation marks (**" "**).
  - The decks will have the suffixes **.MPI** and **.MPR**.
  - The defaults are the fully qualified paths of **XPI** (as specified by the keyword, FILE) and **XPR** (as specified by the keyword, XPR), respectively.

- AUTOFIX dictates whether to delete orphaned resources. If importing a data bearing instance and the resource file exists in the database, the existing resource will be deleted and the incoming resource will be written to the database. The default is **NO**.

  Caution: This keyword should be used with extreme caution and only by an experienced administrator in a controlled manner. Incorrect use could result in the accidental removal of database elements that are critical to performance and operation.

  If AUTOFIX=YES, orphaned resources will be deleted.

- MAP_DOMAIN enables an administrator to import all instances from one domain into a different domain, thereby facilitating application management.
  Use MAP_DOMAIN=*source_domain*/*target_domain* to import all instances that originated in one domain, *source_domain*, into a domain with a different name, *target_domain*. Doing so triggers the creation of a new deck. All object IDs from a domain that matches *source_domain* are placed in the new deck. In the new deck, their domain value is replaced by that of *target_domain*.

  For example, import all instances of the SOFTWARE Domain to the domain, SOFTBACK, by specifying,

  ```
  MAP_DOMAIN=SOFTWARE/SOFTBACK
  ```

  Note: The new domain must exist in the database. If it doesn't, the process will stop.

- COMMIT_CHANGES dictates whether to commit to the database, the data in the import deck. The default is **YES**.
  COMMIT_CHANGES=**YES** (the default) will write the changes to the database.

  If COMMIT_CHANGES=NO, the changes will not be written to the database, but new **.MPI** and **.MPR** decks will be produced (if required).

  If there are no changes, this keyword is ignored.

- LOGFILE (see the section, "LOGFILE" on page 132).

Retired Syntax

As the EDMAMS verbs have evolved, changes have been made to enhance their processing. Because of this, and to maintain logic for the user, there have been changes to the syntax of some of the verbs.

This section details the keywords that have been retired from use for the IMPORT_INSTANCE verb.

Note: While retired, these keywords are still supported. These are superseded by new keywords where indicated.

- TIME=OLD (the default) retains the original OBJDATE, OBJTIME, and OBJID. (This is superseded by DUPLICATES=STOP.)

TIME=NEW generates a new OBJDATE, OBJTIME, and OBJID. (This is superseded by DUPLICATES=MANAGE.)

TIME=MOD generates a new OBJDATE and OBJTIME, but retains the original OBJID.

- FROMDOMA=<source_domain> and TODOMA=*target_domain* have been replaced by MAP_DOMAIN.
- FORCE=YES/NO. (This is superseded by CONTINUE.)
- CHGCONS specifies whether any embedded references to the name specified by FROMDOMA should be changed to the name specified by TODOMA.
- IMPORT_RESOURCE dictates whether this operation should import resources.

Usage Considerations

This section addresses some of the more noticeable and critical effects that might result from using this verb, as well as some of the new features.

- After the data from an import deck has been written to the CSDB, the changes are considered permanent. Therefore, it is imperative that a CSDB administrator using this verb be certain of the changes that are being considered.

  Note: It is recommended to
  Shut down the Configuration Server to ensure that the CSDB contents are not changing during the processing.
  Back up the CSDB before running this verb.
  Specify PREVIEW=YES and check the resulting log before committing any changes to the database.

- Executing this verb might result in the creation of new decks (**MPI**/**MPR**). The circumstances under which this might happen are:
  - There exist duplicate object IDs (ZOBJID) in the database instances and the import deck instances, and DUPLICATES=MANAGE.
  - There is a domain name change for the imported data (using the MAP_DOMAIN keyword).
  - The ZOBJRCRC (the object resource CRC) is NULL or empty and the value can be calculated and assigned in the process.

    Note: All of these occur in conjunction with the existence of the XPR deck, it being specified on the command line, and the values of the CONTINUE and REPLACE allowing the processing to continue.
    The XPR deck is necessary so that if changes are required, it is available to be updated at that time.

- This version of IMPORT_INSTANCE has more consistency checks that must be passed before the import deck is considered valid for import.
- Combining instances and resources on the same command line allows the import deck to be more thoroughly examined.
- Use the CONTINUE option to prevent data loss during import.

- In this version, an administrator:
  - Can manage the processing behavior if an import deck OID collides with a database OID.
  - Can specify the filename prefix if a new deck is generated.
  - Has a single keyword to facilitate changing domains.
- A new time-based format of object ID generation has been introduced, thereby eliminating the chance of randomly generating a duplicate OID. For more information on this feature, see the "MGR_STARTUP" on page 75.

Examples

The following examples offer a look at the ways this verb can be used.

Note: Even though some keywords are dependent on another, the order in which they are specified on the command line is not significant.
If a keyword is not specified on the command line, but has a default, the default will be assumed.
Some keywords, while optional, become mandatory based on the specifications of others and the results of processing. For an example, see DUPLICATES=MANAGE.

Example 1

Import the instance data that is contained in `DB_001.XPI` and `DB_001.XPR` to the PRIMARY file that is specified in the `edmprof` file. Do not write the changes to the database. Do not manage duplicate object IDs. Write the results to `C:\Temp\EDMAMS\DB001\Test01.log.`

```
ZEDMAMS VERB=IMPORT_INSTANCE,FILE=DB_001.XPI,XPR=DB_
001.XPR,PREVIEW=YES,LOGFILE=C:\Temp\EDMAMS\DB001\Test01.log
```

In this run, the implied values (defaults) that affected the processing are DUPLICATES=STOP and CONTINUE=NO.

Example 2

Import the instance data that is contained in **DB_001.XPI** and **DB_001.XPR** to the PRIMARY file that is specified in the `edmprof` file. Do not write the changes to the database. Manage any duplicate object IDs that are encountered. Query the database and deck for matching records and, if any are found, continue processing. Delete any orphaned resources that are encountered. Write the results to `C:\Temp\EDMAMS\DB001\Test02.log`.

```
ZEDMAMS VERB=IMPORT_INSTANCE,FILE=DB_001.XPI,XPR=DB_
001.XP-
R,PREVIEW=YES,DUPLICATES=MANAGE,CONTINUE=YES,AUTOFIX=YES,LOGFILE=
C:\Temp\EDMAMS\DB001\Test02.log
```

In this run, the implied value (default) that affected the processing is REPLACE=NO.

Example 3

Assume that the Example 2 command line has completed as specified. Execute the same run, but this time, write the changes to the database and the results to `C:\Temp\EDMAMS\DB001\Test02.log`.

```
ZEDMAMS VERB=IMPORT_INSTANCE,FILE=DB_001.XPI,XPR=DB_
001.XP-
R,PREVIEW=NO,DUPLICATES=MANAGE,CONTINUE=YES,AUTOFIX=YES,LOGFILE=
C:\Temp\EDMAMS\DB001\Test02.log
```

In this run, the implied values (defaults) that affected the processing are: COMMIT_
CHANGES=YES.

Note: The only difference between examples 2 and 3 is the value of PREVIEW=.

Example 4

Import the instance data (contained in **DB_001.XPI** and **DB_001.XPR**) from the SOFTWARE Domain to the SOFTBACK domain in the PRIMARY file in the database. Do not write the changes to the database. Do not manage duplicate object IDs. Query the database and deck for matching records and, and if any are found, continue processing. Delete any orphaned resources that are encountered. Write the results to `C:\Temp\EDMAMS\DB001\Test03.log`.

```
ZEDMAMS VERB=IMPORT_INSTANCE,FILE=DB_001.XPI,XPR=DB_001.XPR,MAP_
DOMAIN=SOF-
TWARE/SOFTBACK,PREVIEW=YES,REPLACE=NO,CONTINUE=YES,AUTOFIX=YES,
LOGFILE=C:\Temp\EDMAMS\DB001\Test03.log
```

In this run, the implied value (default) that affected the processing is DUPLICATES=STOP.

Example 5

Import the instance data that is contained in **DB_001.XPI** and **DB_001.XPR** to the PRIMARY file that is specified in the `edmprof` file. Write the changes to the database. Do not query the database and deck for matching records. Write the results to `C:\Temp\EDMAMS\DB001\Test04.log`.

```
ZEDMAMS VERB=IMPORT_INSTANCE,FILE=DB_001.XPI,XPR=DB_
001.XP-
R,PREVIEW=NO,REPLACE=YES,LOGFILE=C:\Temp\EDMAMS\DB001\Test04.log
```

In this run, the implied values (defaults) that affected the processing are COMMIT_
CHANGES=YES, DUPLICATES=STOP, and CONTINUE=NO.

Import and Export Files

"Import and Export Files" above presents a list of the six default import/export files that are generated by the CSDB. Their level in the CSDB is part of the logic in their naming.

Import and Export File Names

Database Level	Import File Name	Export File Name
Class	`.MPC` – modified class file	`.XPC` – exported class file
Instance	`.MPI` – modified instance file	`.XPI` – exported instance file
Resource	`.MPR` – modified resource file	`.XPR` – exported resource file

Note: The export files (`XPI` and `XPR`) are the original decks that might be generated during the export process, and are the files that are imported either back into the existing database or into another database. So, each `XPI` and `XPR` file can be an export *and* import file.
The `MPI` and `MPR` files are decks that are generated during a database import that resulted in correcting duplicate OID issues, changing domains, or correcting empty or NULL OBJRCRCs.

IMPORT_RESOURCE

This verb imports resource data from an exported data set or file to a RESOURCE File as specified in the `edmprof` file.

The keyword, VERIFY, was added to this verb's functionality in version 4.4 of the Configuration Server.

- If REPLACE=NO, the class template will not be replaced.
- If VERIFY=YES, an implied PREVIEW=YES is set.
  If a resource exists, its size is compared to the size of the incoming resource to verify that they match.
- If the deck is being imported into a *locale* or *codepage* that differs from the database from which it was exported, the appropriate keyword (TO_LOCALE or TO_CODEPAGE) must be specified.
  If the target *locale* or *codepage* is the same as the database from which the deck was exported, neither keyword needs to be specified.
- If the deck is being imported into a *locale* or *codepage* that differs from the database from which it was exported, the appropriate keyword (FROM_LOCALE or FROM_CODEPAGE) must be specified.
  If the target *locale* or *codepage* is the same as the database from which the deck was exported, neither keyword needs to be specified.
  - The values for the CODEPAGE keywords must be integers, such as 1252, 65001, and 936.
  - The LOCALE keywords' values will accept the following values only: **LEGACY**, **UTF8**, and **UTF-8**.
    - **LEGACY** is an alias for the local machine's code page, such as CP_ACP, 1252.
    - **UTF8** and **UTF-8** are aliases for a UTF-8 code page, such as CP_UTF8, 65001.

- Translated data must fit the field size limits.
- Legacy XPR headers in EBCDIC are automatically converted to ASCII.

Note: The keywords FROM_LOCALE, FROM_CODEPAGE, TO_LOCALE, and TO_CODEPAGE will not show up on the command line when the syntax/usage is queried; however, they are functional and, when specified, will work as designed.

Syntax:	`FILE=(,FROMDOMA=)(,TODOMA=)(,PREVIEW=YES/NO)(,REPLACE=YES/NO)(,VERIFY=YES/NO)`
Example:	Import all resources in the file specified by FILE to a RESOURCE file specified in `edmprof`: `FILE=RESOURCE,PREVIEW=NO`
Tip:	N/A

LIST_CLASSES

This verb displays a list of class names, object IDs, and other 60-byte prefix information, such as ZOBJDATE, ZOBJTIME, persistence flag, sequence sensitive flag, Distributed Configuration Server flag and db type and count totals.

Syntax:	`FILE=,DOMAIN=`
Example:	List all classes in the PRIMARY file: `DOMAIN=*`
Tip:	N/A

LIST_CONNECTS

This verb displays a list of connect-to values (type C) for the specified instances.

- INSTANCE can be partially specified.
  To display only one instance, the entire name must be specified.

Syntax:	`DOMAIN=,CLASS=(,INSTANCE=)`
Example:	List all of the connect-to values in the ZSERVICE Class of the SOFTWARE Domain: `DOMAIN=SOFTWARE,CLASS=ZSERVICE`
Tip:	To list data for all type C values in all instances of the specified class, omit INSTANCE.

LIST_CONS_VARS

This verb automatically displays a list of connect-to data (type C) and, optionally, variable data (type V) for the specified instances.

- Wildcards (*) can be specified in INSTANCE.
  For example, specify DIFF* to select all the instances that contain DIFF as the first part of the string.

  To display only one instance, the entire name must be specified.
- If VTYPE=YES, variable data will be included in the display.

Syntax:	`(FILE=,)DOMAIN=,CLASS=(,INSTANCE=)(,VTYPE=YES/NO)`
Example:	From the USER Class in the SOFTWARE Domain, list the connect-to and variable values only for those instances prefixed with DIFF: **`DOMAIN=SOFTWARE,CLASS=ZSERVICE,INSTANCE=DIFF,VTYPE=YES`**
Tip:	To list all C- and V-type data in all the instances of the specified class, omit INSTANCE.

LIST_DOMAINS

This verb displays an alphabetical list of domains for a specified file (the default is the PRIMARY file).

- FROMDOMA is the domain from which to start the list of domains.
  If omitted, all the domains through the TODOMAIN will be listed.
- TODOMAIN is the domain at which to end the list of domains.
  If omitted, all the domains following FROMDOMA will be listed.

Syntax:	`FILE=(,FROMDOMA=)(,TODOMAIN=)`
Example:	In the PRIMARY file, list all of the domains that follow the domain ACCT: `FILE=PRIMARY,FROMDOMA=ACCT` In the PROFILE File, list the domains in the range ACCT through SALES (inclusive): `FILE=PROFILE,FROMDOMA=ACCT,TODOMAIN=SALES`
Tips:	To list all the domains of the selected file, simply omit FROMDOMA and TODOMAIN. To list one domain, specify it for FROMDOMA and TODOMAIN.

LIST_FLAGS

This verb displays the attribute name, length, type, and Configuration Server and RCA agent flags for a specific class template.

- Specify README=YES to include the README attributes.

Syntax:	`DOMAIN=,CLASS=(,README=YES/NO)`
Example:	List the attribute information for the attributes of the ZSERVICE Class of the SOFTWARE Domain and omit the README attributes: `DOMAIN=SOFTWARE,CLASS=ZSERVICE`
Tip:	N/A

LIST_INST_DATA

This verb displays, in a concise format, the attribute data of the specified instances.

- Wildcards (*) can be specified for INSTANCE.
  For example, specify DIFF* to select all the instances that contain DIFF as the first part of the string.

  To display only one instance, the entire name must be specified.
- Use FIELDS to specify up to six attribute names.
- Specify the fields with a space separating each name, and the entire string enclosed in quotation marks, as in:
  "field1 field2 field3 field4 field5 field6"

  To list all attribute data of all instances of the specified class, omit FIELDS.

Note: If FIELD is omitted, all attribute data of all instances of the specified class will be displayed. This might produce a very large log, which might hinder locating data.

If a fieldname does not exist in the template, it will be ignored.

Syntax:	`DOMAIN=,CLASS=(,INSTANCE=)(,FIELDS=)`
Example:	From the ZSERVICE Class in the SOFTWARE Domain, list the attribute data of all instances with the prefix, RAD: `DOMAIN=SOFTWARE,CLASS=ZSERVICE,INSTANCE=RAD*`
Tip:	To list all instances of the specified class, omit INSTANCE.

LIST_INSTANCE

This verb displays a list of instance names and object IDs for the class specified. It also displays the ZOBJTIME and resource size.

- The value of FROMINST can be partially specified.
  For example, specify FROMINST=DIFF to select all the instances that contain DIFF as any part of the string.

  Use a wildcard (*) to specify this value as a prefix, as in RAD*.
- The value of SUFFIX can be partially specified.
  For example, specify SUFFIX=INT to select all the instances that have INT as a suffix.

Syntax:	`DOMAIN=,CLASS=(,FROMINST=)(,SUFFIX=)`
Example:	From the USER Class of the POLICY Domain, list the instance names and object IDs for all instances with the prefix RAD, and all instances with the suffix, PORT: `DOMAIN=POLICY,CLASS=USER,FROMINST=RAD*,SUFFIX=PORT`
Tip:	To list all instances, omit FROMINST.

LIST_PACKAGE

This verb lists the instances and all mated components of the PACKAGE class.

- The value of INSTANCE can be partially specified.
  For example, specify INSTANCE=DIFF to select all the instances that contain DIFF as any part of the string.

  Use a wildcard (*****) to specify this value as a prefix, as in, RAD*; and as a suffix, as in, *INT.

Syntax:	`(FILE=,)DOMAIN=,INSTANCE=`
Example:	From the SOFTWARE Domain, list all instances with the prefix RAD: `DOMAIN=SOFTWARE,INSTANCE=RAD*`
Tip:	N/A

Note: CLASS is not an option because this verb applies to the PACKAGE class only.

LIST_PREFIX

This verb displays data from the Distributed Configuration Server prefix.

If CLASS is omitted, all class prefixes will be displayed.

Syntax:	`(FILE=,)DOMAIN=(,CLASS=)`
Example:	List the Distributed Configuration Server prefixes for all classes in SOFTWARE Domain of the PRIMARY file: `DOMAIN=SOFTWARE`
Tip:	N/A

LIST_RESOURCES

This verb displays a list of resource names (with promote dates, times, data names, data sizes, and object IDs) from the PRIMARY File.

As of version 4.3 of the Configuration Server Database, this verb was renamed. Its original name was LIST_RESOURCE.

- If there is no mated instance in the PRIMARY file, an appropriate message will be generated.
- If ORPHANS=YES, only the orphans will be listed.
- If CHKSIZE=YES, the size listed (from ZOBJRSIZ) is compared to the actual size (from the NvdDBFind).
  Only those resources that have a size anomaly will be listed.
- SIZE is the size of the resource.
  SIZE must be specified as a range (not a single byte size), and the range must be defined with a dash (**-**), as in, `100-500`.

Syntax:	`DOMAIN=,CLASS=(,ORPHANS=YES/NO)(,CHKSIZE=YES/NO)(,SIZE=`*nnnn-nnnn*`)`
Example:	From the FILE Class of the SOFTWARE Domain, list all resources that are between 64 and 1024 bytes in length, and compare this with the actual size: **`DOMAIN=SOFTWARE,CLASS=FILE,CHKSIZE=YES,SIZE=64-1024`**
Tip:	N/A

LIST_ZRSC_FIELDS

This verb displays the data in all fields that begin with **ZRSC**, such as ZRSCSIZE and ZRSCVRFY. This verb allows wildcards for INSTANCE.

INSTANCE can be specified with a partial name. For example, specify INSTANCE=DIFF to select all the instances that contain DIFF as any part of the string.
Use a wildcard (*) to specify this value as a prefix, as in, RAD*; and as a suffix, as in, *INT.
To display only one instance, the entire name must be specified.

Syntax:	`DOMAIN=,CLASS=(,INSTANCE=)`
Example:	From the ZSERVICE Class of the SOFTWARE Domain, list the instances that begin with CICS: `DOMAIN=SOFTWARE,CLASS=ZSERVICE,INSTANCE=CICS*`
Tip:	To list all instances of the specified class, do not specify INSTANCE.

MATCH_RESOURCES

This verb will compare resource data from the RESOURCE file against instance names from the PRIMARY file to determine and display whether those resources are mated (orphaned). This comparison is made by reading the resource data, extracting the instance name from the resource prefix, and attempting to find the mated resource.

The keyword, PREVIEW, was added to this verb as of version 4.3 of the Configuration Server Database.

- If PREVIEW=NO, and there is resource data that has been determined to be orphaned, a search of the PRIMARY file instances will occur, to locate a matching instance object ID.
  If a match is made, the instance name is placed in the resource data prefix and the resource is updated, with the time stamp for the resource data being updated with the ZRSCDATE and ZRSCTIME.
- Totals at completion include resources found, as well as total resources (mated and orphaned).
- When using PREVIEW=NO, a great deal of I/O might occur.

Syntax:	`DOMAIN=,CLASS=(,PREVIEW=YES/NO)`
Example:	Match and display resource data names for all the SOFTWARE.FILE Instances: `DOMAIN=SOFTWARE,CLASS=FILE,PREVIEW=YES`
Tip:	N/A

If many orphans are detected with PREVIEW=YES, a more efficient way to update the resource data file is to use the verb "ZRSOURCE_UNMATES" on page 176.

PACKAGE_UNMATES

This verb lists all PACKAGE class instances that do not have mated components in the domain that is specified.

- DOMAIN must be one that has a PACKAGE class.
- CLASS defaults to **PACKAGE**.

Syntax:	`DOMAIN=(,CLASS=)`
Example:	From the SOFTWARE Domain, list all the PACKAGE class instances that have no mated components: `DOMAIN=SOFTWARE`
Tip:	N/A

REFRESH_DMA

This verb will recount all the instances, classes, and domains in the PRIMARY file and, optionally, refresh the **count** (TotalInstanceCount, TotalClassCount, and TotalDomainCount) and **date** (LastUpdateDate, LastInstanceUpdateDate, and LastClassUpdateDate) fields in the appropriate Distributed Configuration Server prefix areas. Additionally, the updated output can be displayed in the log.

This verb was introduced with version 4.2 of the CM Configuration Server. It replaced the verb, REFRESH_COUNTS.

The keywords, DOMAIN and CLASS, were added with version 4.3; and the keyword, COUNTS_ ONLY, was added with version 4.5.2.

- If PREVIEW=YES, a count of instances by class, domain, and file will be listed to the log, but no data will be written.
  The log will display the actual count (as calculated by running this verb) and the current count (current values in the total count fields [mentioned in the introductory paragraph] in the database) in the Distributed Configuration Server prefix. If the actual count differs from the current count, the latter will be flagged with an asterisk (*****).
- If PREVIEW=NO, the TotalInstanceCount, TotalClassCount, and TotalDomainCount (for each applicable Distributed Configuration Server prefix) will be computed and updated, in addition to updating the current counts.
- If COUNTS_ONLY=**NO** (the default) and PREVIEW=NO, the current count and date fields in the Distributed Configuration Server area for each class will be updated.
  If COUNTS_ONLY=NO and PREVIEW=YES, the current count and date fields in the Distributed Configuration Server area for each class will be displayed.
- If COUNTS_ONLY=YES and PREVIEW=NO, the current count fields only will be updated. This is effective for very large database files because, by not refreshing the date fields and not having to read every instance in the database, the function executes in less time.

If COUNTS_ONLY=YES and PREVIEW=YES, the current count and actual count fields will be displayed, but not updated; the date fields are not touched.

- Each class for each domain will be previewed separately, and at the end of each domain, a domain summary will be presented.
- After the last domain, a file summary will be presented by listing the ZBASE.ZBASE template information.

Syntax:	`PREVIEW=YES/NO(,DOMAIN=)(,CLASS=)(,COUNTS_ONLY=YES/NO)`
Example:	Preview the counts and update dates in the Distributed Configuration Server prefix for the entire PRIMARY file: `PREVIEW=YES`
Tip:	N/A

RENAME_INSTANCE

This verb will rename instances and the internal name of any mated resource data.

- KEEP indicates whether the old instance will be deleted.
- OLDPREFIX specifies existing instances that are to be renamed.
  If a single instance is to be renamed, specify the entire name.

  If multiple instances are to be renamed, a wildcard (*****) is required.

  For example, to change the names of the instances, **east_sales** and **north_sales** to **US_Sales**, specify **OLDPREFIX=*_sales,NEWPREFIX=US_Sales**.
- NEWPREFIX is that which will replace OLDPREFIX.

Syntax:	`DOMAIN=,CLASS=,OLDPREFIX=,NEWPREFIX=(,KEEP=YES/`**`NO`**`)(,PREVIEW=`**`YES`**`/NO)`
Exam-ple:	In the ZSERVICE Class of the SOFTWARE Domain, rename all instances prefixed with EAST to NORTH_EAST, and delete the old prefix: **`DOMAIN=SOF-TWARE,CLASS=ZSERVICE,OLDPREFIX=EAST,NEWPREFIX=NORTH_EAST,PREVIEW=NO`**
Tip:	N/A

SEARCH_INSTANCES

This verb will search the specified instances for the data contained in STRING.

- The data specified is not case-sensitive; if it contains embedded spaces, it must be enclosed in quotation marks (**“ ”**).
- The output log will contain the name of the instance, attribute, and the specified STRING value.
- If the value of STRING is not found in the specified instances, this will be reported in the log.

- FROMINST can be specified with a partial name.
  For example, specify FROMINST=DIFF to select all the instances that contain DIFF as any part of the string.

  Use a wildcard (*) to specify this value as a prefix, as in, RAD*; and as a suffix, as in, *INT.

  To display only one instance, the entire name must be specified.

Syntax:	`DOMAIN=,CLASS=(,FROMINST=,)STRING=`
Example:	Search for the string “J. Q. Public” in all instances that end in EAST in the USER Class of the SOFTWARE Domain: `DOMAIN=SOFTWARE,CLASS=USER,STRING="J. Q. Public",FROMINST=*EAST`
Tip:	To list all instances of the specified class, omit FROMINST.

SORT_OBJECT_ID

This verb will sort object IDs within a Domain.

- FILE will default to **PRIMARY** if it is omitted or if no value is specified.
- DOMAIN can be a single Domain (*domain_name*) or all Domains (ALL) of the specified File.
  To sort the object IDs of multiple (but not all) Domains in a File, execute this verb once for each Domain.
  To sort the object IDs of multiple Domains in multiple Files, execute this verb once for each Domain.
- ORDER=NOSORT means that the object IDs will be listed in CLASS.INSTANCE groups.
  ORDER=ASCEND/DESCEND specifies the order of sorting, based on object IDs.
  Duplicate object IDs will be flagged in ascending or descending order.
- ALLIDS is valid only if ORDER=ASCEND or DESCEND.
  ALLIDS=YES will duplicate (list all object IDs) the entire File.
  ALLIDS=NO will list only duplicate object IDs.

Syntax:	`(FILE=,)DOMAIN=(ALL/DOMAIN)(,ORDER=ASCEND/DESCEND/NOSORT)(,ALLIDS=YES/NO)`
Example:	Sort, in descending order, all the object IDs of all domains, and duplicate the entire file: `DOMAIN=ALL,ORDER=DESCEND,ALLIDS=YES`
Tip:	All domains can be selected by specifying `DOMAIN=ALL`.

SYNC_CLASS

This verb will synchronize an existing (target) class with a newly formatted (source) class, and re-organize all existing class instances according to the mapping in the source class template.

- All target class attributes that have a match in the new template will adopt the characteristics of that matching attribute.

Any target class attributes that do not have a match in the new template will be deleted.

- TODOMAIN specifies the class that contains the target class template that is to be synchronized.
- SYNCDOMA specifies the class that contains the source class template. The default is **ZEDMSYNC**.
- CLASS is the target class (within the domain that is specified by TODOMAIN *and* SYNCDOMA) that will be synchronized.
  The value of CLASS must exist in the TODOMAIN and SYNCDOMA domains; if it doesn't, the function will fail.
- CACHE=YES will update loaded cache when running as a Configuration Server method.
- If BASE=YES, this verb will copy the _BASE_INSTANCE_ from the source (SYNCDOMA) to the target (TODOMAIN).

Syntax:	`TODOMAIN=(,PREVIEW= YES/NO)(,SYNCDOMA=ZEDMSYNC,)CLASS=(,CACHE=YES/ NO)(,BASE=YES/NO)`
Example:	In the POLICY Domain, synchronize the existing USER Class with a newly formatted USER Class, imported from ZEDMSYNC. Include the _BASE_ INSTANCE_ and update the loaded cache: `TODOMAIN=POLICY,CLASS=USER,PREVIEW=NO,CACHE=YES,BASE=YES`
Tip:	N/A

UPDATE_INSTANCES

This verb will update instances in the specified domain from data in an edited text file, INFILE. INFILE must conform to the following specifications.

- The first line must contain a new instance name (max. 32 bytes) starting in column 1.
  The next line contains the variable field name (max. eight bytes) starting in column 2; then a blank in column 10; a 1-byte index value (1-9) in column 11; a blank in column 12; and the data to be populated beginning in column 13.
- REPLACE=YES specifies that an attribute that contains existing data (non-blank) be overlaid.
- Use a slash-asterisk combination (**/***) in columns 1 and 2 to denote the end of instance data (see lines 5 and 10 below).

Column Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Line 1	U	S	E	R	_	N	A	M	E															
Line 2		N	A	M	E								J	.	A	.	D	E	V	E	L	O	P	
Line 3		E	D	M	S	E	T	U	P		5		5	T	H	A	T	T	R	I	B	U	T	E
Line 4		Z	P	R	I	O	R	I	T				9	9	9									

Line 5	/	*																						
Line 6	U	S	E	R	_	N	A	M	2															
Line 7		N	A	M	E								L	.	Z	I	M	M	E	R				
Line 8		E	D	M	S	E	T	U	P		2		2	N	D	A	T	T	R	I	B	U	T	E
Line 9		E	D	M	S	E	T	U	P		3		3	R	D	A	T	T	R	I	B	U	T	E
Line 10	/	*																						
Line 11	U	S	E	R	_	N	A	M	E	_	3													
Line 12		N	A	M	E								J	I	M		D	O	E					
Line 13		E	D	M	S	E	T	U	P		4		4	T	H	A	T	T	R	I	B	U	T	E

Syntax:	`INFILE=,DOMAIN=,CLASS=(,REPLACE=YES/NO)`
Example:	Update the instances in the MYINPUT.TXT file in the POLICY Domain's USER Class: `INFILE=MYINPUT.TXT,DOMAIN=POLICY,CLASS=USER`
Tip:	Run LIST_INST_DATA or LIST_FLAGS to determine the index of a like named variable to be updated.

UPDATE_MGRIDS

This verb updates the specified Manager ID, Manager name, managing Manager ID, and managing Manager name in the Distributed Configuration Server prefix.

Note: The Configuration Server was previously called the *Manager*.
Therefore, the keywords associated with this verb retain the 'Manager' designation, as in, Manager name (MNAME).

- All keywords are optional. However, at least one keyword other than DOMAIN and CLASS must be specified to avoid a usage error being displayed to STDERR.

Syntax:	`(FILE=)(,DOMAIN=)(,CLASS=)(,MID=)(,MMID=)(,MNAME=)(,MMNAME=)`
Example:	Update the managing Configuration Server IDs and managing Configuration Server names in the USER Class of the POLICY Domain in the PRIMARY file: `FILE=PRIMARY,DOMAIN=POLICY,CLASS=USER,MMID=010,MMNAME=New_ Mngng_RCS`
Tip:	To update an entire file, omit DOMAIN and CLASS.

VERIFY_CLASS

This verb will display class templates, as specified, to determine if any gaps, overlays, or other anomalies exist. The output log will consist of a template entry number, the attribute name, its length, and its displacement in the heap. If an error is found, it will be indicated in the appropriate place.

- DOMAIN must be specified.
  All domains can be selected by specifying the value as ALL.
- CLASS must be specified.
  All classes can be selected by specifying the value as ALL.

Syntax:	`(FILE=,)DOMAIN=,CLASS=`
Example:	In the POLICY Domain of the PRIMARY file, verify all the class templates: `FILE=PRIMARY,DOMAIN=POLICY,CLASS=ALL` Verify the ZSERVICE Class templates in the PRIMARY file: `DOMAIN=ALL,CLASS=ZSERVICE`
Tip:	N/A

VERIFY_DATABASE

This verb will validate the integrity of a CSDB. It can be used at any time to check database integrity, and can run standalone or as a Configuration Server method. This database validation must be done in read-only mode.

This verb was introduced with version 4.5.1 of the CM Configuration Server.

- DOMAIN can be a single domain or all domains of the CSDB. The default is ALL.
- DEPTH can be DOMAIN, CLASS, INSTANCE, or RESOURCE. The default is RESOURCE.
- ZFILE is the name of the output file, which contains suggested actions to be carried out when there are errors in the database. This can be used as input file for running ZFILE option of ZEDMAMS.
- LARGEDB=YES, will use ‘file based storage’ for verify database processing rather than using limited ‘process virtual memory’. This option is added to support verify database of very large database with millions of instances. The default is NO.
  When this option is used, a temporary file with the format ‘zedmams_ldb_<datatime>.dat’ is created for processing and automatically deleted after the processing is over.
- STMPPATH is the path used for creating the temporary file, with the option LARGEDB=YES. The temporary file default location assumed on Windows is the bin folder in which the zedmams.exe utility is running.

To verify the integrity of multiple (but not all) domains in the CSDB, execute this verb once for each domain.

Note:

- This is an exhaustive check of the database and, as such, might take a long time to run, possibly several hours. It makes two passes over the database – first, checking the PRIMARY file and any associated resources; and second, checking the RESOURCE file for orphans. As a result of the two passes, it has sufficient information to do a check for duplicate object IDs.

- If you verify a very large database by running ZEDMAMS command with verb VERIFY_DATABASE and DEPTH=RESOURCE / INSTANCE, you can set the OBJECTID_VERDB_INITIAL_ENTRIES parameter in MGR_CACHE section of the `edmprof.dat` file, to avoid memory allocation errors. Set the value for the OBJECTID_VERDB_INITIAL_ENTRIES parameter to approximately 1.2 times the total number of instances in the database.
- If you have a very large database (with millions of instances) and memory allocation errors still occurs after the above suggested step. Use the option LARGEDB=YES, which will use 'file based storage' for processing rather than using limited 'process virtual memory'.

Syntax:	`DOMAIN=(`**ALL**`/`*any_domain*`), (DEPTH=), (ZFILE=), (LARGEDB=YES/`**NO**`), (STMPPATH=`*file_path*`)`
Example:	Verify the integrity of the CSDB SOFTWARE Domain: `DOMAIN=SOFTWARE`Verify the integrity of all domains in the database: `DOMAIN=ALL`Verify the integrity of all domains in the database using LARGEDB option: `DOMAIN=ALL,DEPTH=RESOURCE,LARGEDB=YES,STMPPATH=D:\temp`
Tip:	Specify DOMAIN=ALL to select all domains in the CSDB.

ZRSOURCE_UNMATES

This verb will match Instances in the PRIMARY File with the resource data in the RESOURCE file, based on the specified Domain and Class, and the object ID from the PRIMARY File instances.

- If resource data is not found, the Instance is considered *unmated*.
- If resource data is found, the Instance name in the resource data prefix is compared with the Instance name from the PRIMARY File.
  If it does not match, it is reported in the log as an inconsistency.
- If PREVIEW=NO, the resource prefix is updated to reflect the true Instance name from the PRIMARY file.
- At completion, totals are listed in the log to indicate the number of unmated Instances, inconsistencies, and so forth.
- DOMAIN is that which contains resource data, usually SOFTWARE.
- If DEBUG=YES, all Instances are listed as they are verified.

Syntax:	`DOMAIN=,CLASS=(,PREVIEW=YES/NO)(,DEBUG=YES/NO)`
Example:	In the FILE Class of the SOFTWARE Domain in the PRIMARY File, list all unmated Instances and inconsistencies: `PREVIEW=YES,DOMAIN=SOFTWARE,CLASS=FILE,DEBUG=YES`
Tip:	N/A

Chapter 5

Configuration Server Database Utility (RadDBUtil)

Caution: It is recommended to create a back up the Configuration Server Database before executing any of the commands that are shown in this chapter.

Introduction

RadDBUtil (`raddbutil.exe`) is the Configuration Server Database tool that manages:

- Configuration Server Database updates (imports, exports, and deletions),
- Configuration Server communications,
- activity logging, and
- version queries.

This chapter focuses on the functionality, syntax, and common-use attributes of this tool, and provides examples of each of these capabilities.

Caution: It is recommended to create a back up the CSDB before executing any of the commands that are shown in this chapter.

Note: The Configuration Server does not need to be running in order for RadDBUtil to run. Additional information is detailed in the section, "Running RadDBUtil from a Command Line" on next page.

Components & Processes

This section details the components and processes that benefit from using RadDBUtil.

Components

- Configuration Server
- Configuration Server Database
- Distributed Configuration Server

Processes

- Importing/exporting to/from the CSDB
- Output materials produced by, or during, importing and exporting
- Deleting instances and resources from the CSDB
- Querying and manipulating the Configuration Server lock status
- Version information queries
- Activity logging

Running RadDBUtil from a Command Line

This section provides information regarding running RadDBUtil from a command line.

> **Note:** It is important to note that RadDBUtil must be able to lock the CSDB, which it cannot do when it is run from a command line if:
> The Configuration Server is running
> *and*
> MANAGER_TYPE=STANDALONE.

To run RadDBUtil from a command line, do either (or both) of the following *before* running RadDBUtil.

- Shut down the Configuration Server.
- Open the `edmprof` file and, in the MGR_STARTUP section, set MANAGER_TYPE to something other than STANDALONE—either DISTRIBUTED or SERVER.

Implementation Details

RadDBUtil has a dependency on the `edmprof` file, and should be placed in the same directory as it—typically the Configuration Server `bin` directory on Windows, RadDBUtil must be able to find a valid CSDB and the Configuration Server `log` directory.

> **Caution:** It is recommended to create a back up the CSDB before executing any of the commands that are shown in this chapter.

Radia Client Automation Patch Manager Considerations

This section contains important information and warnings about using RadDBUtil to import and export Radia Client Automation Patch Manager (Patch Manager) bulletins.

Note: Important information regarding the deleting of Patch Manager bulletins is detailed in the section, "Deleting Bulletins from a Database" on page 189.

IMPORT

Caution: When using the `-domain` switch, all instances and resources (of the XPI file that is specified for INPUT) will be imported to the target domain. Therefore, it is imperative that the necessary domains and classes exist in the target domain.

EXPORT

Note: In order for the RadDBUtil tool to successfully export bulletins, in the PATCHMGR.ZSERVICE Class template there must be an attribute named SYNC of the type CONNECTION with the appropriate default value.
See "EXPORT" on page 186 for an example of the SYNC-CONNECTION attribute being specified for a Patch Manager bulletin.
If this attribute is not present in the PATCHMGR.ZSERVICE Class template, it must be added.

EDMPROF File Settings

The following edmprof file settings affect the operation of RadDBUtil. Be sure to verify these settings and adjust them accordingly.

MGR_LOG.DIRECTORY

specifies where the activity and audit logs will be generated. For more information, see "Output Files" on page 185

MGR_DIRECTORIES.DBPATH

determines the CSDB location.

MGR_STARTUP.MGR_ID

is the unique, three-character identifier of the Configuration Server that was specified during its installation.

MGR_STARTUP.MGR_NAME
is the identifying name of the Configuration Server that was specified during its installation.

RadDBUtil Verbs

In this section, each of the six RadDBUtil verbs is detailed with syntax options, syntax descriptions, and verb-specific considerations.

General Syntax

- The executable, verbs, keywords, and values are not case-sensitive.
- The verbs are "VERSION" below, "LOG" below, "IMPORT" on next page, "EXPORT" on page 186, "DELETE" on page 188, and "RCS" on page 190.
- The verbs can be specified in either of the following formats:
  `-keyword` *value*

  If this syntax is used, pairs of combinations must be separated by a space, as in:

  `-keyword` *value* `-keyword` *value* `-keyword` *value*

  or

  `keyword=`*value*

  If this syntax is used, pairs of combinations must be separated by a space, a comma, or both, as in:

  `keyword=`*value* `keyword=`*value*`,keyword=`*value*`, keyword=`*value*

Note: A double-dash (--) indicates the end of the options.

The acceptable Boolean values are:

TRUE: 1, YES, ON, and TRUE

FALSE: 0, NO, OFF, and FALSE

VERSION

The VERSION verb produces build information for the RadDBUtil tool and all embedded executables. For examples of the syntax, see "Examples" on page 191.

Syntax

The syntax of the verb, VERSION, is shown below.

```
Raddbutil version
```

LOG

This verb places RadDBUtil-specific messages in the audit log and activity log.

The audit log information is:

BuildInfo	CommandLine
CompletionCode	Timestamp (ISO)

There are two messages per command—one when RadDBUtil begins and one when it ends.

```
20050126 15:24:04 C:/RadDBUtil/raddbutil.exe 60 --> import20050126
15:24:04 C:/RadDBUtil/raddbutil.exe 60 <-- rc=8
```

Syntax

The syntax of the LOG verb is shown below. For examples of the syntax, see "Examples" on page 191.

`RadDBUtil LOG "`*record this information"*

The RadDBUtil entry in the log can be accompanied by customized text.

The text must be enclosed within quotation marks, as shown above. For example, to delineate a nightly log entry for a specific date, specify:

```
RadDBUtil log "Nightly Log for June 22, 2005"
```

IMPORT

Caution: It is recommended to create a back up the CSDB before executing any of the commands that are shown in this section.

The IMPORT verb simplifies the importing of materials into the CSDB. It can import materials from one domain into a domain of a different name within a CSDB, and from one CSDB to another. It offers the following options to optimize the import operation.

- Automatically recognize and re-use the instances that exist in the target domains, and
- Import only those elements that do not exist in the target domains.

Additionally, by having a feature that allows the IMPORT verb to identify packages and dialogs that exist in the database and to dynamically adapt to them, RadDBUtil reduces the size of the targeted domain—as well as Distributed Configuration Server execution times—without affecting the integrity of the imported materials. Other features of the IMPORT verb are:

- Parameter validation
- Database receptiveness to receiving the import materials
- Deck verification (XPI and XPR)
- Attributes that are dropped from the import operation will be noted with a warning level (**--?**) in the activity log.

Note: For the IMPORT verb, an "all-or-nothing" rule applies—that is, if RadDBUtil *cannot* do *all* of that which is requested, it will do *none* of that which is requested.
If a RadDBUtil IMPORT operation fails, the Configuration Server log will have an entry reflecting this and the return code will be rc=8.

Syntax

The syntax of the IMPORT verb is shown in this section. For examples of the syntax, see "Examples" on page 191.

The keywords and values are not case-sensitive, as can be seen in the following examples.

Note: Optional keyword-value combinations are in parentheses. Default values are underlined.

```
Raddbutil import –input FileName (-output value) (-domain value) (-
commit false) (-root *) (-accept a) (-reject u+d) (-ignore s)
```

```
raddbutil IMPORT INPUT=value(,OUTPUT=value(,DOMAIN=domain_name
{:REUSE})-
(,COMMIT=TRUE/FALSE)(,ROOT=*)(,ACCEPT=A)(,REJECT=U+D)(,IGNORE=S)
```

IMPORT Keywords

The following table lists and defines the keywords for the verb, IMPORT.

RadDBUtil IMPORT Keywords

Keyword	Explanation
INPUT	This is the prefix (the fully specified drive and path) of the input files. • If a fully specified drive and path is not provided with the input file name, the current directory is used. • If `.xpi` is specified, it will be stripped off. • This keyword is mandatory; it does not have a default value. • The associated resource file must be in the same location and have the same name, but with the extension `.XPR`. This is the only location that will be searched if any of the import Instances require resource data. • If the fully specified drive and path contains blanks or other special characters, it is necessary to enclose in quotation marks the entire file identifier, including the drive letter and all directories.
OUTPUT	This is the prefix of the output files (the merged “accepts” and “ignores”). Specify a filename to be used for the result of importing this and any modifications to the input media that were required due to events such as duplicate OIDs, different domains, different parentage, etc.
	• This value is required if any of the output of the modified XPI and XPR decks is required. It defines the directory and name, in XPI and XPR, of the generated, modified output. • If `.xpi` is specified, it will be stripped off. While this keyword is optional, there is no default—if it is omitted or specified

	without a valid value, no output files other than logs will be available after the completion of RadDBUtil.
COMMIT	Specifies whether the CSDB will be updated. `-commit true` allows those elements that meet the import criteria to be passed into the CSDB for processing. `-commit false` (the default) forces RadDBUtil to only scan and reconcile the input files and the database.
ROOT	Specifies the root instance of the model (either FILE.DOMAIN.CLASS .INSTANCE, DOMAIN.CLASS.INSTANCE, CLASS.INSTANCE, or CLASS). • Only matching instances and those that are referenced (directly or indirectly) by root will be processed. • The default is *****. • COMPONENT classes and instances cannot be specified.
DOMAIN	Specifies an existing CSDB domain into which this import deck is to be placed. • If the specified value does not correctly identify an existing domain in the target database, RadDBUtil will end with an error. If a domain is specified, the input files will be imported into that CSDB domain. However, if no domain is specified (the default), the input files will be imported into the domains that are specified in the decks. • All of the instance and resource data that are contained in the INPUT-specified decks will be examined and considered for import into the single CSDB domain that is specified by this keyword. • Domain names in the CSDB are limited to 32 bytes. The `-domain` switch is designed to put the contents of the input decks into a single destination. Therefore, when importing a Patch Manager bulletin, do not use this switch because if the instances are coming from multiple domains they must be imported back into multiple domains.
REUSE	Specify as: `-domain` *domain_name*`:REUSE`This is an optional directive that allows RadDBUtil to compare the target database and import deck for matching package instances and, when present, re-use those that are in the database. This allows RadDBUtil to avoid creating duplicates of package instances and resources in the target domain if they already exist, in either the original source or target domain. *For each non-root and non-component element in the deck*: the domain that is specified by the DOMAIN keyword is searched first, followed by the domain that is identified in the package instance of the import deck. If a package instance has the *same name* as the package instance that is found in the CSDB, RadDBUtil will: • ABORT if it is determined that the contents are different from that which is in the import deck. • REUSE it if it is determined that the contents are identical to that which is in the import deck. This means that the existing database package instance will be referenced by the appropriate connection attributes in the import deck, and the matching Package Instances in the import deck will be removed.

	While the REUSE option might remove duplicates from the import process, the output files (`INPUT.XPI` and `INPUT.XPR`) will contain the instances and resources that were in the original import deck, and which could have been used to update the database but were deferred as a result of this option.
ACCEPT, REJECT, and IGNORE	The instances in the import deck are compared to the instances in the database. These optional keywords dictate the action—based on that comparison—that RadDBUtil is to take on the import deck and database instances. There are four instances *types*. They are: • **Adds (A)** are instances that are in the import deck and are to be added to the database. The default behavior is to **ACCEPT** these additions. • **Deletes (D)** are instances that are to be deleted from the database. The default behavior is to **REJECT** these deletions. • **Sames (S)** are instances in the import deck that are identical to those in the database. The default behavior is to **IGNORE** these instances. • **Updates (U)** are instances in the database that will be updated by a matching instance in the import deck. The default behavior is to **REJECT** these instances. These actions (ACCEPT, REJECT, and IGNORE) are applicable only to these four instance types (A, D, S, and U), and act on only those instance types that have been specified for each. The following points are additional items for consideration when using these keywords. • An instance type cannot be specified for multiple actions in a single RadDBUtil execution. That is, **Adds** cannot be configured to be accepted *and* ignored. • Multiple instance types can be specified for one action (as in, `-reject d+u`). If multiple instance types are specified for an action, a plus sign (**+**) must separate them. • Any operations that match the REJECT parameters will cause the tool to not commit, and will return a non-zero return code when processing has completed.

MSI Files

RadDBUtil has an option that allows an import error to be overruled if inconsistent MSI files are discovered. This might occur if the package with the materials matching the IDX file has been renamed (and, as a result, cannot be found by either import or tree export), or where the ACP file might not yet have been imported into the database, but is in a set of imports that is to be processed subsequent to the current import materials.

The command to overrule the import error is IGNORE=BADMSI.

Output Files

This section provides information about the log files that will be automatically generated by the IMPORT verb. It also discusses additional logs and files, and the conditions under which they might be generated.

- **Standard Files**
  - **RADDBUTIL.LOG**

    > **Note:** To save the logs of previous RADDBUTIL.EXE executions, rename the (RADDBUTIL.LOG) file.

  This log queries the `edmprof` file and identifies the location of the edmprof file, the CSDB, and the Configuration Server log. It also contains return codes and summary information about execution results.

  - **RADDBUTIL.AUDIT.LOG**located in the directory specified by DIRECTORY in the MGR_ LOG section of the edmprof file. This file contains a record of all RadDBUtil calls and the corresponding return codes; it is designed for archival reference only.
  - **STDERR**
    contains the same information as RADDBUTIL.LOG but, by default, is directed to the console. This log can be redirected as desired.
- **Conditional Files**
  XPR and **XPI** Files
  If RadDBUtil import updates the database, these two files will be created in the directory that is optionally specified by the keyword OUTPUT. The contents of these files can be returned to the customer's **Digital Source Library** (DSL) as a record of the materials as imported into the target CSDB.

EXPORT

> **Caution:** It is recommended to create a back up the CSDB before executing any of the commands that are shown in this section.

The EXPORT verb allows for the simultaneous exporting of the XPC, XPI and, optionally, the XPR decks that are needed to ensure that the exported portions are accurately reproduced in another database. This includes class templates, instances and, optionally, the resources. It also allows the specifying of the entire database, or individual parts of the database (such as domain, class, instance, package, and service) to be exported. EXPORT offers the ability to:

- Perform deletions based on the results of object resolution.
- Specify multiple inputs, such as exporting four services on one export operation.

Additionally, exporting can include:

- The associated resources, and
- All required packages (similar to a client resolution).

Syntax

The syntax of the EXPORT verb is shown below. For examples of the syntax, see "Examples" on page 191 .

> **Note:** Optional keyword-value combinations are in parentheses. Default values are underlined.

```
RADDBUTIL EXPORT (,SUBSTITUTE=TRUE|FALSE)(,DATA=TRUE|FALSE)
(,WALK=TRUE|FALSE) (,OUTPUT=stemname) INPUT
```

```
Raddbutil export (-substitute 0|1) (-data 0/1) (-walk 0/1)(-
outputstemname) input
```

EXPORT Keywords

The table "RadDBUtil EXPORT Keywords" lists and defines the keywords for the EXPORT verb.

RadDBUtil EXPORT Keywords

Keyword	Explanation
DATA	Indicates whether to export the resource files in the CSDB. The default is **0** (**FALSE**, **NO**).
WALK	Indicates whether to do a resolution—CSDB should be traversed. The default is **1** (**TRUE**, **YES**). *INPUT is not a keyword, like the others; it is a documentation placeholder. A value must be specified without "input" being used as an indicator, as shown in this table.*
OUTPUT	This is the prefix of the output files. • If `.xpi` is specified, it will be stripped off. • This keyword does not have a default value.
SUBSTITUTE	Indicates whether to do a *variablized substitution*. The default is **1** (**TRUE**, **YES**). It is used while resolving the variablized substitutions when exporting. • `-substitute true` (the default): variablized substitution done to *, export all • `-substitute false`: variablized substitution done to _UNKNOWN_, export none For example, when `-substitute true`: `PRIMARY.SOFTWARE.ZSERVICE.A001&(ZCONFIG.ZHDWCOMP)` would resolve to `PRIMARY.SOFTWARE.ZSERVICE.A001*`
INPUT	This, the only mandatory parameter, indicates the CSDB Instances that are to be exported. *INPUT is not a keyword, like the others; it is a documentation placeholder. A value must be specified without "input" being used as an indicator, as shown below*: `Raddbutil export (-data 0/1) (-walk 0/1) (-output` *stemname*`)` *file.domain.class.**
	• The format can be either: ▪ `FILE.*.*.*`, ▪ `FILE.DOMAIN.*.*`, ▪ `FILE.DOMAIN.CLASS.*`, ▪ `FILE.DOMAIN.CLASS.INSTANCE`, or ▪ `FILE.DOMAIN.CLASS.INSTANCE(msg)`.

	• Wildcards (*) are valid values for the domain, class, and instance keywords. • More than one database instance can be specified; multiples must be separated a blank space. The INPUT value must be specified at the end of the command line; otherwise the operation will fail.

Output Files

The EXPORT verb will always generate `XPC` and `XPI` files. If `-data` is specified, an `XPR` file will also be generated.

DELETE

Note: It is recommended that you back up the CSDB before running any of the commands that are shown in this section.

This verb deletes instances and resources from the CSDB. It offers the ability to perform deletions based on:

- The results of object resolution.
- The contents of an `XPI` file.

Note: The `XPI` file that is passed to RadDBUtil when using this verb must be an `XPI` file that was exported from the CSDB from which the instances and/or resources are to be deleted.

Syntax

The syntax of the DELETE verb is shown below. For examples of the syntax, see "Examples" on page 191.

Note: Optional keyword-value combinations are in parentheses.
Default values are in underlined.

```
RADDBUTIL DELETE
(,PREVIEW=TRUE|FALSE)(,FILE=value)(,WALK=TRUE|FALSE)(,IGNORE=value),
INPUT
```

```
Raddbutil delete (-preview 1|0) (-file value) (-walk 1|0) (-ignore
value) input
```

```
RADDBUTIL DELETE
(,PR-
EVIEW=TRUE|FALSE)(,FILE=value)(,WALK=TR-
UE|FALSE)(,LOGFILE=value)(,IGNORE=value),INPUT
```

```
Raddbutil delete (-preview 1|0) (-file value) (-walk 1|0) (-logfile
value) (-ignore value) input
```

DELETE Keywords

The following table " RadDBUtil DELETE Keywords" lists and defines the keywords for the verb, DELETE.

RadDBUtil DELETE Keywords

Keyword	Explanation
PREVIEW	Indicates whether to preview the changes only, or make the deletions. The default is **0** (**FALSE**, **NO**, **OFF**).
FILE	The name of XPI file that specifies what to is to be deleted from the CSDB.
WALK	Indicates whether to do a resolution—CSDB should be traversed. The default is **1** (**TRUE**, **YES**, **ON**).
LOGFILE	The name of the custom log file where logs will be written. If LOGFILE parameter is not added, the logs are written to the default log file.
IGNORE	This keyword specifies (in) which CSDB instances should not be deleted. • The format can be either F.D.C.I or f.d.c.i format. • Wildcards (*) are valid in the domain, class, and instance specifications. • More than one database instance can be specified; multiples must be separated a plus sign (**+**).
INPUT	This parameter indicates which CSDB instances are to be deleted. *INPUT is not a keyword, like the others; it is a documentation placeholder. A value must be specified without "input" being used as an indicator, as shown below.* `Raddbutil delete file.domain.class.*` • The format can be either: ▪ `FILE.*.*.*`, ▪ `FILE.DOMAIN.*.*`, ▪ `FILE.DOMAIN.CLASS.*`, ▪ `FILE.DOMAIN.CLASS.INSTANCE`, or ▪ `FILE.DOMAIN.CLASS.INSTANCE(msg)`. • Wildcards (*) are valid in the domain, class, and instance specifications. More than one database instance can be specified; multiples must be separated a blank space.

Deleting Bulletins from a Database

To permanently delete Patch Manager bulletins from the Configuration Server Database, specify the following classes with the IGNORE keyword.

- PRIMARY.PATCHMGR.CMETHOD
- PRIMARY.PATCHMGR.OPTIONS
- PRIMARY.PATCHMGR.METADATA
- PRIMARY.SYSTEM.ZMETHOD
- PRIMARY.SYSTEM.PROCESS
- PRIMARY.PATCHMGR.PRODUCT
- PRIMARY.PATCHMGR.SP
- PRIMARY.PATCHMGR.RELEASE
- PRIMARY.PATCHMGR.PATCHARG
- PRIMARY.PATCHMGR.PG2PR
- PRIMARY.PATCHMGR.PROGROUP

The following example shows these classes being included with the IGNORE option. Make sure that there is no space between * and + when you run this command.

```
Raddbutil.exe delete -walk 1 -ignore
PRIMARY.SYSTEM.PROCESS.*+PRIMARY.SYSTEM.ZMETHOD.*
+PRIMARY.PATCHMGR.CMETHOD.*+PRIMARY.PATCHMGR.METADATA.*
+PRIMARY.PATCHMGR.OPTIONS.*+PRIMARY.PATCHMGR.PATCHARG.*
+PRIMARY.PATCHMGR.PRODUCT.*+PRIMARY.PATCHMGR.RELEASE.*
+PRIMARY.PATCHMGR.SP.*+PRIMARY.PATCHMGR.PG2PR.*
+PRIMARY.PATCHMGR.PROGROUP.*
 -preview 0 PRIMARY.PATCHMGR.ZSERVICE.MS07-042(SYNC)
```

RCS

This verb communicates with the CSDB and allows for:

- Querying of the CSDB **lock status**.
- Unlocking of the CSDB.

Syntax

The syntax of the RCS verb is shown below. For examples of the syntax, see the section, "Examples" on next page.

```
raddbutil rcs status
```

```
raddbutil rcs unlock
```

RCS Keywords

The following table "RadDBUtil RCS Keywords" lists and defines the keywords for the verb, RCS.

RadDBUtil RCS Keywords

Keyword	Explanation
STATUS	Displays the lock status of the Configuration Server status.
UNLOCK	Unconditionally unlocks the Configuration Server.

Return Codes

The following table " RadDBUtil Return Codes" shows the return codes that are associated with the RadDBUtil executable.

RadDBUtil Return Codes

Return Code	Meaning
0	SUCCESS
4	WARNING
8 or 16	FAILURE No database update occurred, or A database update was started but not completed. Note: If the latter, the database might be in an error state.

Examples

This section presents a few examples of the simpler and more direct **RADDBUTIL.EXE** syntax.

Note: As previously stated, the `RADDBUTIL.EXE` utility supports the following two syntax formats:
`-keyword` *value*
and
`keyword=`*value*
The examples in this section are presented in the `-keyword` *value* format.

IMPORT Examples

Performing a Simple Import

Run a routine, daily import of the file, `sample.xpi` to add instances to the CSDB.

```
raddbutil import -input sample -commit yes
```

Stop if there are updates and/or deletes.

```
raddbutil import -input sample -accept A -reject U+D-commit yes
```

Ignore any updates and deletes.

```
raddbutil import -input sample -accept A -ignore U+D-commit yes
```

Results:

- The instances and resource data from the files `sample.xpi` and `sample.xpr` are imported directly into the domain and class that are specified in the input deck.
- No domain mapping is performed

Performing a Simple import and Replacing the Old Instances

```
raddbutil import -input sample -accept A+U+D -commit yes
```

Importing to a Domain Other Than SOFTWARE

Import all instances from the input deck into the SOFT0002 domain. Re-use any database elements that are identical to elements of the deck, overrule the inconsistent MSI file import error, and reject any updates.

Example 3a

Note: The syntax of the following example is fully supported by RadDBUtil.

```
Raddbutil Import -Input Sample -Domain SOFT0002:Reuse-Commit Yes -
Ignore Badmsi
```

Example 3b

Note: The following example is the same as Example 2a, but includes the fully specified drive and path (with special characters) of the input material, and the name of the output decks.

```
RADDBUTIL IMPORT -INPUT "G:\CONTAINS BLANKS\SAMPLE"-OUTPUT Sample_
Soft_2 -DOMAIN SOFT0002:REUSE -COMMIT YES -IGNORE BADMSI
```

EXPORT Examples

An Easy Method to Create Export Files

The following command will create `AMORTIZE.XPC` and `AMORTIZE.XPI` containing just the specified class and instance.

```
raddbutil export -output amortize PRIMARY.SOFTWARE.ZSERVICE.AMORTIZE
```

Using the -walk Command

The following command will create `AMORTIZE.XPC` and `AMORTIZE.XPI` containing the classes and instances. This command will resolve the package, and include any other required packages because `-walk` is specified.

```
raddbutil export -output amortize -walk 1
PRIMARY.SOFTWARE.ZSERVICE.AMORTIZE
```

Note: No resources will be exported.

Using the -walk and -data Commands

Example 6a

The following command will create `AMORTIZE.XPC`, `AMORTIZE.XPI`, and `AMORTIZE.XPR` which contain the classes, instances, and resources (because `-data` is specified), and the package will be resolved because `-walk` is specified.

```
raddbutil export -output AMORTIZE -walk 1 -data 1
PRIMARY.SOFTWARE.ZSERVICE.AMORTIZE
```

Example 6b

The following command will create `ALL.XPC`, `ALL.XPI`, and `ALL.XPR` containing the classes, instances, and resources of all services in the SOFTWARE Domain.

```
raddbutil export -output ALL -walk 1 -data 1
PRIMARY.SOFTWARE.ZSERVICE.*
```

Exporting and Importing the PRIMARY File (ZEDMAMS Used to Import the Class)

The following series of commands will export and then import the entire PRIMARY file of the CSDB.

```
raddbutil export -output ALL -walk 1 -data 1 PRIMARY.*.*.*
```

```
zedmams verb=import_class,file=ALL.xpc,preview=no,logfile=ALL_
PRIMARY.log
```

```
raddbutil import -input ALL -commit yes
```

Exporting, Deleting, and Importing a Bulletin

The following commands will export, delete, and import (respectively) the Patch Manager bulletin, MS07-042.

Export

```
./raddbutil export -walk 1 -data 1 -output ms07-042
"PRIMARY.PATCHMGR.ZSERVICE.MS07-042(SYNC)"
```

Delete

```
raddbutil.exe delete -walk 1 -ignore
PRIMARY.SYSTEM.PROCESS.*+PRIMARY.PATCHMGR.ZMETHOD.*
+PRIMARY.SYSTEM.CMETHOD.*+PRIMARY.PATCHMGR.METADATA.*
+PRIMARY.PATCHMGR.OPTIONS.*+PRIMARY.PATCHMGR.PATCHARG.*
+PRIMARY.PATCHMGR.PRODUCT.*+PRIMARY.PATCHMGR.RELEASE.*
```

```
+PRIMARY.PATCHMGR.SP.*+PRIMARY.PATCHMGR.PG2PR.*
+PRIMARY.PATCHMGR.PROGROUP.* PRIMARY.PATCHMGR.ZSERVICE.MS07-042(SYNC)
```

Import

raddbutil import -input ms07-042 -commit yes

Note: In the import example, the **-domain** switch was not used because the instances are coming from multiple domains.
See the warning under DOMAIN in "IMPORT Keywords" on page 183.

The DELETE Verb

This series of example focuses on the DELETE verb commands.

- Delete the service PRIMARY.SOFTWARE.ZSERVICE.MSOFFICE.
  ```
  raddbutil delete PRIMARY.SOFTWARE.ZSERVICE.MSOFFICE
  ```
- Delete the service PRIMARY.SOFTWARE.ZSERVICE.MSOFFICE; do not delete the methods.
  ```
  raddbutil delete -walk 1 -ignore PRIMARY.SYSTEM.ZMETHOD.*
  PRIMARY.SOFTWARE.ZSERVICE.MSOFFICE
  ```
- Delete the instance PRIMARY.SOFTWARE.ZSERVICE.MSOFFICE.
  ```
  raddbutil delete -walk 0 PRIMARY.SOFTWARE.ZSERVICE.MSOFFICE
  ```
- Delete the instance PRIMARY.TD1.ZSERVICE.MSOFFICE and write the log to the `user_log.log` file with increased logging level
  ```
  raddbutil delete -preview 0 -walk 1 -loglvl 9 -logfile user_log.log
  PRIMARY.TD1.ZSERVICE.MSOFFICE
  ```
- Delete the contents of `foo.xpi`.
  ```
  raddbutil delete -file foo.xpi
  ```
- Delete the content of the `foo.xpi` and write the logs to the `user_log.log` file
  ```
  raddbutil delete -file foo.xpi -preview 0 -walk 1 -loglvl 9 -logfile
  user_log.log
  ```

The Configuration Server Database

The following commands pertain to the Configuration Server Database.

- Show the current status of the CSDB.
  ```
  raddbutil rcs status
  ```
- Unlock the CSDB.
  ```
  raddbutil rcs unlock
  ```

Chapter 6

Configuration Server Performance

An Overview of Performance Issues

The purpose of this chapter is to discuss system performance issues as they relate to the Configuration Server. The next chapter, explores some problem determination issues.

Performance issues are associated with enhancing the efficiency of a working system, while troubleshooting deals with features, functions, and components that are not operating as expected. Taking into account performance and usage considerations before configuring the Configuration Server might prevent many of the conditions that require troubleshooting.

The Configuration Server is a multi-processing server framework for:

- Policy Management
- Component Management
- Network Management
- Version Management
- Asset Management
- State Management

There are many aspects of Configuration Server performance. In addition, there are specific phases of Configuration Server operations. Each phase has different performance characteristics and requirements. Because of these many variables, there is no easy “cook book” approach to Configuration Server performance.

General Performance and Usage Considerations

Three important performance and usage considerations must be taken into account before beginning any discussion of Configuration Server issues.

- *What is the overall system infrastructure?*
  This includes numbers, types, and speeds of processors; total size and type of memory; and network capability and configuration.
- *What are the performance benchmarks?*
  These include average performance levels, as well as the high and low levels.
- *What are the workload parameters?*
  These include average demand, peak load requirements, and idle times.

Before undertaking any further performance initiatives, become familiar with the above considerations as they apply to your Configuration Server.

How this Chapter is organized

This chapter is divided into three areas that dramatically influence performance:

- "CPU Requirements" below, starting .
- "Memory" on next pagestarting .
- "Bandwidth Throttling" on page 200starting .

CPU Requirements

There are minimum CPU requirements specified at installation for each Configuration Server platform. It must be noted, however, that these are *minimum values*. The real requirements for CPU utilization can only be determined by workload—essentially, the number of resolutions that a Configuration Server can process.

The CPU and Object Resolution

As each RCA agent connects to the Configuration Server, an identifier object (ZMASTER) is sent from the RCA agent to the Configuration Server triggering the dynamic construction of an object model for that RCA agent. This process is known as *object resolution*. The object resolution process exhausts most of the processor time required by the Configuration Server.

When trying to determine how many object resolutions can take place simultaneously, use the simple formula outlined below.

Total Number of Possible User Resolutions = Total Number of Available CPU Seconds/Total Number of CPU Seconds Required per User

Total Number of Available CPU Seconds

This value is obtained by multiplying the number of CPUs by the number of seconds in the connection window of opportunity. The window of opportunity is the timeframe in which the RCA agents need to connect.

For example: a two-processor machine with a six-hour window of opportunity (e.g., 12:00 AM - 6:00 AM) would result in a total number of available CPU seconds of 43,200. (2 processors X 6 hours (21,600 seconds) = 43,200 seconds).

Total Number of CPU Seconds Required Per User

By taking the average number of FILE objects per user (ZRSOURCE being the most common object in a user's model) and multiplying it by three (an estimate of how many objects are resolved to end up with a fully resolved ZRSOURCE), we get the average number of objects to be resolved per user. We then divide that by the number of objects the Configuration Server can resolve in a CPU second, and get the number of CPU seconds required to resolve the average RCA agent's object model.

For example, let us assume that the average number of ZRSOURCE objects per user in your environment is 1000. We multiply that by 3, and get 3000 objects per user. Now divide 3000 by 860 (the average number of objects resolved per second by the Configuration Server), and you get approximately 3.5 seconds of CPU time required to resolve the average user's model.

1000 x 3 = 3000
3000/860 = ~3.5 (3.488…)

Total Number of Possible User Resolutions

This value is obtained by dividing the (*Total Number of Available CPU Seconds*) by the (*Total Number of CPU Seconds Required per User*). Using the results of our previous examples, we would divide the 43,200 (available CPU Seconds based on a two-processor machine with a six-hour window) by 3.5 (number of CPU seconds required to resolve the average user's model) resulting in a (*Total Number of Possible User Resolutions*) of approximately 12,000.

43,200/3.5 = ~12,000 (12,342.8571…)

Note: This calculation does not mean that 12,000 users could be successfully configured in the six-hour period. Other variables must be considered, such as disk I/O for data being transferred/received to/from RCA agents, the platform's network card capability (for concurrent communications between the Configuration Server and RCA agents), and the system's available memory (process/memory swapping requires system overhead).
Also, note that other tasks running on the machine will be sharing system resources with the Configuration Server.

Memory

There are two features that are related to the memory usage, **content caching** and **index caching**. They are established in the MGR_CACHE section of the Configuration Server `edmprof` file.

Content Caching

refers to loading a portion of the CSDB (class templates, base instances, and other instances) into memory to speed up the resolution process. This enhances performance by eliminating disk I/O. In configuring index caching, the size used for each content cache entry needs to reflect the size of the instance in the database before any resolution has been performed. This provides a resolution boost across the Configuration Server.

^{2} **Index Caching**

is used to keep in memory all names of the instances of the class that have been cached. Caching the names of all instances of cached classes eliminates the need to read the directory to determine which instances begin with the specified prefix. Index caching makes a significant performance improvement when the generic resolution feature is utilized. Generic resolutions are those that use a partially specified CLASS.INSTANCE naming format that terminates in an asterisk (*****), indicating that all instances with the same prefix are to be resolved.

MGR_CACHE

The MGR_CACHE section of the Configuration Server `edmprof` file defines the values that determine how much **virtual storage** is reserved for **content cache**. The two values are CACHE_SEGMENTS—which determines the number of separate memory areas—and CACHE_SIZE—

which is allocated at startup and used exclusively for content cache. The product of CACHE_SEGMENTS x CACHE_SIZE is the amount of memory that will not be available for resolution purposes during connection.

In a Windows environment, the maximum virtual storage that will be available for any single process is 2 GB. If the Windows Enterprise Server is used, this is increased to 3 GB, and for Windows 2003 Server x64, up to 4 GB of virtual memory per 32-bit process, while it might be constrained by the size of real memory and the size of the page space available.

The Configuration Server process will attempt to use all of the virtual storage that is available to it and might cause all of the defined page space to be in use, so make sure that the total page file space is at least 4 GB. The value of AVERAGE_OBJECT_SIZE in this section should be set to the size of the largest Instance of the Classes being cached. The default is **2048** bytes.

A parameter called ICACHE_SIZE is available for **index caching**. It is enabled by simply specifying a value for the keyword. The easiest means by which to correctly size this is to take the total of all instances to be cached, multiply by 100, and place the result as the ICACHE_SIZE value size. ICACHE_SIZE is of benefit when general resolution is active, that is, any connection to SOFTWARE.PACKAGE.* that requires the Configuration Server to process all of the potential Instances prefixed by the string. While ICACHE is most important for SOFTWARE.FILE, the caching mechanism (described below in MGR_CLASS) uses the same criteria for selecting which DOMAIN.CLASS Instances to cache.

MGR_CLASS

The MGR_CLASS section controls two separate processes: initial Classes to be cached and the amount of storage to be used for in-storage objects during resolution of each RCA agent (as differentiated from CSDB Classes and Instances where the Class name might be the same as the in-storage object name, as is the case of ZSERVICE). In-storage objects of interest are generally persistent and multi-heap. Controlling the storage and processing of these objects provides for performance improvements.

For index caching and content caching, the contents of MGR_CLASS are processed in the order in which they are presented, so the first DOMAIN.CLASS is processed completely (index cache is loaded and content cache is loaded) before the second, and so on.

For each DOMAIN.CLASS (for example, SOFTWARE.FILE) that is identified in this section, four parameters are specified. The first and second control the caching behavior, and the third and fourth are used exclusively for persistent object virtual storage allocation during resolution.

For a detailed explanation of the MGR_CLASS settings, including performance and usage considerations, see "MGR_CLASS" on page 39. The first of the four parameters (Value1) allows one to specify whether the Class template and _Base_Instance_ are to be cached. A value of Y is recommended because it is generally necessary to load the Class template and _Base_Instance_, and this eliminates a tremendous amount of disk I/O for Classes that are commonly involved in resolution.

Networking

Bandwidth Throttling

Bandwidth throttling refers to reserving a percentage of the available TCP/IP bandwidth for use by other processes on the device. It was designed to help maximize network resources while running the Configuration Server. Bandwidth throttling is configured in the Configuration Server using the SEND_THROTTLE setting of the MGR_TIMEOUT section of the `edmprof` file. It specifies the number of milliseconds that the Configuration Server will wait before sending packets. The default is **0**, meaning no delay. The range of values is 0 to 4 GB.

There are three variables in the RCA agent's ZMASTER object that also have an effect on bandwidth throttling, ZBWMGR, ZBWMAX, and ZBWPCT.

- ZBWMGR=YES means the Configuration Server will be controlling the bandwidth.
- ZBWMAX is the maximum speed (bytes/second) of the "sends."
- ZBWPCT is the percentage of the maximum to use (0–100).

Note: The ZBWMGR, ZBWMAX, and ZBWPCT values will override the SEND_THROTTLE setting.

Appendix A

Configuration Server Methods

This appendix is a reference for Configuration Server methods. For information on configuring and using Configuration Server methods, see "ZTCBG Table of Variables" on page 119

The following table provides an alphabetical list of Configuration Server methods and a description of the method's use.

Configuration Server Methods

Method	Description
"EDMMAILQ" on page 204	Deposits e-mail in the mail queue (outbox) so it can be sent to a remote system user.
"EDMMCACH" on page 205	Refreshes or disables cache.
"EDMMDB" on page 205	Locks and unlocks the database against all components except Distributed Configuration Server.
"EDMMGNUG" on page 206	Retrieves a list of local and global groups to which a specified user belongs.
"EDMMPUSH" on page 207	Puts an inbound object into a notify queue.
"EDMMPUTD" on page 208	Receives multiple data types that are sent by the Inventory Manager and stores the data in files on the Configuration Server.
"EDMMRPRO" on page 209	Adds, updates, or deletes instances in the PRIMARY file based on the variables of an in-storage object.
"EDMMSQLG" on page 211	Imports data from an external SQL database.
"EDMMSQLP" on page 211	Exports data to an external SQL database.
"EDMMULOG" on page 211	Used to write to the user log file.
"EDMSIGN" on page 212	Authenticates users against the database.
"EDMSIGNR" on page 213	Authenticates users against external security systems.
"ZDCLASS" on page 214	Deletes a class from the database.

"ZDELINS" on page 214	Deletes an instance or instances from within a database class.
"ZDELOBJS" on page 215	Deletes an in-storage object.
"ZDELPROF" on page 216	Deletes an object in the PROFILE File.
"ZEXIST" on page 216	Verifies the existence of a given class or instance in the database.
"ZGETPROF" on page 217	Creates an in-storage object from the PROFILE File.
"ZNFYT" on page 218	Executes a TCP/IP notification on a RCA agent.
"ZOBJCMPR" on page 219	Compresses an in-storage object.
"ZOBJCOPY" on page 220	Copies an in-storage object.
"ZOBJDELI" on page 220	Deletes an instance from an in-storage object.
"ZOBJDELV" on page 221	Deletes a variable from all instances of an in-storage object.
"ZOBJSORT" on page 221	Sorts instances, by stems, of in-storage objects.
"ZPROMANY" on page 222	Adds or updates an instance to the database.
"ZPUTHIST" on page 222	Puts an in-storage object into the HISTORY file.
"ZPUTPROF" on page 223	Puts an in-storage object into the PROFILE File.
"ZSIMRESO" on page 223	Resolves specified objects.
"ZTOUCH" on page 224	Updates the date/time stamp of an instance.
"ZVARDEL" on page 225	Deletes all in-storage objects.
"ZVARGBL" on page 225	Migrates values from one in-storage object to another and deletes the source object.
"ZVARLOG" on	Displays the contents of an in-storage object.

page 226	
"ZUPDPROF" on page 226	Updates profile information, only; it does not perform any type of deletion.
"ZXREF" on page 227	Cross-references class and instance usage during the object resolution process.

The following pages describe each Configuration Server method, providing examples of use, a description, its parameters, and the associated possible return codes.

Note: All arguments are expected to be in the format, KEYWORD=VALUE, and delimited by commas.
Quotation marks (“ ”) are required when a value contains commas and/or embedded blanks.

EDMMAILQ

This method deposits e-mail in the mail queue (outbox). For this method to execute correctly, the MGR_SMTP_MAIL section must be added to the Configuration Server `edmprof` file.

EDMMAILQ Parameters

attach	Specifies attachment files. Multiple attachments can be listed by using a semicolon (;) delimiter between each attachment. For example, `c:\config.sys;c:\autoexec.bat`. Attachments are sent using MIME, and can be in binary. This parameter is optional.
from	The sender’s address. This parameter is required.
mesgfile	Specifies the file that contains the message. Used in place of the parameter, message, if the message is greater than 255 bytes. This parameter is optional.
message	Specifies a brief message (limited to 255 bytes). This parameter is required.
subject	Specifies the subject of the e-mail. This parameter is optional.
to	Specifies the e-mail recipients. Multiple users can be listed by using a semicolon (**;**) delimiter between each recipient. This parameter is required.

Note: Parameters are used like keywords and are not case-sensitive.

Example

In the example below, an e-mail with a brief message is sent from **user1@company1.com** to **user2@company2.com**.

```
EDMMAILQ from=user1@company1.com,to=user2@company2.com,Message=“This
is a brief message”
```

Example

In the example below, a text file (`c:\report.txt`) with a subject (`My report`) is sent between the same users.

```
EDMMAILQ
from=user1@-
company1.com,to=user2@company2.com,Mesgfile=c:\report.txt,Subject="My
report"
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

EDMMCACH

This method refreshes or disables caching.

EDMMCACH Parameters

Parameter	Description
option	Caching option values are ENABLE or DISABLE.

Example

```
ADDRESS EDMLINK EDMMCACH 'option=ENABLE' ;
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

EDMMDB

This method locks and unlocks the CSDB to all tasks except the Distributed Configuration Server.

EDMMDB Parameter

option	LOCK locks the CSDB to any incoming tasks except Distributed Configuration Server. UNLOCK makes the CSDB accessible to all incoming tasks.

Example

```
EDMLINK EDMMDB “OPTION=LOCK”
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

EDMMGNUG

This method retrieves a list of local and global groups to which a specified user belongs.

Note: This method is functional in a Windows network environment only.

Usage

This method is used to issue a function call to a specific server to collect information about the Windows group membership of a user. It does not provide authentication of a particular user ID. These calls are issued under security provisions of the user used to start the Configuration Server when it runs as a normal task; or under a system account when the Configuration Server runs as a service. See the Security Requirements below for security limitations as defined by Microsoft.

Method Input Parameters

The only parameter passed to the method is the name of the object containing the request. The following table details the input parameters and defaults for the method.

ZUSERID	Required variable used as user name.
NTSRVNAM	Name of the remote server on which the function is to execute. If this parameter is NULL, the local computer is used. The default is the local server.
ZOBJREQ	Name of the response object that will contain information about local and global (network) groups to which the user specified in ZUSERID belongs. The default is **NTGROUPS**.

Method Return Values

The EDMMGNUG method returns group membership information about a user in the object specified in ZOBJREQ (usually NTGROUPS). The following are the variables delivered by the method.

NTGRP**L**CT	Number of local groups on the server to which the user belongs.

NTGRP**G**CT	Number of global (network) groups on the server to which the user belongs.
NTGRP**S**CT	Total number of groups on the server to which the user belongs. (NTGRPLCT + NTGRPGCT)
NTGRP**L**xx	There are as many of these variables as there are local groups that a user belongs to on the specified server. xx = {1, NTGRPLCT}
NTGRP**G**xx	There are as many of these variables as there are global groups that a user belongs to on the specified server. xx = {1, NTGRPGCT }
MSGGRP**L**E	An error message, returned by the Network Management Functions, for request for the local group list to which the user belongs.
MSGGRP **G**E	An error message, returned by the Network Management Functions, for request for the global group list to which the user belongs.
NTUSER	Name of the user for which the function was executed. (This is the same as ZUSERID)
NTSRVNAM	Name of the remote server on which the function was executed.
ZMRC	Return code (set in in-bound and response objects).

EDMMPUSH

This method receives input requests, gets the required parameters, and then puts the requests to the right queues for processing by a specific Notify Manager. An in-bound object, or even a dynamic object, created because of the object resolution can be used to deliver requests to EDMMPUSH. The return code associated with the in-bound object might initiate further action. (For more information, see "Configuration Server Methods" on page 202.)

EDMMPUSH Parameters

nfydelay	The interval for delay before trying to re-notify an RCA agent. The default is the value specified in the NFYT_TIMEOUT setting of the section of the `edmprof` file.
nfyhndl	The domain name of the NOTIFY File where the results of notifications will be stored. The heap number of the request object will become the instance name.
nfymrtry	The maximum number of retries. The default is the value specified in the NFY_ RETRY setting of the section of the `edmprof` file.
ntfyrtim	HP timestamp defining the time after which the notification should occur.
nfyproc	Controls the processing of the current heap request. If Y, the heap will be processed. If N, the request for the current heap is ignored. The default is **Y**.
nfytype	Defines the type of the notify requested. Valid values are: TCP and EMAIL. Note:

	Only the first three bytes of the type are used for the identification. Therefore, EMAIL and EMA will be treated as the same. There is no default for this variable. If it is not defined, the current heap of the object will be ignored.
nfyuinfo	Allows you to enter user information.

Example

```
EDMLINK EDMMPUSH ZNOTIFY
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

EDMMPUTD

This method is called when EDMSENDF is used to send the ZTRANSF object to the Configuration Server. It closes a security loophole. EDMSENDF will not work with the version 4.4 CM Configuration Server, therefore, it is necessary to modify your CSDB by adding a new instance in ZPROCESS and linking it to a ZMETHOD object that invokes the EDMMPUTD method. This will have to be done for each object that is sent to the Configuration Server using EDMSENDF.

EDMMPUTD enables you to receive multiple data types sent by the Inventory Manager, and enables you to specify where on the Configuration Server to store this data.

- With EDMMPUTD, you can inject a REXX method before EDMMPUTD gets dispatched to alter the location of the data based on appropriate criteria or to do security validation.
- The EDMMPUTD method handles the object and data sent from the RCA agent's EDMSENDF method. Thus, the specifications for the inbound object are the same.
- PROCESS class instances must be configured for each inbound object needing EDMMPUTD to receive the appended inbound data and store it in a file.

The following table lists the attributes that EDMMPUTD expects.

Attributes Associated with EDMMPUTD

ZRSCMFIL	ZRSCMLOC	ZRSCMMEM
ZRSCRASH	ZRSCSTYP	ZOBJCLAS
ZOBJDOMN	ZOBJFILE	ZOBJID
ZOBJNAME	ZPERUID	ZPERGID
ZEDMTYPE		

Note: The ZRSCDATE and ZRSCTIME fields are not referenced, nor are they used to update the date/time of the file received. ZRSCSIZE and ZCMPSIZE are ignored also. However, these attributes might be used in the future.

EDMMRPRO

This method allows the adding, updating, and deleting of heaps in the PRIMARY database based on the contents of the parameter object that is passed. Each heap in the object specifies an instance to be added, updated, or deleted.

EDMMRPRO Parameter

object	The name of the in-storage object. The object can have multiple heaps where each heap in the object represents a CSDB instance to be altered.

Example

```
EDMLINK EDMMRPRO 'ANYOBJECT'
```

Return Codes

0	The method was successful.
>0	An error was detected, the method failed.

The instance indicates which CSDB instance will be altered by specifying five control variables:

ZOBJCLAS	Target class, for example, ZADMIN.
ZOBJDOMN	Target domain, for example, SYSTEM.
ZOBJFILE	Target file, for example, PRIMARY.
ZOBJNAME	Target instance. This can be any valid instance name.
ZOBJFILE	Target file, for example, PRIMARY.

- On a **delete** request, only the control variables are used to identify the instance to be deleted, the remaining variables are ignored.
- On **add** and **update** requests, the variables in each instance contain the values used to update the instance in the CSDB.
- The fields that can be updated are variables, class connections, expressions, and methods.
- There are specification differences for the three field types.
  However, regardless of field type, the target instances' field lengths determine the amount of data moved, and length adjustment is performed, including blank padding and truncation.

Variables

Any variable name found in the parameter object and found in the target instance will be used to update the target instance. Any variable not found in the target instance will be ignored.

Method Fields

Method fields found in the parameter object and found in the target instance will be used to update the target instance. Any method not found in the target instance will be ignored.

Note: The string before the equals sign (**=**) must be eight bytes long.

The methods are indicated by variables in the parameter object that are named MTHD*nnnn*, where *nnnn* is 0001 to 9999. The value of the variable in the object should contain

methodfieldname=xxxxxxxxxx

where *methodfieldname* is used to identify the target method field. For example:

```
_ALWAYS_=SYSTEM.METHOD.SIGNON_METHOD
```

OR

```
EDMSETUP=SYSTEM.METHOD.CHECK_APPL_STATUS.***
```

Connection Fields

Class fields found in the parameter object and found in the target instance will be used to update the target instance. Any variable not found in the target instance will be ignored. The connections are indicated by variables in the parameter object that are named CONN*nnnn*, where *nnnn* is 0001 to 9999. Types of connection fields include CONN*nnnn* (connection), INCL*nnnn* (Includes), ALWA*nnnn* (Always), and REQU*nnnn* (Requires). The value of the variable in the object should contain

connectionfieldname=xxxxxxxxxx

where *connectionfieldname* is used to identify the target class field. For example,

```
_ALWAYS_=SOFTWARE.ZSERVICE.MY_SERVICE.
```

These variable names are the same format as object resolution with a message type = _NONE_. This is designed to enable the output of these resolutions (sometimes referred to as reporting resolutions) to be used unchanged as input to EDMMRPRO.

You can update specific target instances while not overwriting some existing values (for example, EDMSETUP=) via EDMMRPRO in one of two ways:

- Each class named by the CONN*nnnn*, the value of the variable in the object *connectionfieldname=xxxxxxxx*, needs to be specified even if it is not the target of change and will be updated with the specified content. Each CONNnnnn needs to be specified for each variable in the sequence to provide a placeholder for updating the variables. For example, CONN0001 EDMSETUP=COUNTRY.USA_EAST_COAST

CONN0002 EDMSETUP=ZSERVICE.XYZ

CONN0003 EDMSETUP=ZSERVICE.ABC

- Another way of updating specific target instances while not overwriting some existing values is to specify a CSV (comma-separated variable) string for the instance. Empty values specified before a comma indicates that the connection should skip over the existing value and not update it. For example, the format for skipping over the first two variables and updating the third would appear as:
  CONN0001 “EDMSETUP=, ,ZSERVICE.ABC”

Notes on EDMMRPRO Usage

- The file, domain, and class must already exist; EDMMRPRO will not add any of these levels dynamically. In addition, any fields being processed must already be defined in the target class; ZPROMANY will not modify classes.
- Different class instances can be altered during one execution of EDMMRPRO. However, it is not advisable to do so as certain field names might overlap (particularly methods and connections).
- Different databases cannot be altered in one execution of EDMMRPRO.

EDMMSQLG

For a detailed description of this method, including usage, see Appendix B, HP SQL Methods.

EDMMSQLP

For a detailed description of this method, including usage, see Appendix B, HP SQL Methods.

EDMMULOG

This method writes a message returned from the execution of a REXX method to a user log file. For this method to work, the MGR_USERLOG section must be added to the `edmprof` file, and ACTIVATE= must be YES.

EDMMULOG Parameter

msg	The message that will be written to the user log file. This message is returned by a method after its execution.

Example

```
ADDRESS EDMLINK “EDMMULOG” MSG
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

Configuration Server EDMPROF File Sample

```
MGR_USERLOG]
ACTIVATE = YES
COLUMN_WIDTH = 128
DIRECTORY =
FLUSH_SIZE = 128
PIPE_SIZE = 100000
THRESHOLD = 500000
```

EDMSIGN

This method enables the Configuration Server to authenticate RCA agent and RCA administrator sessions against the CSDB. The password that is stored in the ZPWD variable in the specified object on the RCA agent/administrator is compared to that which is stored in the user’s profile in the CSDB.

- If the passwords match, the session continues.
- If the passwords do not match, the message “PASSWORD INVALID” is returned.

Changing Passwords

Passwords can be changed by either of the following methods.

- In the RCA Admin Agent Explorer: specifying a new password in ZNEWPWD and the old password in ZPWD.
- On the RCA Admin CSDB Editor login panel: selecting the **Change Password** option, and specifying the password information.

Note: For information about RCA agent and RCA Administrator password authentication, see the *Radia Client Automation Enterprise Administrator User Guide*.

EDMSIGN Parameter

object	Specifies the name of the object from which the ZPWD variable is extracted. If no object name is specified, the ZMASTER object is used by default.

Example

```
REXX
```

```
EDMSIGN

(Uses the ZMASTER Object)

EDMSIGN & (ZCURRENT>ZCUROBJ)
```

Return Codes

0	The method was successful.
4	New user.
8	An error was detected, the method failed.

EDMSIGNR

This method enables the Configuration Server to authenticate sessions with RCA agent and RCA administrator against an external security system such as Windows security. The password that is stored in the ZPWD variable in the specified object on the RCA agent/administrator is compared to that which is stored in the native security system for that user ID.

- If the passwords match, the session continues.
- If the passwords do not match, the message “PASSWORD INVALID” is returned.

Changing Passwords

Passwords can be changed by either of the following methods.

- In the RCA Admin Agent Explorer: specifying a new password in ZNEWPWD and the old password in ZPWD.
- On the RCA Admin CSDB Editor login panel: selecting the **Change Password** option, and specifying the password information.

Note: For information about RCA agent and RCA Administrator password authentication, see the *Radia Client Automation Enterprise Administrator User Guide*.

EDMSIGNR Parameter

object	Specifies the name of the object from which the ZPWD variable is extracted. If no object name is specified, the ZMASTER object is used by default.

Example

```
/*****************REXX***************************/

EDMSIGNR

(Uses the ZMASTER Object)

EDMSIGNR & (ZCURRENT>ZCUROBJ)
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZDCLASS

This method deletes a class and its associated instances from the database.

Note: ZDCLASS will not delete data-bearing instances.

ZDCLASS Parameters

domain	The 32-byte (maximum) name of the domain that houses the class to be deleted.
class	The eight-byte (maximum) name of the class to be deleted.
file	The file name that contains the class to be deleted. This parameter is optional.

Example

```
/***************************** REXX ****************/
 DOMAIN = 'SOFTWARE';
 CLASS = 'TESTCLAS';
 PARM = SUBSTR(DOMAIN,1,8) || SUBSTR(CLASS,1,8);
 SAY 'PARM STRING IS ' PARM;
 ADDRESS EDMLINK ZDCLASS PARM;
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZDELINS

This method displays or deletes (from the CSDB) an instance, or range of instances, within a class. It permits the use of wildcards (*).

Note: Displayed instances will be written to the Configuration Server log even if all other TRACE settings are OFF.

ZDELINS Parameters

Parameter	Description
file	The file that contains the instances to be displayed or deleted.
domain	The domain that contains the instances to be displayed or deleted.
class	The class that contains the instances to be displayed or deleted.
option	DISPLAY if instances are to be displayed. DELETE if instances are to be deleted.
frominst	The name or starting name of the instance to be deleted or displayed.
toinst	The instance to be deleted or displayed. Blanks in this field indicate that it is a single instance to display or delete, not a range.

Example

```
/************************ REXX ********************/
FILE = 'PRIMARY'
 DOMAIN = 'SOFTWARE'
 CLASS = 'PACKAGE';
 FROMIN = 'TSO_ ;
 TOINS = ' ;
 OPTION = 'DISPLAY';
 PARM = FILE||DOMAIN||CLASS||OPTION||FROMIN||TOINS;
 SAY 'PARM STRING IS 'PARM;
 ADDRESS EDMLINK ZDELINS PARM;
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZDELOBJS

This method deletes an in-storage object.

Note: This method supports the deletion of multiple objects in one call, without spawning additional processes.

ZDELOBJS Parameters

object	The name of the in-storage object to be deleted.

Example

```
ADDRESS EDMLINK ZDELOBJS 'ZTEST';
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZDELPROF

This method deletes a class in the PROFILE File of the CSDB.

ZDELPROF Parameters

domain	The domain that contains the object to be deleted.
class	The class that contains the object to be deleted.

Example

```
EDMLINK ZDELPROF 'TESTP1,TESTPROF'
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZEXIST

This method verifies the existence of classes and instances in the CSDB.

ZEXIST Parameters

file	The file that contains the class or instance to be verified.
domain	The domain that contains the class or instance to be verified.
type	The type of file.
class	The class that contains the instance or class record to be verified.
instance	The instance to be verified.

Example

```
FILE = 'PRIMARY'
DOMAIN = 'POLICY' ;
CLASS = 'USER' ;
INST = 'USER1' ;
PARM = FILE || '.' || DOMAIN || '.' || CLASS || '.' || INST ;

ADDRESS EDMLINK ZEXIST PARM ;

IF RC = 0 THEN

 SAY 'QAREXX ****** OBJECT ' INST ' EXISTS;
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZGETPROF

This method accesses the PROFILE File of the CSDB, gets the *dbobject* object and puts it in storage creating an in-storage object, *inobject*. The domain in the PROFILE database is the USERID, ZUSERID, which is found in the current object or in the ZMASTER object. If ZUSERID is not found, the method will return an error message, “Profile error: user ID not found” in the log.

ZGETPROF Parameters

dbobject	The PROFILE File object name.
inobject	The name of the in-storage object to be created.
domain name	The name of the domain in the PROFILE File (usually the ZUSERID of the ZMASTER object).
instance	The name of the instance. This parameter is *optional*.

Example

```
/*************** REXX ******************************/
 PARM='ZSTATUS,ZSTATUS,'ZMASTER.ZUSERID;
 ADDRESS EDMLINK ZGETPROF PARM;
/* GET OLD PROFILE.?.ZSTATUS */
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZNFYT

This method enables you to initiate a PUSH notification (the execution of a program or programs on an RCA agent from another location). To execute notify successfully, EDMEXECD must be running on the RCA agents on which you are executing a PUSH.

ZNFYT Parameters

"process to run"	The application that you are forcing the RCA agent to execute.
domain	The name of the domain where the notification results are stored. If not specified, this will be automatically generated as a function of date/time.
instance	The instance name containing the results of the single notification. If not specified, this will be automatically generated as an eight-digit number. The default is **00000001**. Note: Only numeric names 00000000 to 99999999 are valid. Specifying anything else will result in 00000000 replacing the erroneous name.
password	The ZNFYPWD for the target terminal's ZMASTER object.
port	The port on which the RCA agent notify daemon is listening. This should be the same as the RCA agent's ZMASTER.ZNTFPORT port number.
target IP address	The IP address of the RCA agent device on which you are executing a PUSH.
user ID	The RCA agent user ID.

Note: If the **domain** and **instance** parameters are omitted, the resulting instance will be written as:
`NOTIFY.mmddyyhhmmss.NOTIFY.00000001`

This does not guarantee the uniqueness of the domain name. In addition, the instance name does not represent anything significant, other than sequence.

Example

```
/*trace i*/

RC = EDMGET(“ZNFYT”,0)

NHEAPS = ZNFYT

DO CURRHEAP = 1 TO NHEAPS BY 1

 RC = EDMGET(“ZNFYT”,CURRHEAP)

 NIPADDR = ZNFYT.IPADDR

NPORT = ZNFYT.PORT

 NUSER = ZNFYT.USER

 NPASSW = ZNFYT.PASSW

 NCMDLINE = strip(ZNFYT.CMDLINE)

 NHANDLE = ZNFYT.HANDLE

/*** CALL ZNFYT TO ISSUE THE NOTIFY ***/

 ADDRESS EDMLINK “ZNFYT” NIPADDR || ',' || NPORT || ',' || NUSER ||
',' || NPASSW || ',”' || NCMDLINE || '“'|| ',' NHANDLE || ',' ||
CURRHEAP;

END
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZOBJCMPR

This method compresses an in-storage object.

ZOBJCMPR Parameter

object	The name of the in-storage object to be compressed.

Example

```
ADDRESS EDMLINK ZOBJCMPR 'ZTEST' ;
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZOBJCOPY

This method copies an in-storage object. The resulting object has the same variables and number of heaps as the original object.

ZOBJCOPY Parameters

fromobject	The name of the existing in-storage object to be copied.
toobject	The name of the in-storage object to be created.

Example

```
ADDRESS EDMLINK ZOBJCOPY 'OBJECT1,OBJECT2' ;
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZOBJDELI

This method deletes a heap of an in-storage object.

ZOBJDELI Parameters

object	The name of the in-storage object from which to delete a heap.
instance#	The heap number to delete.

Example

```
/*********************** REXX ********************/

DPARM = 'TESTOBJ,'||1|| ' ';

ADDRESS EDMLINK ZOBJDELI DPARM;
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZOBJDELV

This method deletes a variable from all heaps of an in-storage object. It verifies the existence of the specified object and finds the specified variable in that in-storage object. The variable value is then removed from each heap of the in-storage object.

ZOBJDELV Parameters

object	The object that contains the variable to be deleted.
variable	The name of the variable to be deleted.

Example

```
/************************ REXX *******************/

ADDRESS EDMLINK ZOBJDELV 'TESTOBJ,VAR00001' ;

SAY 'QAREXX ****** VAR00001 DELETED FROM OBJECT TESTOBJ' ;
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZOBJSORT

This method sorts the heaps of an in-storage object by the values of specified variables and according to the desired collating sequence.

ZOBJSORT Parameters

sort sequence	Ascending (SORT) or descending (SORTD).
object	The name of the object to be sorted.
variable	Up to three variable names can be specified.

Example

```
PARM1 = 'SORT,'

PARM2 = 'ZSERVICE'

PARM3 = ',ZOBJNAME,ZOBJDATE,ZOBJTIME ';

PARM = PARM1 || PARM2 || PARM3;

ADDRESS EDMLINK ZOBJSORT PARM;
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZPROMANY

This method allows the addition and updating of instances in the PRIMARY database, based on the contents of the parameter object that was passed. Each heap in the object specifies an instance to be added or updated.

Note: ZPROMANY requires that spaces be entered after commas to blank out existing services in fields on the CSDB.

ZPROMANY Parameter

object	The name of the in-storage object.

Example

```
EDMLINK ZPROMANY 'ZANYOBJECT'
```

ZPUTHIST

This method puts an in-storage object into the HISTORY file. It takes the *inobject* that is in storage and puts it in the HISTORY file as *dbobject*. The domain used in the HISTORY file is the DATE/TIME stamp found in current object, or in ZMASTER object.

ZPUTHIST Parameters

dbobject	The object name that will be put in the HISTORY file. This parameter is optional.
inobject	The object name of the in-storage object.

Example

```
ADDRESS EDMLINK ZPUTHIST 'ZCOMPARE,ZCOMPARE';

ADDRESS EDMLINK ZPUTHIST 'ZSTATUS';
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZPUTPROF

This method puts an in-storage object (inobject) into the PROFILE File of the CSDB, as dbobject. The domain in the PROFILE File is the ZUSERID found in the inobject, or in the ZMASTER object. If ZUSERID is not found, the dbobject is put in the domain, _UNKNOWN.

Note: The ZPUTPROF method allows you to specify multiple objects simultaneously.

ZPUTPROF Parameters

dbobject	The object name that will be put into the PROFILE database. The default is the object name.
domainid	The value of an optional third operand can be used to specify a domain other than ZUSERID or to eliminate the search for a ZUSERID value.
inobject	The name of the in-storage object.

Example

```
/************** REXX ************************************/

ADDRESS EDMLINK ZPUTPROF 'OBJECTS=ZMASTER,ZCONFIG,ZUSERID'
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZSIMRESO

This method resolves the CSDB instance specified by the parameter string, and the resulting objects are left in storage. Any prerequisite objects needed for a successful resolution must already be built and in storage. For example, to resolve:

USER.&ZUSERID

an object containing the ZUSERID variable might have to be constructed. Otherwise, the resolution might not be completely successful.

ZSIMRESO Parameters

file	The file that contains the instance to resolve.
domain	The domain that contains the instance to resolve.
class	The class that contains the instance to resolve.
instance	The instance to resolve.
message	This specifies the message for conditional resolution paths.

Example

```
ADDRESS EDMLINK ZSIMRESO PRIMARY,POLICY,USER,USER1,EDMSETUP
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZTOUCH

This method updates the date/time stamp of an instance in the CSDB.

ZTOUCH Parameters

class	The name of the class that contains the instance to be updated.
domain	The name of the domain that contains the instance to be updated.
instance	The name of the instance to be updated.

Example

```
DOMAIN = “SOFTWARE”;
CLASS = “ZSERVICE”;
INST = “TEST_OBJECT”;
PARM = SUBSTR(DOMAIN,1,8)||SUBSTR(CLASS,1,8)||SUBSTR(INST,1,32);

ADDRESS EDMLINK ZTOUCH PARM ;
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZVARDEL

This method deletes all in-storage objects. There are no parameters associated with ZVARDEL.

Example

```
ADDRESS EDMLINK “ZVARDEL”
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZVARGBL

This method migrates values from one in-storage object to another, and then deletes the source object.

ZVARGBL Parameters

destination	Object to which the values are being migrated.
source	Object from which the values are being migrated.

Note: Only variables with appropriate flag settings will be migrated.

Example

```
EDMLINK ZVARGBL 'TESTSORT,TESTVGBL'
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZVARLOG

This method writes the contents of an in-storage object to the Configuration Server log.

Note: EDMMOLOG will write to the Configuration Server log even if all other TRACE settings are OFF.

ZVARLOG Parameter

object	The name of the in-storage object to be displayed.

Example

```
ADDRESS EDMLINK ZVARLOG 'ZMASTER';
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZUPDPROF

This method only updates profile information. It does not perform any type of deletion.

- If toobject is not specified, the fromobject (source) name will be used.
- If user ID is not specified, the current user ID will be used.

ZUPDPROF Parameters

fromobject	The source object name.
toobject	The destination object name.
user ID	The user ID.

Example

```
ADDRESS ZUPDPROF OBJECT1,OBJECT
```

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

ZXREF

ZXREF cross references class and instance usage during object resolution, and collects information on the cross-referenced objects. It will generate objects that enable administrators to cross reference users with any, and all, Departments, Workgroups, and Services with which they are affiliated (connected).

To implement the ZXREF method:

1. Add new method instances to the METHOD class. Some are methods to create the objects from the PRIMARY database and others are to write these objects to the PROFILE database.
2. Update the Configuration Server process (ZMASTER) used during the RCA agent connect process by adding new methods to SYSTEM.PROCESS.ZMASTER.
3. Update your class templates, if necessary, to add new method attributes.
4. Update the _BASE_INSTANCE_ to specify the method instances to be executed.

ZXREF Parameter

object	The name of the object containing the cross referenced information.

Return Codes

0	The method was successful.
8	An error was detected, the method failed.

We appreciate your feedback!

If an email client is configured on this system, by default an email window opens when you click here.

If no email client is available, copy the information below to a new message in a web mail client, and then send this message to radiadocfeedback@persistent.co.in.

Product name and version: Radia Client Automation Enterprise Configuration Server, 9.00

Document title: Reference Guide

Feedback: