The Manager Guide

Version 4.3

Second Printing

January 25, 2000

Legal Information

Trademarks

Radia, Enterprise Desktop Manager (EDM), EDM Administrator, EDM Agent, EDM Client, EDM Manager, EDM Stager, and EDM Packager are trademarks of Novadigm, Inc. Other brand names and product names are trademarks or registered trademarks of their respective companies.

No investigation has been made of common-law trademark rights in any word. Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of completely fictitious scenarios, data, or other information, and are intended solely to document the use of Novadigm, Inc. software products.

Radia and the Novadigm Enterprise Desktop Manager (EDM) enable you to manage and distribute software throughout your organization. It does not provide you with the right to make copies of software that you have obtained from third parties. You should make sure that you have obtained the right to make and use the number of copies of each software program which you distribute using Radia or EDM.

Confidentiality Statement

These materials contain the confidential, proprietary information of Novadigm, Inc., and are for the sole use of the party to which they are provided and solely for use with the Novadigm software with which they are provided, and as may be expressly agreed upon between that party and Novadigm, Inc. Any other dissemination, distribution, copying or use of the information disclosed hereunder is strictly prohibited.

Restricted Rights

The software and accompanying documentation are provided with "Restricted Rights." Use, duplication, or disclosure by the US Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFAR §252.227-7013; in subparagraphs (c) (1) and (2) of FAR §52.227-19, Commercial Computer

Second Printing

Software-Restricted Rights; or FAR §52.227.14, Rights in General Data, Alternative III, as applicable. Contractor/Manufacturer is Novadigm, Inc., One International Blvd., Suite 200, Mahwah, NJ, 07495.

Notices

The information contained in this document is subject to change without notice, and does not represent a commitment by NOVADIGM, INC. The software described herein is furnished under licensed agreement and/or nondisclosure agreement. NOVADIGM, INC. software can only be used, copied, or transmitted in accordance with the terms of the license agreement.

Summary of Changes

This second printing of the *Manager Guide* documents Version 4.3 of the Manager. The format and content changes made to this second printing are summarized below.

Format Changes

The format has been changed from the first printing in the following ways:

- The text column is slightly narrower at four inches wide, and a 1-1/4 inch outside column has been added for notes.
- The chapters are now grouped into the following parts:

  Part I. Configuring the Manager
  Chapter 1: Manager Settings

  Part II. Operating the Manager
  Chapter 2: Managing Processing
  Chapter 3: Notifying Clients
  Chapter 4: Logs and Messages

  Part III. Managing the Database
  Chapter 5: Database Utilities
  Chapter 6: EDM Access Method Services

  Part IV. Optimizing the Manager
  Chapter 7: Performance
  Chapter 8: Troubleshooting the Manager

  Part V. Special Purpose Managers
  Chapter 9: Multi-Mode Manager
  Chapter 10: HTTP, SLL, and Multicast Managers

  Appendix A: Getting Help

  Appendix B: Manager Methods

The following table will help you find information is this second printing.

First Printing	Second Printing
Chapter 1: Tuning	Chapter 1: Manager Settings
Chapter 2: Database Maintenance Utilities	Chapter 5: Database Utilities
Chapter 3: Logs and Messages	Chapter 4: Logs and Messages
Chapter 4: Manager Methods	Appendix B: Manager Methods
Chapter 5: Notify	Chapter 3: Notifying Clients
Appendix A: Multi-Mode Manager	Chapter 9: Multi-Mode Manager

Content Changes

This printing of the *EDM Manager Guide* contains the following changes in the following chapters:

Chapter 1: Manager Settings

- Changed the name of MGR_DB_VERIFY to MGR_VERIFY_DEPTH.
- New sections were added to the Manager Settings file:
  - MGR_ERROR_CONTROL
  - MGR_HTTP
  - MGR_MODULE_PRELOAD
  - MGR_MULTICAST
  - MGR_SSL
  - MGR_TRANSLATION
  - MGR_VERIFY_DEPTH

- New settings for the following Manager Settings sections:

Section	New Setting
MGR_NOTIFY	ISSUE_WAKE_ON_LAN
MGR_STARTUP	VERBOSE
MGR_TASK_LIMIT	TASK_HEAP_SIZE
MGR_TIMEOUT	SEND_THROTTLE
MGR_TRACE	POOLMISS

- New performance considerations for the following Manager Settings sections:

Section	Performance Consideration
MGR_ATTACH_LIST	CMD=port number to allocate additional ports for TCP Manager tasks.
MGR_STARTUP	VERBOSE will be disabled if running the Manager as an NT service.
MGR_TIMEOUT	SEND_THROTTLE values can be overridden by bandwidth throttling variables sent up in a client object.
MGR_TRACE	The storage trace (ALL=YES) does not include REXXOFF, TRACELOG, or any VSAM trace settings.

- Updated table for MGR_ATTACH_LIST – Manager tasks.
- New default for MGR_TPINIT.GET_REMOTE_HOST_NAME – 0

Chapter 2: Managing Processing

- Added two new optional attributes for the ZMETHOD class:
  - ZMUSTRUN variable
  - ZMSGONERR variable

- Added descriptions of the following REXX functions:
  - EDMGETV
  - EDMSETV
  - EDMGET
  - EDMSET
  - EDMRESO
- Added descriptions for the following Manager REXX Event Points:
  - ZSTARTUP
  - ZTASKSTA
  - ZTASKEND
  - ZNFYxSTA
  - ZNFYxEND
  - ZSHUTDWN

Chapter 3: Notifying Clients

- Added the following EDMMPUSH input variables:
  - NFYDELAY
  - NFYMRTRY
  - NTFYRTIM
  - NFYUINFO
- Added the following variables to the ZCOMMAND class:
  - ZCMDHNDL
  - ZCMDUINF
  - ZCMDRMAX
  - ZCMDDLAY
  - ZCMDNFYD
  - ZCMDNFYT
- New features added:
  - Scheduling for Notifies
  - Wake-On-LAN

Chapter 4: Logs and Messages

- Updated message format – hh:mm:ss
- New messages added:
  - EDM1977I
  - EDM1987I
- Added note about additional messages for message number EDM0999I.
- Added Julian date format message example.

Chapter 5: Database Utilities

- Removed ZDELORPH.
- Added new parameters for the following utilities:

Utility	New Parameters
EDMMEXPI	PHEX
EDMMIMPC	FROMDOMA=, TODOMA=
EDMMIMPI	FROMDOMA=, TODOMA=, CHGCONS=
EDMMIMPR	FROMDOMA=, TODOMA=

Chapter 6: EDM Access Method Services

- Deleted the following utilities:

Utility	Reason for Deletion
COPY_ZRSOURCE	Its functionality is handled by COPY_CLASS.
LIST_CONNECTS	Its functionality is handled by LIST_CONS_VARS.

- New EDMAMS verb - COPY_FIELD.

- The descriptions for the following utilities were expanded and new parameters were added where indicated:

Utility	New Parameters
REFRESH_DMA (was REFRESH_COUNTS)	(DOMAIN=, CLASS=)
MATCH_RESOURCES	(PREVIEW=YES/NO)
RENAME_INSTANCE	None
SEARCH_INSTANCES	None
SORT_OBJECT_ID	None
UPDATE_INSTANCES	None
VERIFY_CLASS	None
ZRSOURCE_UNMATES	None

Chapter 7: Performance

- This is a new chapter.

Chapter 8: Troubleshooting the Manager

- This is a new chapter.

Chapter 9: Multi-Mode Manager

- This was Appendix A in the first printing.

Chapter 10: HTTP, SLL, and Multicast Managers

- This is a new chapter.

Appendix A: Getting Help

- This is a new appendix.

Appendix B: Manager Methods

- This was Chapter 4.

Editorial Improvements

In addition to the changes listed above, this second printing contains various editorial and style updates to each section and chapter.

Table of Contents

Part I

Configuring the Manager

Part I: Configuring the Manager provides information about the operational characteristics of the Manager found in the Manager Settings file (Chapter 1). The Manager Settings file contains the initialization parameters that were established when you installed your Manager. You can change these parameters to enhance system performance or to add optional settings. New version features can often be configured by adding the appropriate Manager Settings section.

Part I, Configuring the Manager contains one chapter:

- *Chapter 1: Manager Settings*

1

Manager Settings

This chapter shows you how to tune the Manager by working with the various sections of the Manager Settings file.

Understanding the Tuning Process

Note: Many external factors affect the performance of the Manager. These include the number of other applications running on the same machine, the processes running in the same processor, the speed of the processor, the bandwidth of the network, the number of processors and other standard workload tuning issues.

The Manager Settings file contains the operational parameters of the Manager. You can tune many aspects of the Manager by working with settings in various sections of the Manager Settings file.

The performance of the Manager depends on a number of factors, including the number of concurrent clients being processed, the complexity of the configurations for those clients, the volume of the data being processed, and so forth. The configuration of the Manager log, which documents system status for informational purposes and for problem determination, can also alter performance.

This chapter describes your options for working with each section of the Manager Settings file.

You cannot tune these factors by adjusting the Manager Settings interactively. Instead, you must tune these factors outside the Manager.

Viewing and Editing the Manager Settings

You can adjust the parameters that comprise the Manager Settings for Windows NT and OS/2, the Manager Settings can be found in the EDMPROF.DAT file, located in the bin subdirectory of the EDMMGR directory. In Unix, Manager Settings are contained in the file called .edmprof and are located in the home directory of the Unix User ID that installs, starts, stops, and maintains the Manager. The MVS Manager Settings are found in the PARMLIB. Use your preferred editor to adjust individual Manager Settings sections.

Manager Settings at a Glance

The Manager Settings file contains the parameters that determine how your Manager is configured to operate. The various sections of the Manager Settings file establish the settings that control the following aspects of the Manager:

- **Identification**

  The MGR_STARTUP section specifies the Manager by ID, name, type, and communications port.

  The MGR_DIRECTORIES section identifies the directory paths for the databases, REXX methods, and non-REXX methods.

  The MGR_LICENSE section contains your unique Novadigm license number.

- **Specification**

  Manager Settings establish application-wide technical parameters.

  The MGR_CACHE section contains cache processing options.

  The MGR_TPINIT section identifies communications packet sizes.

- **Initialization**

  The MGR_ATTACH_LIST section allows you to define which Manager programs are initiated at startup, and also allows you to define options for their functioning.

  The MGR_CLASS section specifies which classes and instances of the database are cached during the Manager initialization process.

SECTION_DELIMITERS (MVS only) specifies which characters are used to distinguish sections in the MVS Manager Settings (Parmlib).

- **Operations**

  A number of sections in the Manager Settings file contain parameters for system operations.

  - MGR_ACCESS determines Manager access to Administrator and Manager Console functions.
  - MGR_VERIFY_DEPTH specifies the parameters for automatic database verification at initialization.
  - MGR_DIAGNOSTIC specifies the timing, size, and logging options for verifying adequate disk space for database operations.
  - MGR_DMA specifies Distributed Manager parameters.
  - MGR_METHODS identifies options for method processing.
  - MGR_NOTIFY specifies the defaults for the Manager notification of clients.
  - MGR_OBJECT_RESOLUTION specifies parameters used in object resolution.
  - The MGR_POOLS section (MVS, NT, and OS/2 only) allocates available pools (in different sizes) of memory for system tasks.
  - MGR_RETRY values define how long a client is to wait before attempting to reconnect to the Manager.
  - MGR_TASK_LIMIT identifies the upper bound for concurrent Manager tasks, both ongoing and deferred, system related and/or Client Connect related.
  - MGR_TIMEOUT specifies the amount of time a Manager will wait for an inactive client.

- **Monitoring**

  The MGR_LOG and MGR_TRACE sections identify where the Manager log is located, how *elastic* it is, and which individual system traces are being captured in the log.

  You can specify the parameters for using SMTP mail messages to support Manager monitoring in the MGR_SMTP_MAIL section.

  MGR_MESSAGE_CONTROL specifies where log messages are to be sent and if they are to be suppressed.

  MGR_USER_LOG allows you to establish a user logging facility.

Manager Settings

Note: Since the Manager Settings file contains optional sections and may be uniquely configured to your particular requirements, you may not see every setting represented when you view your Manager Settings.

Most of the sections of the Manager Settings file may be uniquely configured, and some are based on operating system and communications requirements. While many of the sections are optional, a number are required for proper Manager functioning.

Manager settings are created during the installation of the Manager. In many cases, you are prompted to enter information that is put directly into the Manager Settings file. Some values are derived from installation options, while other parameters are automatically entered during the installation.

Two types of values are in the Manager Settings file. The *as installed value* represents a manual input during installation or a derived entry for a required setting. The *default value* is the entry established by the Manager if there is a blank value for that setting, whether it is required or manually entered.

Format of Manager Settings File

The Manager Settings file is organized into sections. Each section contains individual keywords or settings. The following is an example of the format of an individual section of the Manager Settings file:

Example

```
[MGR_SECTION]
KEYWORD = VALUE
KEYWORD = VALUE
KEYWORD = VALUE
```

The table below provides a list of each section, and a brief description. This table also tells you whether the section is required or optional.

SECTION_ DELIMITERS specify the prefix and suffix used to delimit Manager Settings section names. If you use SECTION_ DELIMITERS, this must be the very first entry in the Manager Settings file.

Example:

```
SECTION_
DELIMITERS=< >
<MGR_SECTION>
KEYWORD=VALUE
KEYWORD=VALUE
KEYWORD=VALUE
```

The Manager Section List Table

Section Name Description	Required or Optional
MGR_ACCESS Specifies access to the Administrator and Console functions.	Optional
MGR_ATTACH_LIST Specifies the Manager attach list which defines the programs to be attached when the Manager 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_VERIFY_DEPTH Specifies the extent of automatic database verification.	Optional
MGR_DIAGNOSTIC Specifies the parameters for diagnostic Manager tasks (zdiagmgr).	Optional
MGR_DIRECTORIES Specifies the path for the databases, REXX methods, and non-REXX methods.	Recommended but not required
MGR_DMA Specifies the parameters for Manager synchronization.	Optional
MGR_ERROR_CONTROL Specifies how to process errors encountered while the Manager is running.	Optional
MGR_HTTP Specifies the parameters for the HTTP Manager.	Optional

Section Name Description	Required or Optional
MGR_LICENSE Enforces Novadigm licensing.	Required
MGR_LOG Specifies the logging directory and logging options for the Manager logging facility.	Recommended, but not required
MGR_MESSAGE_CONTROL Specifies where messages are to be sent and if they are to be suppressed.	Optional
MGR_METHODS Specifies options for method execution.	Recommended, but not required
MGR_MODULE_PRELOAD Specifies modules to be loaded prior to Manager start-up (MVS only).	Optional
MGR_MULTICAST Specifies the parameters for the Multicast Manager.	Optional
MGR_NOTIFY Specifies the parameters for Notify processing.	Optional
MGR_OBJECT_RESOLUTION Specifies parameters used in object resolution.	Optional
MGR_POOLS Specifies the allocation of pool sizes on start-up.	Optional
MGR_QUEUES Defines options for internal queues used by the Manager.	Optional
MGR_RETRY Specifies when a client should attempt a re-connection with the Manager following rejection.	Optional
MGR_SMTP Specifies the parameters the Manager uses to interface with SMTP.	Optional

Section Name Description	Required or 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 SSL Manager.	Optional
MGR_STARTUP Specifies the Manager ID and TCP/IP port number.	Required
MGR_TASK_LIMIT Specifies the number of concurrent tasks allowed.	Required
MGR_TIMEOUT Specifies how long the Manager will wait for a request from a connected client before disconnecting it.	Optional
MGR_TPINIT Specifies communications packet sizes.	Required
MGR_TRACE Controls and influences diagnostic logging for the Manager.	Optional
MGR_TRANSLATION Specifies customized translation tables.	Optional
MGR_USERLOG Specifies the logging directory and options for the user logging facility.	Optional

This chapter provides a description of each of these sections, as well as the individual settings and the possible values for each setting. In addition, the impact of tunable settings in each section on Manager performance is described.

MGR_ACCESS

The MGR_ACCESS section of the Manager Settings file specifies access for the administrator and operator.

MGR_ACCESS Settings

SETTING NAME Description
ADMIN Specifies access for the administrator. The value is DENY, ALLOW, or IGNORE for this option.
CONSOLE Specifies access for the console. The value is DENY, ALLOW, or IGNORE for this option.

Example

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

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

Performance and Usage Considerations

- A value of ALLOW in the MGR_ACCESS setting will provide access to the administrator without checking of the ZACCESS domain of the database. The result of setting ADMIN=ALLOW is as follows: If the administrator attempts an action for which no access rules have been defined, the attempted action will be allowed.
- Access rules governing administrator actions are defined in the ZACCESS domain of the database when the value specified is DENY. The result of setting ADMIN=DENY is as follows: The administrator will be unable to perform an action unless there is an access rule specifically allowing such an action.
- The result of setting ADMIN=IGNORE is as follows: The administrator will be able to perform any action because the access rules will be ignored by the Manager. Setting the option to IGNORE essentially disables all access rules as they relate to administrator functions.

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

- Access to the console is determined by local password security policy. If CONSOLE=IGNORE, local console security is bypassed. If CONSOLE=ALLOW and local security is not configured, read-only access is granted.

MGR_ATTACH_LIST

The MGR_ATTACH_LIST section of the Manager Settings file specifies which programs (Manager tasks) are to be attached when the Manager is started. This section enables you to specify options for these processes.

MGR_ATTACH_LIST Settings

SETTING NAME Description
ATTACH_LIST_SLOTS Number of slots for the attach list kept in shared memory of the Manager. Every entry is 132 bytes long. We recommend that you specify this parameter (as a minimum) to be the number of cmd_lines in this section plus 1.
CMD_LINE = Command line to use in order to start the requested task. No blanks in sub-string 'CMD_LINE=( )' are allowed. 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 EDM Manager. A third format is allowed when running the LU6.2 interface. This format is 'CMD_LINE=(XXXXXXXX YYYYYYYY,ZZZZZZZZ)' where 'XXXXXXXX' is the name of the module to be executed and 'YYYYYYYY' is the LU name to work with, and 'ZZZZZZZZ' is the TPN (transaction program name) queue. These names should be unique within the EDM Manager.
RESTART = Switch to determine if a Manager task will be restarted by the when abnormally terminated. Values for this setting are YES or NO. No spaces should appear in the "RESTART=YES/NO" string.
RESTART_LIMIT Number of attempts to restart an attached process that has terminated.

Note: No process starts are required, however, system ability is limited if the processes mentioned above are attached. The ztcpmgr, zrexxmgr, and znfytmgr processes are recommended.

SETTING NAME Description
VERIFY_INTERVAL Interval (in minutes) for the task Manager to verify that attached processes are still running. If this value is 0, no verification will occur.

MVS User's Note: When working with the LU6.2 notify function, the second CMD_LINE format represents both the individual instance name as well as the VTAM ACBNAME which will be used to perform the notify. These names are required to be unique. In the third format, the 'YYYYYYYY' represents the VTAM ACBNAME with which the LU6.2 Manager will use to communicate.

The following table lists all the Manager tasks.

Manager Tasks

Process Name	Description
zdiagmgr	Diagnostic Manager task
zhttpmgr	HTTP Manager Task
zlu62mgr	LU6.2 Manager task
zmcstmgr	Multicast Manager Task
zmspxmgr	MIPX Manager task (Microsoft PSX/IPX)
znetbmgr	NetBIOS Manager task
znfy6mgr	LU6.2 Notify Manager task
znfytmgr	TCP Notify Manager task
znfyxmgr	Multiple notify Manager task (znfy1mgr, …)
zrexxmgr	REXX Manager task
zrtrymgr	Notify retry Manager task
zsipxmgr	SIPX Manager task (Novell SPX/IPX)
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
```

Note: The zsipxmgr and znetbmgr are NT and OS/2 Manager specific options. LU options apply to an SNA environment only.

```
CMD_LINE=(zrexxmgr) RESTART=YES
CMD_LINE=(zsnmpmgr) RESTART=YES
CMD_LINE=(zlu62mgr local-lu-name,EDMTRANS)
         RESTART=NO
CMD_LINE=(zsmtrmgr) RESTART=YES
CMD_LINE=(zsmtsmgr) RESTART=YES
CMD_LINE=(zsipxmgr) RESTART=YES
CMD_LINE=(znfytmgr) RESTART=YES
CMD_LINE=(znfy6mgr) RESTART=YES
CMD_LINE=(ztcpmgr)  RESTART=YES
CMD_LINE=(znetbmgr) RESTART=YES
```

Setting Name	Value as Installed	Default Value	Minimum Value	Maximum Value
ATTACH_LIST_SLOTS=	15	15	Number of command lines	CMD_LINE value + 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
VERIFY_INTERVAL=	5	0 = No verification	0 = No verification	3200

Performance and Usage Considerations

- If the ATTACH_LIST_SLOTS value is too low, the Manager will attach as many tasks as there are slots available. Any remaining tasks will not be attached until a slot is freed up. A value that is too high could degrade overall system performance by unnecessarily setting aside resources that remain unused.
- The RESTART_LIMIT value should be set higher when critical Manager functions are being performed. Note that regardless of the value in RESTART_LIMIT, the task will not be reinitiated if RESTART = NO.

- VERIFY_INTERVAL should be set lower when critical Manager functions are being performed to ensure that vital processes are still running. A higher setting, however, might save CPU cycles when total demand is a critical factor.

- The ztcpmgr and zhttpmgr have been changed so that it supports virtual IP addresses. It accepts the IP address and the port number on the command line. `CMD_LINE=(zhttpmgr addr=1.1.1.94,port=4438) RESTART=YES`. If the address is not specified it uses the machine address.

MGR_CACHE

The MGR_CACHE section of the Manager Settings file specifies cache processing options, such as cache segments, size, and statistics.

MGR_CACHE Settings

SETTING NAME Description
AVERAGE_OBJECT_SIZE Average size of an object that will be cached.
CACHE_SEGMENTS Number of cache segments.
CACHE_SIZE Size of each segment.
CACHE_STATS YES/NO switch to accumulate statistics.
ICACHE_SIZE Size of the index cache.

Example

```
[MGR_CACHE]
ICACHE_SIZE =      0
CACHE_SEGMENTS =     2
AVERAGE_OBJECT_SIZE =   2048
CACHE_SIZE =       5242880
CACHE_STATS =      NO
```

Setting Name	Value as Installed	Default Value	Minimum Value	Maximum Value
ICACHE_SIZE	N/A		0 = No index caching	
CACHE_SEGMENTS	2	0 = No caching	0 = No caching	System resource dependent
CACHE_SIZE	5242880 Bytes	0	0	System resource dependent
CACHE_STATS	NO	NO	N/A	N/A
AVERAGE_OBJECT_SIZE	2048	2048 Bytes	2048	6144

Performance and Usage Considerations

- For Unix Managers the recommended value for ICACHE_SIZE is the total number of instances in those classes to be cached (refer to [MGR_CLASS] section) multiplied by 64.
- The ICACHE_SIZE setting should be used in conjunction with the CACHE_SEGMENTS and MGR_CLASS settings.
- The settings in the MGR_CACHE section should be established based on your specific operating environment and performance needs.
- If your AVERAGE_OBJECT_SIZE setting is too low, the Manager will have to reconfigure the cache until the object is accommodated. If the AVERAGE_OBJECT_SIZE setting is too high, you may not be making effective use of the caching function.
- When modifying any of the CACHE parameters in this section, care must be taken to be certain that you do not exceed the amount of virtual memory available. Also, sufficient virtual memory must be available to handle the maximum concurrent resolution workload.

- ICACHE will be enabled only if the value of CACHE_SEGMENTS is greater than 0.
- Classes that are going to be cached are defined in the MGR_CLASS section.
- At a minimum, ZSYSTEM.ZPROCESS & ZPROCESS.ZMETHOD classes should be cached.

MGR_CLASS

The MGR_CLASS section of the Manager Settings file specifies which classes and instances will be cached during the initialization process.

MGR_CLASS Settings

SETTING NAME Description
CLASS Specifies which classes and instances will be cached at initialization. Format: DOMAIN.CLASS={VALUE1,VALUE2,VALUE3,VALUE4} **Note:** If multiple domains have identical class names, the class templates must be identical. If they are not, then performance will be adversely impacted, and objects, larger than necessary, may be created from the resolution process. {Cache Class & Base Instance}, {Cache All Instances}, {Heap Size}, {Maximum Number of heaps}. • **VALUE1** Only **Y** or **N** may be specified. The first parameter specifies whether the class template and base instance of the associated class are to be cached at startup. • **VALUE2** Only **Y** or **N** may be specified. The second parameter specifies whether all instances in the associated class are to be cached at startup. • **VALUE3** This value is all 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 may be used for transient objects (i.e., ZLOCMGR, ZLOCCLNT, ZSCHEDULE). 2048 is the default value, but may be refined for ZSERVICE and ZRSOURCE, which are typically 3 KB-4 KB in size. Examine the client object resulting from a representative resolution to determine the most appropriate value. Values specified are generally in 512-byte increments.

Note: At startup classes are cached in the order they appear if the setting for the class is CLASSNAME=**Y,Y**.

SETTING NAME

Description

- **VALUE4**
  This is a numeric value indicating an estimate of the number of heaps required for resolution. As each client begins resolution, memory Is allocated in blocks equal to the sum of all of the products of **VALUE3*VALUE4.** When memory for a persistent class, which is being cached, is exhausted, the next increment of storage is obtained in a block determined by the product of **VALUE3*VALUE4** for the associated class.

For example, if **SYSTEMX.ZRSOURCE = Y,Y,2048,100** were specified, then:

- The class template and BASE INSTANCE are cached at start-up.
- All of the instances of the SYSTEMX.ZRSOURCE class are cached in memory (if there is enough room in CACHE_SEGMENTS*CACHE_SIZE for the whole class).
- The working value for the size of each ZRSOURCE instance *after resolution* is estimated to be 2048 bytes.

An initial allocation for the in-storage ZRSOURCE object is made for 100 heaps which will require 100*2048 bytes, or 20 KB. IF the resolution mode for the *average* client required 1000 ZRSOURCE instances, then one initial allocation of 20 KB and nine subsequent allocations of 20 KB would be required to satisfy the entire 1000 ZRSOURCE instance in-storage object.

Example

```
<MGR_CLASS>
ZSYSTEM.ZPROCESS  = Y,N,1024,1
ZSYSTEM.ZMETHOD   = Y,N,1024,1
SYSTEMX.ZLOCMGR   = Y,Y,0450,1
SYSTEMX.ZLOCCLNT  = Y,Y,230,1
SYSTEMX.ZRSOURCE  = Y,N,2048,511
SYSTEMX.ZSERVICE  = Y,Y,2048,60
```

Setting Name	Value as Installed	Default Value
CLASS	See example.	See example.

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 impact on storage and performance. A class instance size larger that the actual class size represents wasted storage. A class instance size smaller that 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.
- Classes are cached at startup in the order they appear in the MGR_CLASS section if the setting is Y,Y.

MGR_VERIFY_DEPTH

The MGR_VERIFY_DEPTH section of the Manager Settings file establishes the level of database verification and whether or not errors are automatically corrected. If this function is configured, the Manager will scan the database for Year 2000 compliance, expanded format (ZBASE), and appropriate database ownership (Manager IDs) at start-up. If errors are found, the applicable database utility can be run against the database to correct the error. See *Chapter 5: Database Utilities* for more information

The MGR_VERIFY_DEPTH section enables you to specify four values for these options, CLASS, DOMAIN, INSTANCE or RESOURCE.

MGR_VERIFY_DEPTH Settings

SETTING NAME Description
DB_AUTOFIX= Switch to determine if the database should be automatically fixed (where possible) if errors are discovered. Values for this setting are YES or NO.
VERIFY_DEPTH= Specifies the level of database verification. The values are CLASS, INSTANCE or RESOURCE for this option. • 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 database.

Example

```
[MGR_VERIFY_DEPTH]
MGR_VERIFY_DEPTH=CLASS
DB_AUTOFIX=YES
```

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

Note: Refer to the **MGR_ERROR_CONTROL** section, which offers handling parameters for errors found during the database verification process.

Performance and Usage Considerations

- A value of MGR_VERIFY_DEPTH = CLASS will not result in the validation of the entire database. Specify MGR_VERIFY_DEPTH = RESOURCE to include the entire database.
- A value of MGR_VERIFY_DEPTH = RESOURCE will result in longer start-up times, as the level of detail is deeper. Conversely, MGR_VERIFY_DEPTH = CLASS would result in a much quicker start-up time, because the level of verification is decreased.

MGR_DIAGNOSTIC

The MGR_DIAGNOSTIC section of the Manager Settings file establishes the values the Manager will use to verify that sufficient disk space is available for database operations and logging, as well as the frequency at which the verification occurs.

MGR_DIAGNOSTIC Settings

SETTING NAME Description
DIAGNOSTIC_INTERVAL= Interval (in seconds) for the diagnostic Manager to verify that sufficient disk space is available for the database and log.
DIAGNOSTIC_MIN_DB_BYTES = The minimum established value for database operations.
DIAGNOSTIC_MIN_LOG_BYTES = The minimum established value for logging operations.

Example

```
[MGR_DIAGNOSTIC]
DIAGNOSTIC_INTERVAL= 900
DIAGNOSTIC_MIN_DB_BYTES = 50
DIAGNOSTIC_MIN_LOG_BYTES = 25
```

Setting Name	Value as Installed	Default Value	Minimum Value	Maximum Value
DIAGNOSTIC_INTERVAL=	N/A	900		
DIAGNOSTIC_MIN_DB_BYTES =	N/A	50M		
DIAGNOSTIC_MIN_LOG_BYTES =	N/A	25M		

Performance and Usage Considerations

- You must include zdiagmgr in the MGR_ATTACH_LIST to configure and use this setting.

MGR_DIRECTORIES

The MGR_DIRECTORIES section of the Manager Settings file specifies the path for the databases, REXX methods, and non-REXX methods. See MGR_LOG to specify the directory path for the Manager log.

MGR_DIRECTORIES Settings

SETTING NAME Description
DBPATH Fully qualified directory path for object databases.
METHOD_PATH Fully qualified directory path for non-REXX methods.
REXX_PATH Fully qualified directory path for REXX methods.

Examples

INTEL EXAMPLE:

```
[MGR_DIRECTORIES]
DBPATH=D:\MGR\DB
REXX_PATH=D:\MGR\REXX
METHOD_PATH=D:\MGR\BIN
```

UNIX EXAMPLE:

```
[MGR_DIRECTORIES]
DBPATH=/edm/db
REXX_PATH=/edmmgr/rexx
METHOD_PATH=/edmmgr/bin
```

MVS EXAMPLE:

```
[MGR_DIRECTORIES]
DBPATH=//DDN:
REXX_PATH=//DDN:SYSEXEC
```

```
METHOD_PATH=//DDN:STEPLIB
```

Setting Name	Value as Installed	Default Value
DBPATH	As specified during installation	Current directory
REXX_PATH	As specified during installation	Current directory
METHOD_PATH	As specified during installation	Current directory

While the EDMMGR is implicit in the table above, the implication concerns its functioning with regard to the installation directory.

Performance and Usage Considerations

- The REXX directory specified in the MGR_DIRECTORIES section is further defined by the samples subdirectory. This subdirectory contains a selected set of sample REXX methods.

MGR_DMA

Note: You must have the distributed Manager installed to enable these parameters effectively.

The MGR_DMA section of the Manager Settings file specifies the time-out parameter and the directory path for the distributed Manager. This section enables you to specify values for these options.

MGR_DMA Settings

SETTING NAME Description
ADMIN_LIST = Required only if the security method is specified. Defines the list of administrator User Ids that are allowed to do DMA login. The format is comma-separated, no spaces, case-sensitive. Form EDMSIGNR, the User IDs must be defined in the Manager's native security system.
DMA_TIMEOUT = Interval (in seconds) that the distributed Manager waits for tasks to complete before allowing a DMA commit.
DMA_STAGE_PATH Path in which staging directories will be created.
SECURITY_METHOD Name of security method used to verify DMA logins. (optional) If omitted, no DMA login is required. If DMA security is desired, recommended value is EDMSIGNR.

Examples

INTEL EXAMPLE:

```
[MGR_DMA]
DMA_TIMEOUT=
DMA_STAGE_PATH=D:\MGR\
```

UNIX EXAMPLE:

```
[MGR_DMA]
DMA_TIMEOUT=
DMA_STAGE_PATH=/edmmgr/
```

Setting Name	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

- You must specify the DMA_TIMEOUT setting if you use the MGR_DMA section. The DMA_TIMEOUT setting without a value may cause unpredictable results.
- A low value in the DMA_TIMEOUT setting may not allow critical tasks to finish before synchronization is complete.

MGR_ERROR_CONTROL

The MGR_ERROR_CONTROL section of the Manager Settings file specifies error handling parameters for the Manager. The format of MGR_ERROR_CONTROL is

(ERROR TYPE) = (RESPONSE).

MGR_ERROR_CONTROL Settings

SETTING NAME Description
DB_ERROR = 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 = administrator@yourcompany.com

Example

```
[MGR_ERROR_CONTROL]
DBError = IGNORE,EMAIL
UserEmailErrorsTo = administrator@yourcompany.com
```

Setting Name	Value as Installed	Default Value
DB_ERROR =	N/A	[SHUTDOWN] [NOEMAIL] [NOSNMP]
UserEmailErrorsTo =	N/A	N/A

Performance and Usage Considerations

Note: The actions specified in MGR_ERROR _CONTROL are in response to errors discovered during MGR_VERIFY _DEPTH processing.

- Currently, the only DB_ERROR 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_HTTP

The MGR_HTTP section of the Manager Settings file specifies the operational settings for the HTTP Manager. See *Chapter 10: HTTP, SSL and Multicast Managers* for more information on configuring and using the HTTP Manager.

MGR_HTTP Settings

Setting Name Description
HTTP_PORT = Specifies the port number that the zhttpmgr should listen for connections. By default the PORT number is 80.
RESPONSE_HTML_FILE = Specifies the name of the HTML file that would be sent to the client Web browsers if they accidentally connect to the zhhtpmgr.

Example

```
[MGR_HTTP]
HTTP_PORT = 3466
RESPONSE_HTML_FILE = /test/reject.html
```

Setting Name	Value as Installed	Default Value
HTTP_PORT =	N/A	80
RESPONSE_HTML_FILE =	N/A	N/A

Performance and Usage Considerations

- You must have CMDLINE=(zhttpmgr) RESTART=YES in the MGR_ATTACH_LIST for the HTTP Manager to function.
- If a response file is not specified in the RESPONSE_HTML_FILE setting, the zhttpmgr would send back a status code of 400 (HTTP Bad Request).

MGR_LICENSE

The MGR_LICENSE section of the Manager Settings file contains encrypted character-string data supplied by Novadigm for the software licensing of EDM and Radia.

An additional feature has been added to the product that enables the user to more easily become alerted to license related warnings or apparent problems that may occur as evidenced by a corresponding e-mail message.

MGR_LICENSE Settings

SETTING NAME Description
EMAIL_WARNINGS_TO E-mail address of the system administrator to be notified in case a problem occurs during the license verification process. The message sent out will be exactly the same message as the one written to the Manager log.
LICENSE_STRING Specifies your specific software license. **Format:** 64-character license string consisting of four groups of sixteen characters with one space between the groups.
LICENSE_STRING2 Specifies your specific software license. **Format**: 64-character license string consisting of four groups of sixteen characters with one space between the groups.

Example

```
[MGR_LICENSE SECTION]
LICENSE_STRING = 0123456789ABCDEF 0123456789ABCDEF
0123456789ABCDEF 0123456789ABCDEF
LICENSE_STRING2 = 0123456789ABCDEF 0123456789ABCDEF
E0123456789ABCDEF 0123456789ABCDEF
EMAIL_WARNINGS_TO = administrator@yourcompany.com
```

Setting Name	Value as Installed	Default Value
EMAIL_WARNINGS_TO	N/A	N/A
LICENSE_STRING	Your unique license number	N/A
LICENSE_STRING2	Your unique license number	N/A

Warning: Do not change the MGR_LICENSE section unless a member of the Novadigm Customer Support group instructs you to do so.

Performance and Usage Considerations

- This is essentially the identification number that applies to your configuration as it currently exists. If the configuration is changed, the licensing string will change.

MGR_LOG

The MGR_LOG section of the Manager Settings file specifies the logging directory and logging options for the Manager logging facility.

MGR_LOG Settings

SETTING NAME Description
DIRECTORY Fully qualified directory path where the Manager log is written.
DISABLE_NT_EVENT _LOGGING = YES or NO • If YES is code, messages sent to the Manager log may be sent to the NT event log if they are considered critical. • If NO is coded, such messages are not echoed to the NT event log. This parameter affects only the event logging of Manager log messages. Some NT event log records are written without any corresponding Manager log messages. **Note:** This setting is Windows NT specific.
DISABLE_SNMP_TRAP_LOGGING = YES or NO • If YES is code, messages sent to the Manager log may be sent to the primary SNMP manager as traps if they are considered critical. • If NO is coded, such messages are not sent to the SNMP manager as traps. **Note**: This parameter affects only the trapping of Manager log messages. Some SNMP traps are issued without any corresponding Manager log messages.
FLUSH_SIZE Buffer size (in bytes) that the Manager logger uses to accumulate log messages before writing to the Manager log file.
MESSAGE_DATE Allows insertion of date into every line in the Log. Values are "Julian" and "None".

SETTING NAME Description
MESSAGE_DELIMITER The left and right delimiter used for printing out the Manager Settings in the Manager log. The format is MESSAGE_DELIMITER=xy, where x is the left delimiter and y is the right delimiter. (MVS only) (See also SECTION_DELIMITER.)
MESSAGE_PREFIX Prefix for log messages identifying the application in use: Values are RAD (RADIA) or EDM (EDM).
MESSAGE_WIDTH The maximum width (in characters) of the messages in the Manager log.
PIPE_SIZE Maximum memory size (in bytes) of log messages processed before logged.
THRESHOLD Threshold number of log 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 log file when the limit is reached.

Examples

INTEL EXAMPLE:

```
[MGR_LOG]
DIRECTORY=D:\MGR\LOG
MESSAGE_PREFIX= EDM
FLUSH_SIZE = 1000
THRESHHOLD = -5000000
MESSAGE_WIDTH = 256
PIPE_SIZE  = 1000000
MESSAGE_DATE = JULIAN
```

UNIX EXAMPLE:

```
[MGR_LOG]
DIRECTORY=/edmmgr/log
MESSAGE_PREFIX= EDM
FLUSH_SIZE = 1000
THRESHHOLD = -5000000
MESSAGE_WIDTH = 256
PIPE_SIZE  = 1000000
MESSAGE_DATE = JULIAN
```

MVS EXAMPLE:

```
[MGR_LOG]
DIRECTORY=//DDN:EDMLOGA
FLUSH_SIZE = 1000
THRESHOLD = -5000000
MESSAGE_WIDTH = 500
PIPE_SIZE = 1000000
MESSAGE_DELIMITERS = <>
MESSAGE_DATE = JULIAN
```

Setting Name	Value as Installed	Default Value	Minimum Value
DIRECTORY	edmmgr\log	Current directory	N/A
FLUSH_SIZE	1000 Bytes	5000 Bytes	1
MESSAGE_DATE	N/A	None	N/A
MESSAGE_DELIMITER	N/A	N/A	N/A
MESSAGE_PREFIX	N/A	EDM	N/A
MESSAGE_WIDTH	256	90	80
PIPE_SIZE	1000000 Bytes	6553516 KB	1
THRESHOLD	-5000000 Bytes	100000 Bytes	1

Note: The Manager Settings file accepts both of the following spellings of threshold: "threshold" and "threshhold".

Performance and Usage Considerations

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

- If you are closely monitoring system status using the Manager log, set the THRESHOLD value to a positive value to create and save successive portions of the Manager log. If disk storage is critical, set value to a negative value to reuse the allocated log disk space.
- If you are invoking a number of Manager methods, you can control the size of the Manager log for each method using the MSG_LIMIT setting of the MGR_METHOD section. Additionally, the TASK_LOG_LIM 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, care must be taken to be certain you do not exceed the maximum amount of memory or storage space available.

MGR_MESSAGE_CONTROL

The MGR_MESSAGE_CONTROL section of the Manager Settings file specifies where log messages are to be sent and if they are to be suppressed. MGR_MESSAGE_CONTROL allows you to specify an individual message, a series of messages, or a range of messages and a set of message destinations. Currently there are four message destinations: LOG, WTO (MVS only), EVENTLOG, (Windows NT only) and SNMPTRAP.

MGR_MESSAGE_CONTROL Settings

SETTING NAME Description
XXXX = (Destination) (Message Number(s)) = (Message Destination).

The format of lines in the MGR_MESSAGE_CONTROL section is:

(Message Number(s)) = (Message Destination).

The left side of the equals (=) sign specifies one or more message numbers that are to be affected by the command. There is also a special ALL directive which can be used on the left side of the equals (=) sign to affect all message numbers (0001-9999). If there is no message destination after the equal (=) sign, then the message has no associated destination (it is suppressed).

Example

```
[MGR_MESSAGE_CONTROL]
500 =
520-600=WTO, LOG
400, 220-299, 803, 542-545 = LOG,EVENTLOG,SNMPTRAP
225, 300   =
```

- The first line in the above section would suppress the production of message number 500 (to all message destinations).
- The second line causes the messages numbered 520 to 600 inclusive to be written to the MVS console (via WTO) and to the Manager log.
- The third line causes messages 400, 220, to 299 inclusive, 803, and 542, to 545 to be written to the Manager log, to the NT event log and to the current SNMP Manager as traps. The message is not written to the MVS console because WTO was not specified on the right hand side of the equals (=) sign.
- The fourth and last line causes messages equal (=) 225 and 300 to be completely suppressed.

Setting Name	Value as Installed	Default Value
XXXX = (Destination)	N/A	ALL = LOG

Performance and Usage Considerations

- The MGR_MESSAGE_CONTROL section should only use SNMPTRAP as a destination 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. Also lines that begin with an asterisk (*), double forward slashes (//), or a single forward 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 causes the whole line to be ignored and an error message to be written to STDERR (the SYSTERM DD for an MVS Manager).

MVS User's Note: There is a modify console command which can be used to refresh the message control specifications as well as the trace and pool specifications. That is, the command:

```
F  jobname,R,memname
```

will process any MGR_MESSAGE_CONTROL sections found in member *memname* of parmlib,

MGR_METHODS

The MGR_METHODS section of the Manager Settings file specifies options for method execution.

MGR_METHODS Settings

SETTING NAME Description
LOG_LIMIT Maximum number of messages a method can issue to the Manager 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 Manager method time-out parameter (in seconds).

Example

```
[MGR_METHODS]
LOG_LIMIT = 0
TIMEOUT   = 300
```

Setting Name	Value as Installed	Default Value	Minimum Value	Maximum Value
LOG_LIMIT	0	0 = No Limit		
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 the MGR_TASK_LIMIT section.

- You can tune system resource usage with the TIMEOUT setting. If system resources are critical, a lower TIMEOUT setting will free up unused processing cycles.
- When the TIMEOUT value is reached, the Method is terminated and messages (EDM1975 and 1976) are written to the Manager log.
- If the TIMEOUT value is 0, the Method will continue in effect until its conclusion.

MGR_MODULE_PRELOAD (MVS only)

The MGR_MODULE_PRELOAD section of the Manager Settings file specifies certain load modules that are to be pre-loaded by the Manager during startup to ensure that these modules are loaded only once and to thereby secure a performance improvement. This section should be considered critical for MVS.

MGR_MODULE_PRELOAD Settings

SETTING NAME Description
IEFBR14= YES A sample module name which is to be pre-loaded.

Example

```
[MGR_MODULE_PRELOAD]
IEFBR14 = YES
```

Setting Name	Value as Installed	Default Value
IEFBR14=	N/A	N/A

MGR_MULTICAST

The MGR_MULTICAST section of the Manager Settings file specifies the operational settings for the Multicast Manager. See *Chapter 10: HTTP, SSL and Multicast Managers* for more information on configuring and using the Multicast Manager.

MGR_MULTICAST Settings

SETTING NAME Description
NUMBER_OF_MULTICAST_TASKS Number of concurrent Multicast tasks. Each task caters to a different Multicast host group.
MULTICAST_WORK_DIR The working directory for Multicast.

Example

```
[MGR_MULTICAST]
NUMBER_OF_MULTICAST_TASKS = 5
MULTICAST_WORK_DIR = c:\temp\multicast\
```

Setting Name	Value as Installed	Default Value
NUMBER_OF_MULTICAST_TASKS	N/A	N/A
MULTICAST_WORK_DIR	N/A	N/A

Performance and Usage Considerations

- You must specify zmcstmgr as one of the tasks in the MGR_ATTACH_LIST for the Multicast Manager to function.

MGR_NOTIFY

The MGR_NOTIFY section of the Manager Settings file specifies the defaults for Manager notification of clients.

MGR_NOTIFY Settings

SETTING NAME Description
ISSUE_WAKE_ON_LAN Enables support for TCP Wake On LAN. (supported only on NT and Unix)
NFY_RETRY Number of times to attempt re-notification.
NFY6_LOGMODE LU6.2 default LOGMODE entry name for Notify.
NFY6_TIMEOUT LU62 Notify time-out (in minutes). **Note:** Supported only on the MVS Manager.
NFY6_TPN LU62 default TPNAME for Notify. **Note:** Supported only on the MVS Manager.
NFYT_TIMEOUT Notify time-out (in seconds) for TCP/IP clients.
RECOVERY_DOMAIN The domain that the Retry Manager accesses to reinitiate incomplete Notifies after the Manager has shut down prematurely.

Example

```
[MGR_NOTIFY]
NFYT_TIMEOUT    = 120
NFY6_TIMEOUT    = 62
NFY_RETRY   = 5
NFY6_LOGMODE    = NVDAPCPS
NFY6_TPN    = EDMINIT
ISSUE_WAKE_ON_LAN = YES
RECOVERY_DOMAIN   = ALL
```

Setting Name	Value as Installed	Default Value	Value Range
ISSUE_WAKE_ON_LAN	N/A	NO	YES/NO
NFY_RETRY	N/A	0	32000
NFY6_LOGMODE	N/A	NVDAPCPS	N/A
NFY6_TIMEOUT		0	32000
NFY6_TPN	N/A	EDMINIT	N/A
NFYT_TIMEOUT	120 seconds	0	32000
RECOVERY_DOMAIN	N/A	ALL	RETRY/ALL

Performance and Usage Considerations

- You should establish your MGR_NOTIFY settings based on your network operations parameters.
- The MGR_NOTIFY settings should also be coordinated with values in the MGR_RETRY section, as well as the MGR_TASK_LIMIT section.
- You must have zrtrymgr in your MGR_ATTACH_LIST for the WAKE-ON-LAN Notify function to operate and to allow retry of failed operations.

MGR_OBJECT_RESOLUTION

The MGR_OBJECT_RESOLUTION section of the Manager Settings file specifies the parameters to use during object resolution.

MGR_OBJECT_RESOLUTION Settings

SETTING NAME Description
ALLOW_CIRCULAR_REFERENCES Allow or disallow circular references between classes in the same resolution path. Recognized values are YES and NO.
ALLOW_DUPLICATE_INSTANCES Allow or disallow duplicate instances to be used during object resolution. Recognized values are YES and NO.
ALWAYS_CALL_ZADMIN Force object resolution to call to ZADMIN method to process 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
```

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

Performance and Usage Considerations

- The ALLOW_CIRCULAR_REFERENCES setting should not be set as YES unless specific conditions warrant it and provisions have been made for its operation.
- An ALLOW_DUPLICATE_INSTANCES setting of NO will cause the Manager to eliminate duplicate services at resolution time. This may result in elimination of duplicate ZRSOURCE instances on the client. As the size of the ZRSOURCE object grows, the resolution time will increase by setting this value to NO.
- Novadigm does not recommend the modification or removal of ALWAYS_CALL_ZADMIN = YES for this may prohibit Administrator type functions.

MGR_POOLS (MVS, NT and OS/2 only)

The MGR_POOLS section of the Manager Settings file allows you to specify allocations (pools) of memory in different sizes, prior to Manager start-up. These allocations can be changed dynamically while the Manager is running by using modify commands. In addition, you can establish tolerances for the percentage of waste (the amount of memory in a pool 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 your unique environment, workload, and operating requirements.

When memory is required by the Manager (with a few exceptions), 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 (e.g., MVS getmain) 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 elements).

The specific percentage of waste tolerance parameter is optional. For example, the entry 12000 = 20,75 establishes twenty 12000-byte elements, with a 75 percent waste tolerance.

MGR_POOLS Settings

Setting Name Description
WASTE_TOLERATED The general percentage of waste that will be tolerated to fit a specific requirement to an available pool. Default is 100%.
XXXXXX = The allocation for the XXXXXX byte pool. Any reasonable number of these can be specified.

By default, the MGR_POOL section establishes the pool sizes shown in the example below. In the example, the pools of 64, 400, 630, 2048, and 4120 do not have a specific waste-tolerance specified. This is due to the fact that these pool sizes are so small that the waste in them will be too small to worry about. By not specifying their WASTE_TOLERATED, it has defaulted to 100 percent.

Example

```
<MGR_POOLS>
WASTE_TOLERATED = 100
  168=150
  296=50
  552=20
 1064=10
 2100=5000
 2700=100
 4136=4
 8232=2
12500=2
```

Setting Name	Value as Installed	Default Value	Minimum Value	Maximum Value
WASTE_TOLERATED	100	100 %		100 %
XXXXXX =		(See above)	0	32767

Performance and Usage Considerations

- **MVS Only**: The sum of all of the pools requested must not exceed the MVS Regions size specified for the JOB. Also, additional local address space must be left available for MVS operations that require local storage in the address space to complete. These operations include file opens and closes, such as EDM would perform if the Manager was configured to automatically switch between EDMLOGA and EDMLOGB under certain conditions, as well as REXX's which may dynamically allocate and open MVS datasets.
- Provisions should be made for pools that are the product of the MGR_CLASSES 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). For each of the following classes, special attention should be given to the pools that will be identified by the product of the third and fourth values specified in MGR_CLASSES.

  For example: if the MGR_CLASSES entry for SYSTEMX.ZRSOURCE is 'Y,N,3072,500'.

- The product of the third and fourth values of this is a value exactly equal to 1.5 Megabytes, and one of these will be allocated at the initiation of each resolution. Some 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 TASK_LIMIT values specified (i.e., one element for each concurrent task), and the number of ZRSOURCE instances resolved for each client.

MVS User's Note: You can refresh options set from the MVS parmlib with the following command:

```
F  jobname,REFRESH,memname
```

or just

```
F  jobanme,R,memname
```

where *memname* is a member in the parmlib with which the Manager was started. The parmlib with which the Manager was started is the dataset on the PARMLIB DD card in the Manager JCL. The Manager's trace options, message filtering routing options, and pools options will be updated from any corresponding MGR_TRACE, MGR_MESSAGE_CONTROL, and MGR_POOLS sections found in that member. You can use the REFRESH command to revert back to the original manager start-up options. You can monitor, modify, and refresh pool allocations on an on-going basis by using the following MVS Console commands:

```
F  jobname,R,memname
```

This command refreshes the three above noted sections from the member *memname* of the current parmlib dataset.

There also exists a POOLS modify command which allows us to monitor pool usage and to add or delete pools and pool allocations dynamically. The commands are:

```
F  jobname,POOLS,STATS
```

This will display a line on the Console (and in the EDM log) for each pool we are using. An example follows:

Pool States

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

- *Size* is the size of the elements of the pool.
- *Number* is the number of elements currently in the pool.
- *Waste* (column 3) is the percentage of waste tolerated at storage element allocation time.
- *HighWM* is the high water mark that shows the maximum number of elements that have been allocated at any one time since start-up or a clear command.
- *InUse* is the number of elements in use when the stats call was made.
- *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 one's available.
- *Waste* (column 8) is the number of times we failed to get an element because the waste was higher than the tolerated waste percent.

  In the above command POOLS can be abbreviated as POOL, POO, PO, or just P and STATS can be abbreviates as STAT, STA, ST or just S so:

  ```
  F  jobname,P,S
  ```

  is equivalent to the above command. There is a DISABLE command that looks like:

  ```
  F  jobname,DISABLE,2048
  ```

  or just:

```
F  jobname,D,2048
```

The above 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 providing the tolerated waste percent is met.

A disabled pool will show up with a "D" after the size on the command. A disabled pool can be re-enabled with the enable command which looks like:

```
F  jobname,E,2048
```

The pool counts can be cleared with the clear command:

```
F  jobname,C,2048
```

The HighWM, Allocs, Empty, and Waste counters are cleared by the above command. Finally 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, the waste tolerated percent value, 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. The pool count will not go to 0 however as long as there are allocated elements. New pools can be added on the fly with the ADJUST command.

Finally, the 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 Manager's use of dynamic memory.

MGR_RETRY

The MGR_RETRY section of the Manager Settings file specifies when (in minutes) a client can attempt another connection to the Manager, after being rejected due to an exceeded task limit or a disabled connection.

MGR_RETRY Settings

SETTING NAME Description
BUSY_RETRY Number of minutes for a client to wait before reconnecting when the Manager is at task limit (TASK_LIMIT).
DISA_RETRY Number of minutes for a client to wait before reconnecting when the Manager has logons halted.

Example

```
[MGR_RETRY]
BUSY_RETRY=1
DISA_RETRY=999
```

Setting Name	Value as	Default	Minimum Value	Maximum Value
BUSY_RETRY		w	0 = allow connect now	999 = do not retry
DISA_RETRY		o y.	0 = allow connect now	999 = do not retry

Note: Novadigm strongly recommends values of at least 1 for these settings to avoid unnecessarily tying up the Manager.

Performance and Usage Considerations

- You can raise both MGR_RETRY values if processing resources are critical. Lowering values will provide greater assurance that connections will be re-established if broken during Client Connects.
- The MGR_RETRY settings should also be coordinated with values in the MGR_NOTIFY section, as well as the MGR_TASK_LIMIT section.

MGR_SMTP_MAIL

The MGR_SMTP_MAIL section of the Manager Settings file specifies the ID of the Manager, and the TCP port number of the Manager to be used.

MGR_SMTP_MAIL Settings

SETTING NAME Description
DNS_SERVER Used to specify the domain name service (DNS) server that is used to query the mail server address.
MAIL_DIR Directory to spool and queue outgoing mail from the SMTP send Manager.
MAIL_TIMEOUT Time-out interval (in seconds) for establishing communications with the mail server.
MAX_TIME_IN_SPOOL Specifies (in minutes) how long the Manager will retain spooled messages before deleting them.
MGR_MAIL_ID Mail ID for the Manager. This setting takes the form of a fully qualified address (e.g., edm@novadigm.com) and has a maximum length of 255 characters. The mail-receiving Manager will reject mail addressed to any other User ID.
RETRY_INTERVAL Specifies (in seconds) how long the Manager will wait before retrying to deliver undelivered mail from the spool.
SMTP_PORT Port to wait for incoming mail.

Examples

INTEL EXAMPLE:

```
[MGR_SMTP_MAIL]
RETRY_INTERVAL = 300
MAX_TIME_IN_SPOOL = 4320
SMTP_PORT=25
MAIL_DIR=D:\MGR\MAIL
MGR_MAIL-ID=edm@novadigm.com
MAIL_TIMEOUT = 60
DNS_SERVER = NONE
```

UNIX EXAMPLE:

```
[MGR_SMTP_MAIL]
RETRY_INTERVAL = 300
MAX_TIME_IN_SPOOL = 4320
SMTP_PORT=25
MAIL_DIR=/edmmgr/MAIL
MGR_MAIL-ID= edm@novadigm.com
MAIL_TIMEOUT = 60
DNS_SERVER = NONE
```

Setting Name	Value as Installed	Default Value
MAIL_DIR	as specified during installation	current directory
MAX_TIME_IN_SPOOL	4320 minutes	4320 minutes (= 3 days)
MGR_MAIL_ID	as specified during installation	sending address
RETRY_INTERVAL	300 seconds	300 seconds
SMTP_PORT	as specified during installation	25
MAIL_TIMEOUT	N/A	60
DNS_SERVER	N/A	NONE

Performance and Usage Considerations

- If storage capacity is an issue, lower the MAX_TIME_IN_SPOOL value. This will decrease the length of time messages are retained.
- Lowering RETRY_INTERVAL values will use more system processing resources.
- The DNS_SERVER setting should be specified to ensure that mail is delivered if you need to send mail outside of your network.

MGR_SNMP

The MGR_SNMP section of the Manager Settings file contains parameters to specify where SNMP traps are to be sent and controls the behavior of the built-in SNMP agent.

The section has the following nine settings:

MGR_MESSAGE_CONTROL Settings

SETTING NAME Description
COMMUNITY_NAME This parameter should be set to a character string. The string will be used as the SNMP community name by the agent. The community name is effectively a password that incoming SNMP transactions must match. A default value of "public" is used if this keyword is not specified. This keyword is only effective when RUN_AS_EXTENSION is set to NO.
RUN_AS_EXTENSION **Note:** This setting is Windows NT specific. This receives a YES or a NO value indicating whether or not the Manager will act as the primary SNMP agent. If YES is coded, NT's SNMP service is the primary SNMP agent and SNMP transactions are processed by an SNMP extension DLL. (In this case, SNMP_COMMUNITY, SNMP_PORT, and SNMP_IP_ADDR are not used, since SNMP port access is handled by NT's SNMP service.) If NO is coded, the Manager will act as the primary SNMP agent. In that case, the SNMP_COMMUNITY parameter should be specified, while SNMP_IP_ADDR and SNMP_PORT may be specified to override their default values.

SETTING NAME Description
SNMP_IP_ADDR This parameter specifies the TCP/IP address of the network adapter card on which the agent is to receive SNMP transactions. When not specified or if specified as 0.0.0.0, any adapter on the machine can be used. The default is that any adapter on the machine can be used. This keyword is only used when RUN_AS_EXTENSION is set to NO and there are several adapters on the machine and a specific adapter to is to receive SNMP transactions.
SNMP_LOGGER_PORT This parameter is used to specify the local TCP/IP port on which the Manager sends its traps. If this parameter is not specified, the Manager uses an ephemeral port, that is, a system assigned port. (An ephemeral port works well in all cases).
SNMP_MANAGER_IP_ADDR **SNMP_MANAGER_IP_ADDR2** **SNMP_MANAGER_IP_ADDR3** The SNMP managers at these IP addresses are authorized to issue GET and SET commands for variables supported by the agent. If all three of these parameters are not specified, any SNMP manager with the correct community name is authorized to run SNMP SET and GET commands on the agent. If any of these three parameters are specified, then only commands coming from those IP addresses will be processed. The SNMP manager specified by SNMP_MANAGER_IP_ADDR is considered the primary SNMP manager. The primary SNMP manager is the manager that will receive all traps generated by the Manager. If this parameter is not specified or if it is set to 0.0.0.0 then traps will not be issued by the Manager.
SNMP_MANAGER_PORT This parameter is used to specify the remote TCP/IP port to which the EDM Manager 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. If not specified, the default is port 161. This keyword is only effective when RUN_AS_EXTENSION is set to NO.

SETTING NAME Description
SNMP_SET_COMMUNITY This parameter may 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. This keyword is only effective when RUN_AS_EXTENSION is set to NO.
SNMP_ZERROR_SEVERITY This parameter is used to specify the severity of ZERROR instances to send as SNMP traps. The trap is sent when the Manager 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.

Examples

```
*-------------------------------------------------*
* MGR_MESSAGE_CONTROL SECTION                     *
* RUN_AS_EXTENSION = YES or NO                    *
*                    YES means NT's SNMP service  *
*                    is the primary SNMP agent and*
*                    EDM SNMP transactions are    *
*                    processed by an EDM SNMP     *
*                    extension DLL.  If set to YES*
*                    COMMUNITY_NAME, SNMP_PORT and*
*                    SNMP_IP_ADDR are not used    *
*                    since SNMP port access is    *
*                    handled by NT's SNMP service *
*                    in this case.                *
*                    NO means the EDM Manager will*
*                    act as the primary SNMP agent.*
*                    In that case, COMMUNITY_NAME *
*                    should be specified while    *
*                    SNMP_IP_ADDR and SNMP_PORT   *
*                    may be specified to override *
*                    their default values.        *
* Example 1:                                      *
* [MGR_MESSAGE_CONTROL]                           *
* NT's SNMP service will be run the SNMP agent port: *
* RUN_AS_EXTENSION = YES                          *
* Address of SNMP manager that receives traps and *
* performs get/sets:                              *
* SNMP_MANAGER_IP_ADDR = 204.7.83.99              *
*                                                 *
* Send traps for ZERROR of severity 8 or more:    *
* SNMP_ZERROR_SEVERITY = 8                        *
*                                                 *
* Example 2:                                      *
* The EDM Manager agent will own the SNMP agent port: *
* RUN_AS_EXTENSION = NO                           *
*                                                 *
* Password for gets:                              *
* COMMUNITY_NAME = AG00021                        *
* Password for sets:                              *
* SET_COMMUNITY_NAME = AG0550081                  *
*                                                 *
* Address of SNMP manager which receives traps and *
* performs get/sets:                              *
* SNMP_MANAGER_IP_ADDR  = 204.7.83.99             *
* Address of second SNMP Manager which can also   *
* performs get/sets:                              *
* SNMP_MANAGER_IP_ADDR2 = 204.7.83.206            *
* COMMUNITY_NAME  = public                        *
*                   The community name is         *
*                   effectively a password        *
*                   that incoming SNMP            *
*                   transactions must match.  The *
*                   above value of "public" is    *
*                   also used as the default      *
```

```
*                    community name when this       *
*                    keyword is not specified.      *
*                    This keyword is only effective *
*                    when RUN_AS_EXTENSION is set   *
*                    to NO.                         *
* SNMP_PORT       = 161                             *
*                    This is the TCP/IP port on     *
*                    which the agent receives SNMP  *
*                    transactions.  If not          *
*                    specified, the default is port *
*                    161.  This keyword is only     *
*                    effective when RUN_AS_EXTENSION*
*                    is set to NO.                  *
* SNMP_IP_ADDR    = 0.0.0.0                         *
*                    This specifies the TCP/IP      *
*                    address of the network adapter *
*                    card on which the agent is to  *
*                    receive SNMP transactions.     *
*                    When not specified the default *
*                    is port 0.0.0.0 which means    *
*                    that any adapter on the machine*
*                    can be used.  This keyword is  *
*                    only used when RUN_AS_EXTENSION*
*                    is set to NO and there are     *
*                    several adapters on the machine*
*                    and a specific adapter to is to*
*                    receive SNMP transactions.     *
*-------------------------------------------------*
* SNMP_MANAGER_IP_ADDR                            *
* SNMP_MANAGER_IP_ADDR2                           *
* SNMP_MANAGER_IP_ADDR3                           *
* The SNMP managers at these IP addresses are     *
* authorized to issue get and set commands for    *
* variables supported by the EDM agent.  If all   *
* three of these parameters are not specified, then*
* any SNMP manager with the correct community name *
* is authorized to run SNMP set and get commands on*
* the EDM agent.  If any of these three parameters *
* are specified, then only commands coming from   *
* those IP addresses will be processed.           *
*                                                 *
* The SNMP manager specified by                   *
* SNMP_MANAGER_IP_ADDR is considered the primary  *
* SNMP manager.  The primary SNMP manager is the  *
* manager which will receive all traps generated  *
* by the EDM Manager.  If this parameter is not   *
* specified or if it is set to 0.0.0.0, then traps *
* will not be issued by the EDM Manager.          *
*                                                 *
* SNMP_LOGGER_PORT                                *
* This parameter is used to specify the local     *
* TCP/IP port on which the EDM Manager sends its  *
* traps.  If this parameter is not specified, the *
* EDM Manager uses an ephemeral port, that is, a  *
* system assigned port.  (An ephemeral port works *
* well in all cases.)                             *
*                                                 *
```

```
* SNMP_ZERROR_SEVERITY                              *
* This parameter is used to specify the severity of*
* ZERROR instances to send as SNMP traps.  The trap*
* is sent when the EDM Manager 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.                                   *
*--------------------------------------------------*
```

Setting Name	Value as Installed	Default Value
RUN_AS_EXTENSION	Y for yes. NT's SNMP service is primary SNMP agent.	N for no. Novadigm's SNMP agent is primary SNMP agent.
COMMUNITY_NAME	public	public
SNMP_PORT	161	161
SNMP_IP_ADDR	0.0.0.0	

Note: If running as an extension, you need the agent running (zsnmpmgr) as well as the extension .dll.

Performance and Usage Considerations

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

MGR_SSL

The MGR_SSL section of the Manager Settings file specifies the operational settings for the SSL Manager. See *Chapter 10: HTTP, SSL, and Multicast Managers* for more information on configuring and using the SSL Manager.

MGR_SSL Settings

Setting Name Description
CA_FILE Used to set the Certificate Authority's certificate. The CA certificate is usually stored in a file in PEM format. The SSL Manager needs a CA certificate to startup. An expired or corrupt CA certificate would prevent the SSL Manager from starting up.
CERTIFICATE_FILE Sets the Manager or the server certificate the certificate is usually stored in a file in PEM format. The value for this keyword should be a valid and existing certificate file. The SSL Manager needs a certificate to startup. An expired or corrupt certificate would prevent the SSL Manager from starting up.
DH_FILE Used to set the Diffie-Hellman parameters. The Diffie Hellman parameters are used to exchange keys using the Ephermal Diffie Hellman method during the SSL handshake.
KEY_FILE Sets the private key. The private key is usually stored in a file in PEM format. The value for this keyword should be a valid and existing key file. Usually the private key might be stored in the same file as the server certificate, in those cases you don't have to specify any value for the key file.
KEY_PASSWORD Used to specify the password 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 the password.

Setting Name Description
SSL_PORT Used to set the port that the SSL Manager should listen on for client connects.
VERIFY_CLIENT Used to specify if the manager should verify the client by requesting a certificate from the client. If the client doesn't have a certificate the connection would be dropped.

Example

```
[MGR_SSL]
CERTIFICATE_FILE = w:\openssl\ms\srvcert.pem
KEY_FILE = w:\openssl\ms\srvprvk.pem
KEY_PASSWORD = violin
DH_FILE = w:\openssl\ms\dhparams.pem
SSL_PORT = 3456
CA_FILE = w:\openssl\ms\cacert.pem
VERIFY_CLIENT = Y
```

Setting Name	Value as Installed	Default Value
CERTIFICATE_FILE	N/A	
KEY_FILE	N/A	
KEY_PASSWORD	N/A	
DH_FILE	N/A	
SSL_PORT	N/A	
CA_FILE	N/A	
VERIFY_CLIENT	N/A	

Performance and Usage Considerations

- You must have CMDLINE=(zsslmgr) RESTART=YES in the MGR_ATTACH_LIST for the HTTP Manager to function.

MGR_STARTUP

Note: Do not change MGR_ID, MANAGER_TYPE, or MGR_NAME after a successful install unless instructed to do so by Novadigm Technical Services.

The MGR_STARTUP section of the Manager Settings file specifies the ID and type of the Manager, communications settings for the Manager, as well as other startup information. In addition, if you are processing both EDM and Radia Clients with a single Manager (multi-mode Manager), you can define the resolution start point for each type of client. Use the EDM_STARTUP section to define start domain and start class for all the EDM Clients, the RADIA_STARTUP section for all the Radia Clients. This way the resolution start point may be different depending on the type of the connection.

MGR_STARTUP Settings

SETTING NAME Description
ALLOW_DUPLICATE_ IP_ADDRESS Values are YES or NO. A setting of NO will cause the Manager to reject a second login (edmdemon session) with the Manager when one is already active from the same IP address. YES will allow multiple concurrent IP connections from the same Client IP address.
BYTE-LEVEL_DIFF Turns on byte-level differencing during resolution. Values are YES or NO.
LU62_TPNAME LU6.2 Transaction Program Name (TPN).
LU62ACB LU6.2 Application Control Block (ACB).
MANAGER_TYPE Type of Manager: values are DISTRIBUTED, SERVER, or STANDALONE.

Note: The LU62_ TPNAME setting may be overridden by specifying an address in the MGR _ATTACH_LIST.

SETTING NAME Description
MGR_ID Qualifier for Manager log file. Format: 3-character hexadecimal. This ID is also passed to the EDM Client as ZOBJMID variable, and is used in the database to identify the source manager in a distributed Manager environment. Values 001-EFF are available for use. Values F00-FFF are reserved.
MGR_NAME Manager name.
MSPX-SOCKET Socket to listen on for Microsoft SPX/IPX client connections. NT specific.
NETB_LANANUM NetBIOS LAN adapter number. Must be a valid NetBIOS LAN adapter number. Values supported are 0 to 256 NT specific and OS/2 specific.
NETB_NAME NetBIOS name. Must be a valid NetBIOS name. NT and OS/2 specific.
NFY6ACB LU6.2 Notify Application Control Block.
SHOW_VERINFO Displays version information in the Manager log at startup. Values are YES or NO.
SIPX_SOCKET Socket to listen on for Novell SPX/IPX EDM Client connections. NT specific and OS/2 specific.
TCP_PORT Port to listen on for TCP/IP Client connections. Must be greater than or equal to 1029 and less than 64 KB.

Note: The TCP_PORT setting may be overridden by specifying an address in the MGR _ATTACH_LIST.

SETTING NAME Description
VERBOSE Sends messages to screen (stderr) when Manager starts up, verifies the license, verifies the database, loads cache, IP Managers (tcp, http, ssl and multicast) are ready for processing and when shutting down. Values are YES or NO.

Example

```
[MGR_STARTUP]
MANAGER_TYPE = DISTRIBUTED
MGR_ID = 001
MGR_NAME = EDM
SHOW_VERINFO = YES
SIPX_SOCKET = 3460
TCP_PORT = 3460
NETB_LANANUM = 0
NETB_NAME = EDMNET
LU62ACB = EDM
LU62_TPNAME = EDMTRANS
NFY6ACB = EDM62M
BYTE-LEVEL_DIFF = NO
VERBOSE = NO
```

Setting Name	Value as Installed	Default Value	Minimum Value	Maximum Value
ALLOW_DUPLICATE_IP_ADDRESS	YES	YES	N/A	N/A
BYTE-LEVEL_DIFF=	N/A	NO	N/A	N/A
LU62_TPNAME	N/A	N/A	N/A	N/A
LU62ACB	N/A	N/A	N/A	N/A
MANAGER_TYPE	As specified during installation	DISTRIBUTED	N/A	N/A
MGR_ID	As specified during installation	001	000	EFF VALUES F00-FFF are reserved
MGR_NAME	As specified during installation	EDM	N/A	N/A
NETB_LANANUM	As specified during installation	0	0	256
NETB_NAME	As specified during installation	EDMNET	N/A	N/A
NFY6ACB	N/A	N/A	N/A	N/A
SHOW_VERINFO	N/A	YES	N/A	N/A
SIPX_SOCKET	Installation default is 3460 unless otherwise specified during installation	1029	System dependent	System dependent
TCP_PORT	Installation default is 3460 unless otherwise specified during installation	1029	System dependent	64 KB
VERBOSE	N/A	NO	N/A	N/A

Performance and Usage Considerations

- If your MGR_TYPE is not set for DISTRIBUTED, you will not be able to use the Distributed Manager Adapter functionality.
- The MGR_ID value must be a three-character hexadecimal string. (Hexadecimal characters include: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.)
- You must include zlu62mgr in the MGR_ATTACH_LIST to configure and use the LU62ACB and LU62_TPNAME settings.
- You must include znfy6mgr in the MGR_ATTACH_LIST to configure and use the NFY6ACB setting.
- VERBOSE will be disabled if running the Manager as an NT Service.

EDM_STARTUP

The EDM_STARTUP section of the Manager Settings file specifies the resolution starting point for EDM clients in a multi-mode manager environment.

EDM_STARTUP Settings

SETTING NAME Description
START_CLASS The class starting point for resolution for EDM Clients.
START_DOMAIN The domain starting point for resolution for EDM Clients.

Example

```
[EDM_STARTUP]
START_DOMAIN=ZSYSTEM
START_CLASS=ZPROCESS
```

Setting Name	Value as Installed	Default Value
START_DOMAIN	N/A	ZSYSTEM
START_CLASS	N/A	ZPROCESS

RADIA_STARTUP

The RADIA_STARTUP section of the Manager Settings file specifies the resolution starting point for RADIA Clients in a multi-mode Manager environment.

RADIA_STARTUP Settings

SETTING NAME Description
START_CLASS The class starting point for resolution for Radia Clients.
START_DOMAIN The domain starting point for resolution for Radia Clients.

Example

```
[RADIA_STARTUP]
START_DOMAIN=SYSTEM
START_CLASS=PROCESS
```

Setting Name	Value as Installed	Default Value
START_DOMAIN	N/A	SYSTEM
START_CLASS	N/A	PROCESS

MGR_TASK_LIMIT

The MGR_TASK_LIMIT section of the Manager Settings file specifies the maximum number of the following:

- Concurrent tasks allowed including Manager tasks and Client Connects.
- Lines that can be written to the Manager log per client.
- Resolutions allowed per client.

MGR_TASK_LIMIT Settings

SETTING NAME Description
TASK_HEAP_SIZE This setting controls the heap size of a task. (used in conjunction with MGR_POOLS)
TASK_LIM The maximum number of concurrent tasks for the Manager. These tasks include all Manager system tasks, tasks specified in ATTACH_LIST, and all the tasks created as a result of the connect process initiated by clients and administrators. No connection to the Manager is accepted once this limit is reached.
TASK_LOG_LIM The maximum number of lines that can be written to the Manager log per client.
TASK_RESO_LIMIT The maximum number of resolutions allowed per client.
TASK_STACK_SIZE This setting defines how many variables may be used per task.

Example

```
[MGR_TASK_LIMIT]
TASKLIM         = 20
TASK_LOG_LIM    = 0
TASK_RESO_LIM   = 64000
TASK_STACK_SIZE = 64000
TASK_HEAP_SIZE  = 0
```

Setting Name	Value as Installed	Default Value	Minimum Value	Maximum Value
TASK_HEAP_SIZE	0	0	0	4G
TASK_LIM			15	
TASK_LOG_LIM	0	100000 Lines	0	N/A
TASK_RESO_LIMIT	64000	64000 Resolutions	1	64000
TASK_STACK_SIZE	64000	64000	16384	64000

Warning: Do not change the TASK_STACK_SIZE setting in this section unless a member of the Novadigm Technical Services group instructs you to do so.

Performance and Usage Considerations

- The MGR_TASK_LIMIT settings should also be coordinated with values in the MGR_NOTIFY section, as well as the MGR_RETRY section.
- TASK_HEAP_SIZE is used only for MVS.
- TASK_STACK_SIZE will affect the depth of resolution. Higher values will result in deeper resolution.

MGR_TIMEOUT

The MGR_TIMEOUT section of the Manager Settings file specifies (in seconds) how long the Manager will wait for a request from a connected Client before disconnecting that Client due to inactivity (no requests/responses from the Client).

MGR_TIMEOUT Settings

SETTING NAME Description
ADMIN_TIMEOUT Time-out (in seconds) for Administrator System Explorer functions.
SEND_THROTTLE Time-out (in milliseconds) before each send.
TIMEOUT_COMM Communications (receive) time-out (in seconds).
TIMEOUT_NCOM Non-communications time-out (in seconds) between two conversations in the LU6.2 protocol.

Example

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

Setting Name	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
TIMEOUT_NCOM	0=Never Time-out	0=Never Time-out	0	32767

Performance and Usage Considerations

- You can raise MGR_TIMEOUT values if processing resources are critical.
- The MGR_TIMEOUT settings should also be coordinated with values in the MGR_RETRY section.
- The SEND_THROTTLE value can be overridden by Bandwidth Throttling variables sent up in a client object.
- TIMEOUT_COMM is the Manager analog to the Client ZTIMEO. If a connect is active and the Manager has not received any data from a Client for the TIMEOUT_COMM value, then the Manager will terminate the session.

MGR_TPINIT

The MGR_TPINIT section of the Manager Settings file specifies packet sizes to send to clients.

MGR_TPINIT Settings

SETTING NAME Description
BUFLU62 LU6.2 buffer size used for send/receive. (MVS only)
BUFMSPX Microsoft SPX/IPX buffer size used for send/receive. (NT specific)
BUFNETB NetBIOS buffer size used for send/receive. (NT and OS/2 only)
BUFSIPX SPX/IPX buffer size used for send/receive. (NT specific)
BUFTCP TCP buffer size used for send/receive.
GET_REMOTE_HOST_NAME For use in a dynamic TCP/IP environment (DNS, DHCP, etc.). Default is NO.
MAXREC Maximum record size.

Warning: Do not change any setting in this section unless a member of the Novadigm Technical Services group instructs you to do so.

Example

```
[MGR_TPINIT]
BUFTCP  = 12288
BUFNETB =  4096
BUFLU62 = 12288
BUFSIPX = 10240
MAXREC  =  6144
GET_REMOTE_HOST_NAME = NO
```

Setting Name	Value as Installed	Default Value
BUFTCP	12288	12288
BUFLU62	1228	1800
BUFNETB	4096	4096
BUFSIPX	10240	4096
MAXREC	6144	6144
GET_REMOTE_HOST_NAME	NO	NO

Performance and Usage Considerations

- The buffer sizes reflected in the MGR_TPINIT settings should be the same for both Manager and Clients.
- Any buffer-size settings in the MGR_TPINIT section should only be changed (after being directed by a member of the Novadigm support staff) in coordination with equivalent changes to clients.
- Do not use GET_REMOTE_HOST_NAME if you are not in a dynamic TCP/IP environment. This will cause the Manager to expend unnecessary processing time attempting to associate Remote Host Name.
- If GET_REMOTE_HOST_NAME.=YES, the Remote Host Name will appear in the each associated line of the Manager log in place of the IP address.

MGR_TRACE

The MGR_TRACE section of the Manager Settings file controls and influences diagnostic logging for the Manager. This section contains a list of keywords that you can specify.

All diagnostic output produced by TRACE keywords is written to the active Manager log. To activate a TRACE keyword, type `YES`. To deselect a TRACE keyword, type `NO`.

TRACE keywords specified in the MGR_TRACE section are invoked at Manager initialization and remain in effect until changed by altering the MGR_TRACE setting and restarting the Manager or while the Manager continues to run using the Console, REXX, or by changing a ZCVT value. The trace settings that are in effect at Manager initialization are displayed at the beginning of the Manager log.

Note: Beginning with the EDM Manager 4.10, the operation of the trace flags is altered. The settings for each keyword are evaluated in the order in which they are presented in the Manager Settings. The results can be non-intuitive. For example, if ALL=YES is the first value specified, and each following keyword is set to "NO", the effect is to *turn off tracing*. If ALL=YES is the last trace keyword specified, then ALL tracing will be active. If ALL=NO is the last entry, then all tracing is turned off.

The Manager TRACE Keyword List

TRACE Keywords	Definitions
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.
CPIC	Traces LU6.2 protocol flows. MVS only.
DATA	Traces data buffers to or from the client.
DES	Traces encryption/decryption.
DMA	Traces distributed Manager activity.
DSCOMP	Traces data stream compression. MVS only.
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 Manager method execution/return codes.
MSPX	Traces Microsoft SPX/IPX protocol activity.
NETBIOS	Traces NetBIOS activity.
NOTIFY	Traces notify processing.

TRACE Keywords	Definitions
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 data base activity.
PROMOTE	Traces file promotion.
RESOURCE	Traces resource file activity.
REXX	Traces REXX environment.
REXXOFF	Suppresses all REXX activity.
SIPX	Traces Novell SPX/IPX protocol activity.
STORAGE	Traces storage in conjunction with the MGR_LOG's STORAGE_INTERVAL setting.
STATS	Traces statistics. HIGHLY RECOMMENDED TO BE 'YES'.
SUBST	Traces variable substitution.
TCP	Traces TCP/IP activity.
TEST	-Reserved-
TRAN	Traces data transformation buffers. MVS only.
VAR	Traces the variable references.
VARSTG	Traces variable processing storage usage.
VARSUB	Traces variable substitution activity.
VSAM	Traces VSAM I/O. MVS only.
VSAMDATA	Traces VSAM data. MVS only.
VSAMRPLS	Traces VSAM request parameter list (RPL). MVS only.
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
CPIC     = NO
DATA     = NO
DES      = NO
DMA      = NO
DSCOMP   = NO
ENQDEQ   = NO
EXPL     = NO
FILE     = NO
IMPL     = NO
LOOKASID = NO
METHOD   = NO
MSPX     = NO
NETBIOS  = 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
SIPX     = NO
STATS    = NO
STORAGE  = NO
SUBST    = NO
TCP      = NO
TEST     = NO
TRAN     = NO
VAR      = NO
VARSTG   = NO
VSAM     = NO
```

```
VSAMACB  = NO
VSAMAPI  = NO
VSAMRPLS = NO
YEAR2000 = NO
ALL      = YES
```

Performance and Usage Considerations

- Turning on trace flags generates a large number of Manager log messages. This will degrade the performance of the Manager due to the disk I/O load. However, this may be necessary at times for problem resolution. Ensure that your Manager log is properly configured (MGR_LOG).
- Tracing can be clustered in order to troubleshoot a particular aspect of Manager operations while at the same time preserving an appropriate level of logging activity. For example, turning on such flags as OBJCRC, OBJRES, OBJRES1, OBJRESO, and OBJXFER can help identify problems that might be occurring during object resolution. Likewise, CPIC, DATA, MSPX, SIPX, NETBIOS, TCP, and TRAN focus on communications activities. (Note that some of these traces pertain to specific protocols.) Special purpose trace flags include DMA, NOTIFY, REXX, and METHOD.
- The 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 Manager's use of dynamic memory.
- The storage trace (ALL=YES) does not include REXXOFF, TRACELOG or any VSAM trace settings.

MVS User's Note: You can refresh options set from the MVS parmlib with the following command:

```
F  jobname,REFRESH,memname
```

or just

```
F  jobanme,R,memname
```

where *memname* is a member in the parmlib the Manager was started. The parmlib with which the Manager was started is the dataset on the PARMLIB DD card in the Manager JCL. The Manager's trace options, message filtering, routing options, and pools options will be updated from any corresponding MGR_TRACE, MGR_MESSAGE_CONTROL, and MGR_POOLS sections found in that member.

Both ALL=YES and VSAM=YES now include the VSAMAPI trace. This change has been made because the VSAMAPI trace is no longer as voluminous as it previously was. It now issues only a single one-line message on entry and one on exit to and from a VSAM module. Since it shows the FDCI key the VSAM requests are for it is quite useful in understanding a program's database access.

The other VSAM traces, VSAMRPLS, and VSAMDATA have not changed. They are not set by ALL=YES and should be used only when specifically requested by development. Note that VSAMRPLS also has an alias name of VSAMCB.

MGR_TRANSLATION (MVS only)

The MGR_TRANSLATION section of the Manager Settings file specifies customized ASCII to EBCDIC and EBCDIC to ASCII translation tables.

MGR_TRANSLATION Settings

SETTING NAME Description
A2E xx = yy Specifies conversion from ASCII to EBCDIC. The hex value(s) yy…will replace the standard default translation table values beginning at offset xx.
E2A xx = yy… Specifies conversion from EBCDIC to ASCII. The hex value(s) yy…will replace the standard default translation table values beginning at offset xx.

Notes

- Offsets from 00 to FF can be specified in any order.
- Any individual character or groups of characters can be replaced in the default table.
- In case of duplicate or overlapping settings, the last value specified will be used.
- Translations should always be symmetrical (i.e., ASCII 30 translates to EBCDIC F0 and EBCDIC F0 translates back to ASCII 30).

Setting Name	Value as Installed	Default Value
A2E XX	See example	See example
E2A XX	See example	See example

Example

```
[MGR_TRANSLATION]
* The default ASCII to EBDCIC translation table
A2E 00 = 00010203372D2E2F1605250B0C400E0F
A2E 10 = 101112133C3D322618193F27221D351F
A2E 20 = 405A7F7B5B6C507D4D5D5C4E6B604B61
A2E 30 = F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F
A2E 40 = 7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6
A2E 50 = D7D8D9E2E3E4E5E6E7E8E9BAE0BBB06D
A2E 60 = 798182838485868788899919293949596
A2E 70 = 979899A2A3A4A5A6A7A8A9C06AD0A107
A2E 80 = 040608090A1415171A1B1C1E20212324
A2E 90 = 28292A2B2C303133343638393A3B3E41
A2E A0 = 424344454647484 94A51525354555657
A2E B0 = 585962636465666768697071727374 75
A2E C0 = 767778808A8B8C8D8E8F909A9B9C9D9E
A2E D0 = 9FA0AAABACAEAFB0B1B2B3B4B5B6B7B8
A2E E0 = B9BABBBCBEBFCACBCCCDCECFDADBDCDD
A2E F0 = DEDFE1EAEBECEDEEEFFAFBFCFDFEFF40


* The default EBDCIC to ASCII translation table
E2A 00 = 000102038009817F8283840B0C0D0E0F
E2A 10 = 10111213858608871819888 98A1D8B1F
E2A 20 = 8C8D1C8E8F0A171B9091929394050607
E2A 30 = 95961697981E99049A9B9C9D14159E1A
E2A 40 = 209FA0A1A2A3A4A5A6A7A82E3C282B7C
E2A 50 = 26A9AAABACADAEAFB0B121242A293B5E
E2A 60 = 2D2FB2B3B4B5B6B7B8B97C2C255F3E3F
E2A 70 = BABBBCBDBEBFC0C1C2603A2340273D22
E2A 80 = C3616263646566676869C4C5C6C7C8C9
E2A 90 = CA6A6B6C6D6E6F707172CBCCCDCECFD0
E2A A0 = D17E737475767778797AD2D3D45BD5D6
E2A B0 = 5ED8D9DADBDCDDDEDFE05B5DE35DE4E5
E2A C0 = 7B41424344454 6474849E6E7E8E9EAEB
E2A D0 = 7D4A4B4C4D4E4F505152ECEDEEEFF0F1
E2A E0 = 5CF2535455565758595AF3F4F5F6F7F8
E2A F0 = 303132333435363738 39F9FAFBFCFDFE
```

Performance and Usage Considerations

- Once established, extreme care should be used if the MGR_TRANSLATION is changed. Changing the translation after adding or updating anything in the database, could cause data to become unreadable or unviewable.

- All managers in a DMA group must use the same MGR_TRANSLATION.

MGR_USERLOG

Note: MVS is not supported.

The MGR_USERLOG section of the Manager Settings file specifies the logging directory and logging options for the user logging facility.

MGR_USERLOG Settings

SETTING NAME Description
ACTIVATE Activate user log at Manager start-up. Values for this setting are Yes or No.
DIRECTORY Fully qualified directory path where the user log is written.
FLUSH_SIZE Buffer size (in bytes) that the logger uses to accumulate log messages before writing to the user log file.
MESSAGE_WIDTH The maximum width (in characters) of the messages in the user log.
PIPE_SIZE Maximum memory size (in bytes) of log messages processed before logged.
THRESHOLD Threshold number of log messages that will be written to a log before automatically switching to the next log. When limit is reached, new log files are created. Specify a negative number to overwrite log file when limit is reached.

Examples

UNIX EXAMPLE:

```
[MGR_USERLOG]
ACTIVATE      = NO
DIRECTORY     = /edmmgr/log
THRESHOLD    = 5000000
FLUSH_SIZE    = 256
MESSAGE_WIDTH = 256
PIPE_SIZE     = 1000000
```

INTEL EXAMPLE:

```
[MGR_USERLOG]
ACTIVATE      = NO
DIRECTORY     = D:\MGR\LOG
THRESHOLD    = 5000000
FLUSH_SIZE    = 256
MESSAGE_WIDTH = 256
PIPE_SIZE     = 1000000
```

Setting Name	Value as Installed	Default Value	Minimum Value
ACTIVATE	NO	NO	N/A
DIRECTORY	/edmmgr/log	current directory	N/A
THRESHHOLD	-5000000 bytes	5000 bytes	1
FLUSH_SIZE	256 bytes	100000 bytes	1
MESSAGE_WIDTH	256 characters	90	80
PIPE_SIZE	1000000 bytes	65535 bytes	1

Performance and Usage Considerations

- Increasing the buffer size (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 utilization, care must be taken to be certain the maximum amount of memory or storage space is available.

SECTION_DELIMITERS (MVS only)

Note: If you use SECTION_ DELIMITERS, it must be the very first entry in the Manager Settings file.

The SECTION_DELIMITERS section of the Manager Settings file specifies the prefix and suffix used to delimit section names in the MVS Manager PARMLIB. The SECTION_DELIMITERS entry must be the first non-blank line in the PARMLIB. If it is not, the Manager will not be able to read in the license string and will not start up.

SECTION_DELIMITERS Settings

SETTING NAME Description
SECTION_DELIMITER= Establish the type of characters used to distinguish separate sections in the Manager Settings. The format is SECTION_DELIMITER=*xy,* where *x* is the left delimiter and *y* is the right delimiter.

Examples

```
SECTION_DELIMTERS=< >
```

The result of the above setting is as follows:

```
<MGR_LICENSE>
LICENSE_STRING= FFCDAB
```

Setting Name	Value as Installed	Default Value
SECTION_DELIMITER=		[ ]

Warning: The SECTION_DELIMITERS entry must be the first non-blank line in the PARMLIB. If it is not, the Manager will not be able to read in the license string and will not start-up.

Performance and Usage Considerations

- Use care when refreshing sections and remember to use SECTION_DELIMITERS explicitly.

Part II

Operating the Manager

Part II: Operating the Manager provides information about day-to-day Manager operations. The flow of Manager processing can be customized using Manager REXX programs and Manager Methods (Chapter 2). There are a number of ways that you can notify clients, including drag-and-drop Notify and EDMPUSH (Chapter 3). The Manager log provides useful information about Manager Operations from startup to shutdown (Chapter 4).

Part II: Configuring the Manager contains three chapters:

- *Chapter 2: Managing Processing*
- *Chapter 3: Notifying Clients*
- *Chapter 4: Logs and Messages*

2

Managing Processing

This chapter discusses Manager processing and shows you how to customize processing flow using Manager REXX programs and Manager Methods.

Manager Operations

Manager operations has three basic phases:

- Startup
- Processing Requests
- Shutdown

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

A request can be sent to the Manager as a system command, an in-bound object, a client connection, an Admin transaction or a DMA command. During the Process Requests phase, the Manager performs the requests (tasks) that are submitted to it. There are four types of tasks, System tasks, Client tasks, Admin tasks and DMA tasks. System tasks are Manager functions, Client tasks are client requests, Admin tasks pertain to System Explorer or Publisher operations and DMA tasks are Distributed Manager functions. The type of task is shown in each line of the Manager log as seen below.

```
EDM1955I 13:49:52 [208.244.225.166 /16B] EDMV4 Client
    Max size of local memory allocated      : 230805

EDM0999I 13:50:30 [ztoptask  /17B]    System Task
    REXX Method <D:\DEV\MGR\REXX\ZSHUTDWN> with parms <>
    started at <13:50:30:574>
```

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

Customizing Manager Processing

The values in your Manager Settings file allow you to customize the overall configuration of the Manager. There are two main ways of customizing the flow of Manager processing, Manager REXX programs and Manager methods. The main difference between the two is that Manager REXX programs are pre-architected, while Manager methods can be inserted anywhere in the flow of processing. The following sections describe Manager REXX programs and Manager methods.

Manager REXX Programs

There are six major pre-architected event points at which the Manager issues calls to standard REXX programs:

Event Points

- Manager Startup - ZSTARTUP
- Manager Startup - ZPCACHE
- Manager Startup - ZINIT
- Task Start - ZTASKSTA
- Task End - ZTASKEND
- Notify Start - ZNFYxSTA
- Notify End- ZNFYxEND
- Log Switch - ZLOGSWCH
- Log Wrap - ZLOGWRAP
- Manager Shutdown - ZSHUTDWN

REXX Programs

ZSTARTUP

This is called just before the Manager is enabled to accept and process client connections. ZSTARTUP does not pass any parameters to the REXX and does not accept any information from this REXX. It can be used to perform some user-defined

specific processing previous to allowing connections to be initiated only.

ZSHUTDWN

This is called just before the Manager shuts down. It can be used to perform some user-defined specific processing previous to allowing connections to be initiated only.

ZTASKSTA

This is called when each connection is first accepted. It is passed a single parameter that contains the protocol level identifier for the client (TCPIP Address or LU NAME for SNA).

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 parsable sub-paramaters that are positionally dependent:

- LUNAME => TCPIP address or LUNAME of Client
- Connect termination reason
- USERID of the current connection
- Total number of all object instances of any type processed during this connection
- Total number of all 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 characters transmitted from the manager to the client during this connection (non-compressed count)
- Total number of FILES transferred from the Manager to the Client during this connection

ZNFYxSTA

This is called at the beginning of Notify processing for each individual Client being notified. The **x** defines the type of Notify (6 for LU62, T for TCP, or D for Dial-up). The set of parameters passed includes:

- **UINF=<value1>**
  user info passed via EDMMPUSH method. If no info 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 individual Client being notified. The **x** defines the type of Notify (6 for LU62, T for TCP or D for Dial-up). The set of parameters passed includes:

- **UINF=<value1>**
  user info passed via EDMMPUSH method. If no info 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 re-scheduled for retry.

There are three additional points at which REXX programs could be positioned:

ZLOGWRAP

This can be called when log wrap occurs (the log file is re-used) to insert a user defined command (e.g., zip the old log file and save it).

ZLOGSWCH

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

ZPCACHE

This can be called after cache is loaded during Manager start-up.

Note: This is not configurable. Please consult with Customer Support before using this REXX.

ZINIT

This is called during Manager startup to run REXXs and test for certain conditions. If the conditions are not met (return code = 16), Manager startup will be halted.

Manager Methods

A method is a program or procedure that can be packaged and exchanged as an object, specifically as an instance of the ZMETHOD (or METHOD—RADIA) class. By connecting an instance of this class to another class instance, you can specify where and when that procedure will run. You can also run a method from a REXX script, enabling you to execute methods outside of the object resolution process. The following is an example of the format used to execute a method in this way.

```
ADDRESS EDMLINK EDMMCMPR 'ZTEST    '
```

As you can see, EDMLINK is itself a method, allowing you to process other methods. EDMLINK returns the return code of the invoked method. The format for EDMLINK is, as shown above:

```
ADDRESS EDMLINK (method name) '(Parameter associated
     with Method)'
```

Manager methods allow you to manipulate objects and database components at the system (Manager) level as opposed to the Client or individual workstation objects. Database components are those entities (files, domains, classes, instances, and variables) that reside in the database. In-storage objects are used or created during the object resolution process.

The Manager methods that affect objects and database entities are as follows:

Effect of Manager Methods on Objects and Database Entities

Method Name	Object	Database Entity
EDMMAILQ	✓	N/A
EDMMALLO	N/A	N/A
EDMMCACH	✓	
EDMMCMPR	✓	
EDMMCOPY	✓	
EDMMDCLA		✓
EDMMDALO	N/A	N/A
EDMMDB		✓
EDMMDELI	✓	
EDMMDELV	✓	
EDMMDINS		✓
EDMMDOBJ	✓	
EDMMDPRO		✓
EDMMENCR		✓
EDMMEXIS		✓
EDMMGPRO		✓
EDMMNFTY	N/A	N/A
EDMMOLOG	✓	
EDMMPHIS	✓	
EDMMPPRO	✓	
EDMMPROM		✓
EDMMPUSH	N/A	N/A

Method Name	Object	Database Entity
EDMMRESO	✓	
EDMMRPRO		✓
EDMMSGNR	N/A	N/A
EDMMSGNR	N/A	N/A
EDMMSORT	✓	
EDMMTUCH		✓
EDMMULOG	N/A	N/A
EDMMVDEL	✓	
EDMMVGBL	✓	
EDMMXREF	✓	

Methods are often used in conjunction with other methods to achieve a specific purpose. For example, you can use the EDMMDOBJ method to delete an in-storage object, then execute EDMMCOPY to copy an object, giving it the original object name.

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

The default file and domain used by some methods can be specified in the DBASE and DOMAIN values of the MGR_STARTUP member/setting of the Manager Settings. (MVS PARMLIB, EDMPROF.DAT on NT and OS/2, or ~/.edmprof on Unix Systems).

Alias and Method Names

The names of the Manager methods are provided in two ways: alias names (MVS and Unix), and method names (all platforms). Note that support for alias names will be dropped in future releases.

Manager Method Alias Names

Alias Name	Method Name
NEW	EDMMAILQ
NEW	EDMMCACH
NEW	EDMMDALO
NEW	EDMMDB
NEW	EDMMPUSH
NEW	EDMMRPRO
NEW	EDMMULOG
EDMSIGN	EDMMSIGN
EDMSIGNR	EDMMSGNR
ZDCLASS	EDMMDCLA
ZDELINS	EDMMDINS
ZDELOBJS	EDMMDOBJ
ZDELPROF	EDMMDPRO
ZEXIST	EDMMEXIS
ZGETPROF	EDMMGPRO
ZIRXALOC	EDMMALLO
ZNOTIFY	EDMMNFYT
ZOBJCMPR	EDMMCMPR
ZOBJCOPY	EDMMCOPY
ZOBJDELI	EDMMDELI
ZOBJDELV	EDMMDELV
ZOBJSORT	EDMMSORT
ZPROMANY	EDMMPROM
ZPUTHIST	EDMMPHIS
ZPUTPROF	EDMMPPRO
ZSIMRESO	EDMMRESO
ZTOUCH	EDMMTUCH

Alias Name	Method Name
ZVARDEL	EDMMVDEL
ZVARGBL	EDMMVGBL
ZVARLOG	EDMMOLOG
ZXREF	EDMMXREF

Method Naming Standards

Note: The Manager continues to support methods using either the alias names (for MVS and Unix) and the Manager method names. That is, both the ZDELOBJS and EDMMDOBJ continue to be supported. However, Novadigm recommends that users plan to implement the EDMM*xxxx* method naming standard as the alias names will no longer be supported or provided in Manager releases subsequent to Version 4.

The method naming standards are structured similarly to EDMMDOBJ, as follows:

EDMMDOBJ

Symbol	Definition
EDM	Identifies the method as a Novadigm method.
M	Identifies the method as a Manager method.
DOBJ	Represents an abbreviation that denotes the function for which the method can be employed.

The following table provides you with a list of methods, including both the alias and method names, as well as a brief description.

Manager Methods Defined

Method Name	Alias Name	Description
EDMMAILQ	new	Deposits e-mail in the mail queue so it can be sent to a remote system user.
EDMMALLO	new	Allocates an external data set (MVS only).
EDMMCACH	new	Refreshes or disables cache.
EDMMCMPR	ZOBJCMPR	Compresses an in-storage object.
EDMMCOPY	ZOBJCOPY	Copies an in-storage object.
EDMMDCLA	ZDCLASS	Deletes a class from the database.
EDMMDALO	new	De-allocates an external data set (MVS only).
EDMMDB	new	Locks and unlocks database against all components except DMA and Console.
EDMMDELI	ZOBJDELI	Deletes an instance from an in-storage object.
EDMMDELV	ZOBJDELV	Deletes a variable from all instances of an in-storage object.
EDMMDINS	ZDELINS	Deletes an instance or instances from within a database class.
EDMMDOBJ	ZDELOBJS	Deletes an in-storage object.
EDMMDPRO	ZDELPROF	Deletes an object in the PROFILE database.
EDMMEXIS	ZEXIST	Verifies the existence of a given class or instance in the database.
EDMMGPRO	ZGETPROF	Creates an in-storage object from the PROFILE database.
EDMMNFYT	ZNOTIFY	Executes a TCP/IP notification on a client.

Method Name	Alias Name	Description
EDMMOLOG	ZVARLOG	Displays the contents of an in-storage object.
EDMMPHIS	ZPUTHIST	Puts an in-storage object into the HISTORY database.
EDMMPPRO	ZPUTPROF	Puts an in-storage object into the PROFILE database.
EDMMPROM	ZPROMANY	Adds an instance to or updates an instance in the database.
EDMMPUSH	new	Puts an inbound object into a Notify queue.
EDMMRESO	ZSIMRESO	Resolves specified objects.
EDMMRPRO	new	Adds, updates, or deletes instances in PRIMARY database based on the variables of an in-storage object.
EDMMSIGN	EDMSIGN	Authenticates users against the database.
EDMMSGNR	EDMSIGNR	Authenticates users against external security systems.
EDMMSORT	ZOBJSORT	Sorts instances, by stems, of in-storage objects.
EDMMTUCH	ZTOUCH	Updates the date/time stamp of an instance.
EDMMULOG		Used to write to the user log file.
EDMMVDEL	ZVARDEL	Deletes all in-storage objects.
EDMMVGBL	ZVARGBL	Migrates values from one in-storage object to another and deletes the source object.
EDMMXREF	ZXREF	Cross-references class and instance usage during the object resolution process.

"MUST RUN" METHODS

When you configure methods to run during the resolution process, you expect to achieve specific outcomes. If a method is intended to work in conjunction with another method (as noted on page 117) or has a direct effect on the correct outcome of the resolution, the entire resolution process may be dependent on first the existence, then, the successful launching of this method.

You can designate a method as "Must Run", which means that the Manager will determine that it exists and can be run before continuing with the resolution process. If it is not found or can not 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 Manager does not recognize that method 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 Manager log) and the end result of the resolution path.

To configure a method as "Must Run", insert the ZMUSTRUN variable in the ZMETHOD instance and set to 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 you do not specify a value for MSGONERR, the following will appear: "CONFIGURATION UNCHANGED! UNABLE TO DETERMINE NEW CONFIGURATION".

Novadigm REXX Extensions

There are a number of Novadigm REXX extensions that perform specific functions. These functions include object resolution and the getting and setting of variables and objects. The following are the Novadigm REXX extensions:

- EDMRESO
- EDMGET
- EDMGETV
- EDMSET
- EDMSETV

EDMRESO
(resolve_string)

Function: To perform object resolution.

Returns: OK: rc=0, eval-string= obj.var value

err: rc>0
(in OS/2 causes Error 40: Incorrect call to routine)

Example:
```
ca'PRIMARY.SYSTEMX.USER.USER1(EDMSETUP)'
call EDMRESO 'USER.USER1(EDMSETUP)'
rc = EDMRESO 'USER.USER1(EDMSETUP)'
RC = EDMRESO 'USER ll
EDMRESO.USER1(EDMSETUP)'
```

EDMGETV
(objname,varname,heapnumber,resolveflag)

Function: To get a variable from an object.

heapnumber and resolveflag are optional parameters.

heapnumber defaults to 0 which means current heap.

resolveflag defaults to 0 which means no substitution.

The resolve flag indicates that substitution should be carried out on the variable. The resolve flag is a one-character field (no blanks allowed). A character in the resolve flag field instructs the EDMGETV REXX to perform substitution.

Returns: OK: rc=0, eval-string= obj.var value

err: rc>0
(in OS/2 causes Error 40: Incorrect call to routine)

Examples:
```
call EDMGETV(ZMASTER,ZOS)
call EDMSETV(ZMASTER,ZOS,'NT',1)
```

EDMGET
(objname,heapnumber)

Function: get an object

heapnumber is an optional parameter

heapnumber defaults to 0 which means current heap

Returns: OK: rc=0, REXX Stem Variable named objname created that depicts the object

err: rc>0
(in OS/2 causes Error 40: Incorrect call to routine)

Examples:
```
call EDMGET("TESTREXX" , 1)
CALL EDMGET("TESTREXX" , 1)
rc = EDMGET("TESTREXX" , 1)
RC = EDMGET("TESTREXX" , 1)
```

- Single quotes are okay as well, e.g.,
  `CALL EDMGET 'ZMASTER'`
- The heap number is optional and quotes are okay instead of ()

EDMSETV
(objname,varname,value,heapnumber)

Function: To set a variable in an object.

heapnumber is an optional parameter.

heapnumber defaults to 0 which means current heap.

Returns: OK: rc=0

err: rc>0
(in OS/2 causes Error 40: Incorrect call to routine)

EDMSET
(objname,heapnumber)

Function: To set an object.

heapnumber is an optional parameter.

heapnumber defaults to 0 which means current heap.

Returns: OK: rc=0

err: rc>0
(in OS/2 causes Error 40: Incorrect call to routine.)

Examples:
```
call EDMSET("TESTREXX" , 1)
CALL EDMSET("TESTREXX" , 1)
rc = EDMSET("TESTREXX" , 1)
RC = EDMSET("TESTREXX" , 1)
```

- Single quotes are okay as well, e.g.,
  `CALL EDMSET 'ZMASTER'`

3

Notifying Clients

This chapter tells you about the different forms of invoking the Notify function, how to configure multiple Notify Managers and how Notify retries are established. New Notify features include scheduling and Wake-On-LAN.

An Overview of Notify

The Notify function enables you to initiate the execution of a program or programs on a Client desktop from another location. Usually, the Manager initiates the Notify process. However, a Client that is configured as an administrator can also initiate Notify processing requests.

In a typical manual connect scenario, the Client initiates the Client Connect process to receive the resources configured for that desktop. By using Notify, you can reach out to that Client and request that it connect, or accomplish some other task defined for that desktop, at any time.

Generally, this process depends on the Manager having a reliable means of identifying and contacting each Client. This means may be the LU name in an SNA environment, the IP address used at last Client-Manager connect in a static IP address environment, or the host name of the Client machine in a DHCP environment. Any of these three methods may be used to identify the Client as the target for a Manager initiated Notify. It is by this identifier, as well as the communications environment being used, that the Manager knows how to contact the Client desktop. Once communication between the Manager and the Client is established you can initiate processes to perform a variety of functions. The Client identifier must be unique and reliable in order to predict which Client will receive the Notify.

Notify can be used on all Novadigm-supported platforms. Unix, Windows NT, and OS/2 platforms have the ability to send and receive messages from remote terminals. Windows 3.1, Windows 95, and Macintosh platforms only have the ability to receive messages from remote terminals.

For TCP/IP, in order to be receptive to incoming Notify messages, receiving desktops must have a Notify receive daemon running in the background. The Unix, Windows NT, Windows 95, OS/2, and Macintosh platforms must run the Notify receive daemon and the EDMEXECD.EXE program in

order to receive any incoming messages. To receive messages in Windows 3.1, the EDMNTFYD Notify receive daemon needs to be running in the background.

Notify currently supports TCP/IP, LU6.2, and e-mail. The sender and receiver must be using the same communications protocol if the program is to execute properly.

The following section describes how the Manager can notify clients to initiate the Client Connect process.

Notify and the Client Connect Process

Software distribution is normally discussed in terms of *push* and *pull*. This refers to the concept of *pushing* software out from a central location to a Client, or the Client *pulling* software in from a central location. The major difference between the push and pull scenarios is who initiates the distribution activity — in Novadigm terminology — the Manager or the Client. Novadigm supports both models by offering numerous options that are used to define how, when, and where the distribution process is initiated. Notify provides users with the means to configure and implement a push scenario.

Novadigm's distribution process — the Client Connect process — is part of an entire configuration process during which the Client *connects* to the Manager to determine what configuration that Client should have. This connection process is comprised of a series of programs that execute on the Client to perform comprehensive, continuous configuration management. Many of these programs communicate with the Manager to obtain required information, while other programs perform strictly local processing.

Novadigm supports three basic connect types, as follows:

- **Manual Connect**
  The user invokes the Client Connect process by typing a command (EDM), by choosing an icon (EDM), or by

selecting an application from the Software Mall (RADIA). This process can be defined as a *pull.*

- **Timed Connect**
  A timer process that runs on the client executes the Client Connect process at a pre-determined date and time. Either the native timer services that reside on the Client or the EDM timer can be used to schedule this type of connect. This operation can also be defined as a pull.

Note: Notify can only be executed if the "Notifier" (usually the Manager) and the "Notifyee" support the same communications protocol.

- **Notify Connect**
  The Client is notified from a central control point — such as the Manager — to perform the connection. The Notify process is a *push.*

Notify usage varies from initiation of all connections via Notify, notifying only clients whose connect had errors, to ad-hoc notifies to re-boot a single machine. It is a powerful, flexible tool that can be used for starting processes on a Novadigm-managed desktop that are not Novadigm-specific in nature (e.g., reboot, backup, etc.).

When to Use Notify

As previously mentioned, Notify can be used to force a Client Connect. In this situation the Client is notified from a central control point — such as the Manager — to perform the Client Connect process.

Notify can be used for purposes other than initiating the Client Connect process. Specifically, Notify can be used in the following ways:

- **Using Notification for Other Purposes**
  Notify can be used for the emergency distribution of files outside of a Client Connect process, initiating some event on the Client (such as switching versions), or to collect debugging information about a connect failure.
- **Instead of EDMTIMER**
  EDMTIMER is used to deploy applications at specific time

intervals. It is often set to execute during non-peak hours. In large network environments, initiating a large number of connects at once can slow down deployment by overburdening the source. Notify can be used in place of EDMTIMER to force smaller groups of users to receive information at staggered time intervals.

Types of Notify Invocations

There are three ways of invoking the Notify function:

- Simple Notify
- GUI Configured Notify (Drag-and-Drop Notify)
- EDMMPUSH

Each type of Notify invocation is described below.

Simple Notify

The EDMMNFYT method (Simple Notify) was the first Notify function developed. Simple Notify enables you to notify Clients in a TCP/IP environment to initiate the connect process on a Client desktop from another location. To execute Notify successfully, EDMEXECD must be running on the Clients on which you are executing a push. The EDMMNFYT method stores the results of the notification in the Notify file, producing an instance for each Client notified. This Manager method (ZNFYT for pre-4.0 versions of the Manager) works on all Novadigm-supported Manager platforms. The following are the required parameters for EDMMNFYT:

Parameter Name	Description
domain name	The name of the domain where the notification results are stored. If this parameter is not specified, the domain name will be automatically generated as a function of date/time.
instance name	The instance name containing the results of the single notification. If this parameter is not specified, the instance name will be automatically generated as an eight-digit number (default is 00000001).
password	The ZNFYPWD for the target terminal's ZMASTER object.

Parameter Name	Description
Port	For MVS only. The port number. Should have the same value as the ZMASTER port number.
"process to run"	The application you are forcing the Client desktop to execute.
Target IP Address	The IP address of the Client desktop on which you are executing a push.
User ID	The Client user ID.

Note: You should specify both the domain name and instance name parameters. If these 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.

The EDMMNFY6 method enables you to notify Clients in an SNA environment. The EDMMNFY6 method stores the results of the notification in the NOTIFY file, producing an instance for each client notified. The following are the required parameters for EDMMNFY6:

Parameter Name	Description
domain name	The name of the domain where the notification results are stored. If this parameter is not specified, the domain name will be automatically generated as a function of date/time.
instance name	The instance name containing the results of the single notification. If this parameter is not specified, the instance name will be automatically generated as an eight-digit number (default is 00000001).
LOGMODE	
LUNAME	
TPNAME	The LU6.2 default TPNAME.

Note: Newer Notify features, such as Wake-On-LAN and Scheduling for Notifies are not supported by Simple Notify. To take advantage of these features, you must use the EDMMPUSH form of Notify.

GUI Configured Notify

Version 4.0 of the Manager supports a method for establishing notifies using a standard graphical user's interface. This easy to use process is called *drag-and-drop* notification (DDN). The System Explorer provides the support for drag-and-drop notification on the Administrator side. The drag-and-drop notification feature was developed to make it easier to notify large groups of users. Presently, this feature allows for the notification of all the users of the department, user group, single user, or all the users of the service. It is very important to understand that the destination information is searched for in the PROFILE file, specifically, in PROFILE.<USERID>.ZMASTER.OBJECT.

Note: In order for the DDN to work, all the necessary information should be written to the PROFILE during the Client resolution process before the DDN. If the necessary information is not in the PROFILE file, DDN will not be possible. DDN is not designed to notify new users whose destination information is not in the PROFILE database yet.

There are two aspects to DDN, the source icon (dropper) and the destination icon (dropee). The source icon is what is dragged and dropped on the destination icon. The source may be an instance from the USER class, or an instance from the WORKGRP class, or an instance from the DEPT class. All of these instance types usually are found in the PRIMARY.SYSTEMX domain. The other type of the source may be an instance of the SERVICE class. The destination icon must be a ZCOMMAND class instance, usually found in PRIMARY.ZSYSTEM domain of the EDM Database.

Note: The source instance must belong to one of the following classes: USER, WORKGRP, DEPT, or ZSERVICE. The destination instance must be a member of the ZCOMMAND class.

➢ To perform drag-and-drop notification

1. Open the database tree all the way down to the specific icon that represents the source instance (for example, PRIMARY.SYSTEMX.WORKGRP.PROD_USERS).

2. Highlight the instance with a right-mouse click, and while still holding the right mouse button down, drag the instance until it is on top of the specific icon that

represents the destination instance (for example, ZSYSTEM.ZCOMMAND.NOTIFY). When the source icon is on top of the destination icon, an icon resembling a magic wand will appear to let you know that you are in the command class.

3 Release the source icon onto the destination icon.

The result of the above example is that all instances belonging to WORKGRP.PROD_USERS will be notified.

The response message from the Manager will tell you how many users were found in this particular group and how many were scheduled for notification. Again, if the information about the selected users is not in the PROFILE file, no notification will occur. As usual, all the results of the notification may be found in the Manager log. Additionally, the notification results are going to be written to the NOTIFY file with a domain name created dynamically for each DDN action. The name of the domain will be returned to the Client in the ZADMNHNL attribute of the ZADMIN object. To see the results, point to the domain with a right-mouse click, and then select **Status Display**. Another option, **Status Delete,** will delete the domain when you do not need it.

Types of Notifications Supported

It is important to note that a command class instance is a special type of the instance used to define the command to be executed. The following table lists the ZCOMMAND class variables used for DDN:

ZCOMMAND Class Variables Used for DDN

Variable Name	Description	Length
ZCMDMODE	Log mode (LU6.2 only)	8
ZCMDNAME	Command name (NOTIFY, NOTIFY6, EMAIL)	8
ZCMDPATH	Location of the command. Used only for EXEs that are not pre-established (NOTIFY, NOTIFY6, EMAIL)	255
ZCMDPRMS	Parameters passed to the command	255
ZCMDSEP	Separator used for parameters in user defined commands	1
ZCMDSYNC	Synchronization flag which defines whether to wait until the user command executes and ends or return control immediately (Y/N)	1
ZCMDUCLS	User class name. This is the name of the class to look for users connected to the droppee. For example, if the value is set to COMPUTERS and the droppee is WORKGROP.ACCOUNTING, instances of COMPUTER class that are members of WORKGROP.ACCOUNTING will be the selected audience for the notification. If ZCMDUCLS is not specified (using the above example), then the audience will be created by instances of COMPUTER class that are members of WORKGROP.ACCOUNTING . The default for ZCMDUCLS is USER.	8
ZCMDTPN	TPN (LU6.2 only)	8

Variable Name	Description	Length
ZCMDTYPE	) be executed (REXX,	8
ZCMDHNDL	main name Notify file to store the tification results. The reated depending on the nd instance name is ntial number (for first 1, for next 00000002 not specified the name a function of date/time to	32
ZCMDUINF	Notify start and end cified then ZCMDHNDL instance name (heap d. dle was specified as RED_NOTIFY for the first request user info will be RED_NOTIFY_00000001 RED_NOTIFY_00000002	128
ZCMDRMAX	ies in case of notify failure.	3
ZCMDDLAY	Delay interval in seconds before retry will be scheduled in a case of failure. The default is 300.	4
ZCMDNFYD	Date when the notify request should be executed first time. The format is YYYY/MM/DD. The default is the current date.	10

Note: The user info in ZCMDUINF can be used as a key to write the info into a SQL database. The key would be unique. It is recommended that you not specify the value and let it be generated as described above. However if the value is used in some other fashion in notify methods it can be defined and then further processed in any way in notify methods (for instance may combine the handle and IP address for TCP/IP).

Note: You can also use the ZNFYTSTA REXX in conjunction with all types of notifies. See *Chapter 2:, Managing Processing* for more information.

Variable Name	Description	Length
ZCMDNFYT	Time when the notify request should be executed first time. The format is HH:MM:SS. The default is the current time.	8

Currently, three types of Notify operations are supported in the command class. The ZCMDNAME variable may have the value NOTIFY, NOTIFY6, or EMAIL. This means that you may use TCP/IP, LU6.2, or e-mail for the notification. The notification command, specified as the ZCMDPRMS variable, is the text sent to the Client either as the command line in a TCP/IP or LU6.2 Notify or as a message and a subject in e-mail notify. (See *Scheduling for Notify* on page 154 for information on configuring the command class for deferred notification.)

Note: The source instance must belong to one of the following classes: USER, WORKGRP, DEPT, or ZSERVICE. The destination instance must be a member of the ZCOMMAND class.

It is also important to understand that before the command class instance is taken for the processing, the secondary resolution is done and all the variables of the instance will be substituted. This will allow to partially or completely change the command line and/or even notification type itself.

Necessary Profile Information

In order for the DDN to work the following information must be present in the source instance's PROFILE.userid.ZMASTER.OBJECT:

Variable Name	Description	e-mail	TCP/IP	LU6.2
ZUSERID	User ID of the target must match the one in the ZMASTER object at the time of notification.		X	

Variable Name	Description	e-mail	TCP/IP	LU6.2
ZNFYPWD	Password. If there is ZNFYPWD in the ZMASTER object of the PROLIFE file, Notify will use it, otherwise the default "EDMPASS" will be used.		X	
ZNTFPORT	Port number for Notify demon (default 512)		X	
ZCIPADDR	IP address of the Client machine.		X	
ZIPNAME	Fully qualified host name of the Client machine. If ZCIPADDR is not specified this variable may be used instead.		X	
ZLUNAME	The IP address in the format XXX.XXX.XXX.XXX. If ZCIPADDR is not specified this variable may be used instead.		X	X
EMAIL	Fully qualified e-mail address of the destination.	X		

There are three ways of specifying a Client IP address:

- **ZCIPADDR**
  This variable may be used to point to a specific host name/address and is not resolved by the Manager.
- **ZIPNAME**
  This variable contains the fully qualified symbolic host name of the connecting Client and is resolved by the Manager.
- **ZLUNAME**
  This variable contains the IP address and is resolved by the Manager.

The order of precedence for the sources of Client IP addresses is ZCIPADDR first, then ZIPADDR and ZLUNAME.

In the case when the dropper is an instance of the SERVICE class, the service instance should be generated for each user of the service and this instance should be written to ZSVCSTAT class or ZERVICE class. The ZSRCDOMN and ZSRCCLAS variables of this instance should specify the names of the domain and class of the service. The instance name has to match the name of the service itself. This information is used in the creation of the audience list as described in the previous paragraph.

Programmatically Configuring Notifies

Drag-and-drop was initially designed and implemented to support the System Explorer. However it may be used separately in a well-established infrastructure for mass notification controlled by a timer on the client side. The only input that is necessary for the back-end is a ZADMIN object, that has all the variables in it defining the source instance and the destination instance.

The following table describes the variable names and gives examples of the values.

Variables of the ZADMIN Object

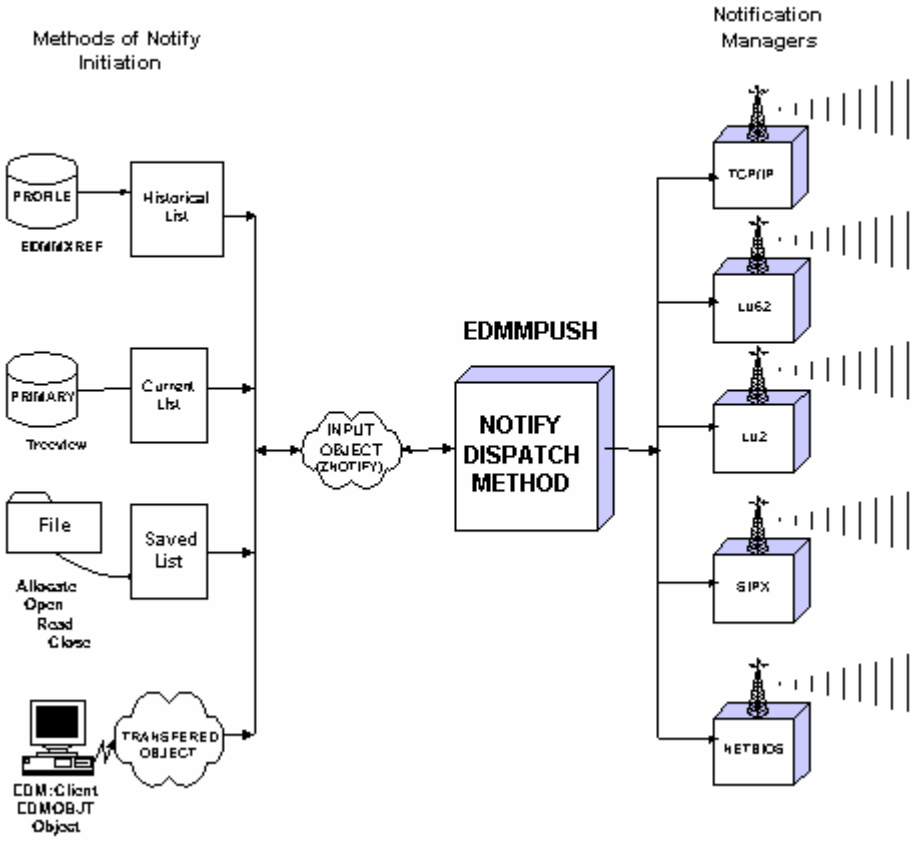
Variable Name	Description	Sample value
ZADMFILE	Destination file name	PRIMARY
ZADMDOMN	Destination domain name	ZSYSTEM
ZADMCLAS	Destination class name	ZCOMMAND
ZADMINST	Destination instance name	NOTIFY
ZADMDFIL	Source file name	PRIMARY
ZADMDDOM	Source domain name	SYSTEMX
ZADMDCLS	Source class name	WORKGRP
ZADMDINT	Source instance name	PROD_USERS
ZADMFUNC	Function name for DDN	EXECUTE
ZUSERID	Administrator user ID	Vladimir
ZNFYPWD	Password	Edm123

As a result of this operation all the users of the PRIMARY.SYSTEMX.WORKGRP.PROD_USERS will be notified with the command defined in PRIMARY.ZSYSTEM.ZCOMMAND.NOTIFY. All the source information will be retrieved from the PROFILE file database.

EDMMPUSH

The EDMMPUSH method may be used to initialize notification of Clients and is an additional implementation of existing notification procedures of Manager. It is not going to substitute any of the existing Notify methods, but rather enhance the process. Please note however, newer Notify features may not be supported by types of Notify other than EDMMPUSH.

The major advantage of the EDMMPUSH method is its independence of communication protocols. The method itself does not do notification, it just receives input requests, gets required parameters and then puts requests to the right queues that later will be processed by a specific Notify Manager. This way the process of configuring the Manager Database is simplified, because all the types of notify requests may be now concentrated in a single object and send to the same method regardless of the notification type. The in-bound object or even dynamic object created as a result of the object resolution may be used to deliver requests to the EDMMPUSH. This object may require different types of notification for each heap (request). The administrator may create a single multi-heap object on the Administrator desktop to notify all the Clients regardless of notification type and then send the object to the Manager to accomplish the notification.



The above figure shows different ways to create the input object for the EDMMPUSH. On the left side there are various methods of building the list of Clients to be notified. All these methods write into input objects that later will be processed by EDMMPUSH as a result of object resolution. Each of these methods may put various types of request. For instance, the CURRENT LIST will use TCP notification for Clients that use TCP/IP type of connect and LU62 type for those using APPC communications. Of course, if the destination client is configured properly, notification may be sent using one protocol while the initiated connect may utilize another protocol (e.g.,

Note: The LU6.2 protocol is not supported by the EDMMPUSH method at this time.

LU62 Notify initiates LU62 Connect). All these various Notify requests are going to be processed by a single method that will route them into right queues for processing (to simplify the picture no queues are shown between EDMMPUSH and notification managers).

Input Object Used by EDMMPUSH

The EDMMPUSH method receives all the information about the Notify requests from the input object. The name of the input object is defined in the Manager Database, specifically in the PRIMARY.ZSYSTEM.ZMETHOD.EDMMPUSH instance field named "Parameters passed to Method". If this field does not specify an input object name, then a default of ZNOTIFY will be used by EDMMPUSH. Each communication protocol used to execute notification requires a specific set of variables as input. However, there are control variables that must be specified for every heap of the input object.

The following sections first discuss these common control variables and then describe specific protocol-dependent variables for each protocol.

Common Control Variables

Common Input Variables

You must specify the following input variables:

EDMMPUSH Input Variables

Variable	Description
NFYDELAY	Specifies the interval for delay before trying to re-Notify a Client. If no value is entered, the default value is the value specified in the NFYT_TIMEOUT or NFY6_TIMEOUT setting of the MGR_NOTIFY section of Manager Settings file.
NFYHNDL	Specifies 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	Specifies the maximum number of retries. If no value is entered, the default value is the value specified in the NFY_RETRY setting of the MGR_NOTIFY section of Manager Settings.
NTFYRTIM	Novadigm timestamp defining the time after which the notification should occur.
NFYPROC	Controls processing of the current heap request. If value is **Y** (yes), then the heap will be processed. If the value is **N** (no), then the request for the current heap will be ignored. The default value for this variable is **Y**.

Variable	Description
NFYTYPE	Defines the type of the Notify requested. The following values are allowed: • TCP • SIPX • NETB • LU62 • EMAIL Just first three bytes of the type are used for the identification, so SIPX and SIP will be treated as the same. There is no default value for this variable. If this variable is not defined, the current heap of the object will be ignored.
NFYUINFO	Allows you to enter user information.

COMMON VARIABLES SET BY EDMMPUSH

As a result of the input request processing following variables are set in the input object:

Variables Set by EDMMPUSH

Variable	Description or Setting
ZMMSG	Message about success or failure of required notification scheduling.
ZMRC	Return code (0 – success, 4 – warning, 16 – failure).
ZOBJCDEL	Set to **Y** in order to enforce control object deletion after object transfer is done.
ZOBJRDEL	Set to **Y** in order to enforce response object deletion after object transfer is done.

Protocol Dependent Input Variables

SIPX

The SIPX Notify uses REXEC protocol to deliver notification to the Client side. That is why the User ID and password are required for this type of notification to work. After the connection is established and the User ID/Password combination is verified successfully, the command line sent with the request will be executed on the remote (destination) machine. This command line should initiate the Client Connect process. It is the administrator's responsibility to make sure that the command line contains the call that may be executed on the Client side and will initiate the Client Connect with the Manager.

Variable	Description
NFYCMD	Command line to be executed on the destination machine. This command line should initiate EDM Client Connect on the Client side.
NFYNETW@	Address of the destination Client.
NFYNODE@	Node address of the destination Client.
NFYPASSW	Password acceptable with user id specified in NFYUSER.
NFYSOCKT	Listening socket number on the destination machine.
NFYUSER	User ID acceptable on the destination system (used by native operating system security system).

TCP

The TCP/IP Notify uses REXEC protocol to deliver notification to the Client side. That is why the User ID and Password are required for this type of notification to work.

After connection is established and User ID/Password combination is verified successfully, the command line sent with the request will be executed on the remote (destination) machine. This command line should initiate the Client Connect process. This is the administrator's responsibility to make sure that the command line contains the call that may be executed on the client side and will initiate the client connect with the Manager.

Variable	Description
NFYCMD	Command line to be executed on the destination machine. This command line should initiate client connect on the Client side.
NFYIPADR	TCP/IP address of the destination Client.
NFYIPORT	Listening port number on the destination machine.
NFYPASSW	Password acceptable with User ID specified in NFYUSER.
NFYUSER	User ID acceptable on the destination system (used by native operating system security system).
NFYMAC	Mac address of the destination machine will be used for “Wake On LAN”. If NFYMAC is not specified “Wake On LAN”

EMAIL

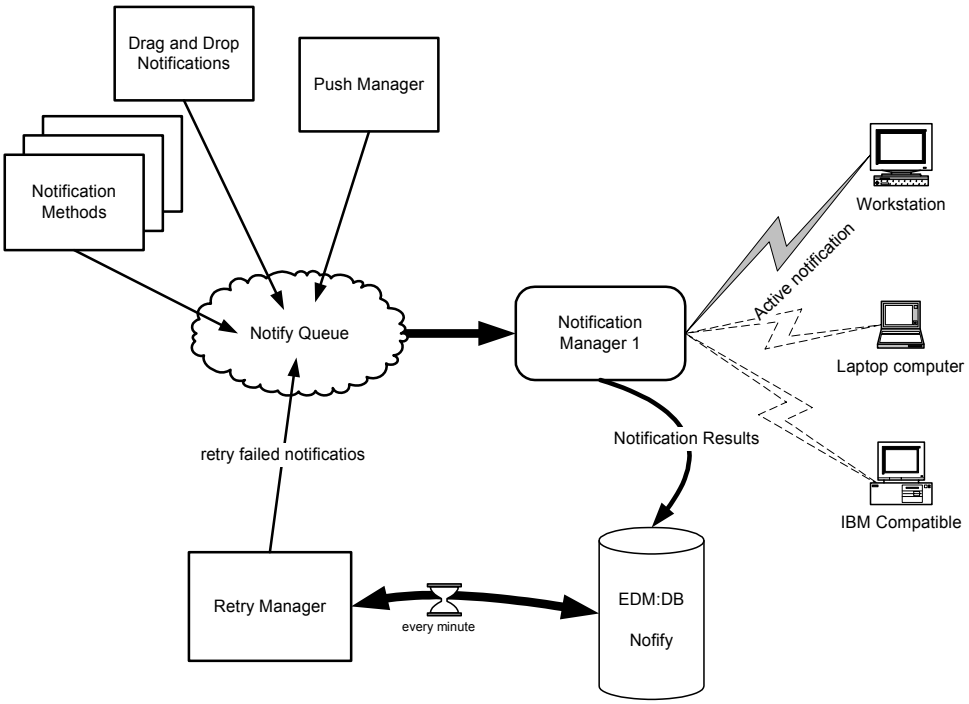
Variable	Description
EMAILATT	Attachment to send with e-mail (optional), user can specify multiple attachments by putting semi-colons (;) between files to separate them.
EMAILFRM	E-mail address of a person who is sending it. (mandatory)
EMAILMFN	Message file name, in case message is greater than 255 characters (mandatory, if EMAILMSG is not used).

Variable	Description
EMAILMSG	Message, which is restricted to 255 characters (mandatory, if EMAILMFN is not used). To send a message with spaces, follow the directions as in EMAILATT.
EMAILSUB	Subject of e-mail (optional). If the user wants to use space in the message, the subject must be enclosed in quotes, e.g., "Hello Test" to send e-mail with subject Hello Test. If the user does not put a quotation mark at the end of the message, then text up to first space will be sent as a subject; "Hello" in this case.
EMAILTO	E-mail address of a person to send e-mail to. (mandatory)

Configuring Multiple Notify Managers

You may now configure multiple Notify Managers (of the same type of communications) in order to accelerate processing of a large number of notification requests. When multiple Managers are used, a single notification pipeline is substituted by as many lines as there are Managers started for the notification type. This approach is based on one request queue and multiple number of Managers reading from this queue.

The queue is resource protected by a read mutex semaphore. Each Notify Manager waits on the semaphore. The Manager that owns the semaphore reads from the queue and then immediately releases the semaphore, so that the other managers may start processing queued requests. At that point, the Manager continues the notification process for the request it currently has.



Start Up Process for Multiple Manager Implementation

All the Notification Managers should be started the same way a single Notify Manager was started in prior versions. In other words, you should specify as many Managers in the ATTACH_LIST section of the Manager Setting file as you require. Additional parameters in the command line of the MGR_ATTACH_LIST section of the Manager PROFILE file should be used to uniquely identify each of the starting Notify Managers and would look as follows:

```
CMD=(znfytmgr NAME=ZNFYTM001) …
CMD=(znfytmgr NAME=ZNFYTM002) …
CMD=(znfytmgr NAME=ZNFYTM003) …
```

This will cause the task manager to start and maintain three EDM tasks with names:

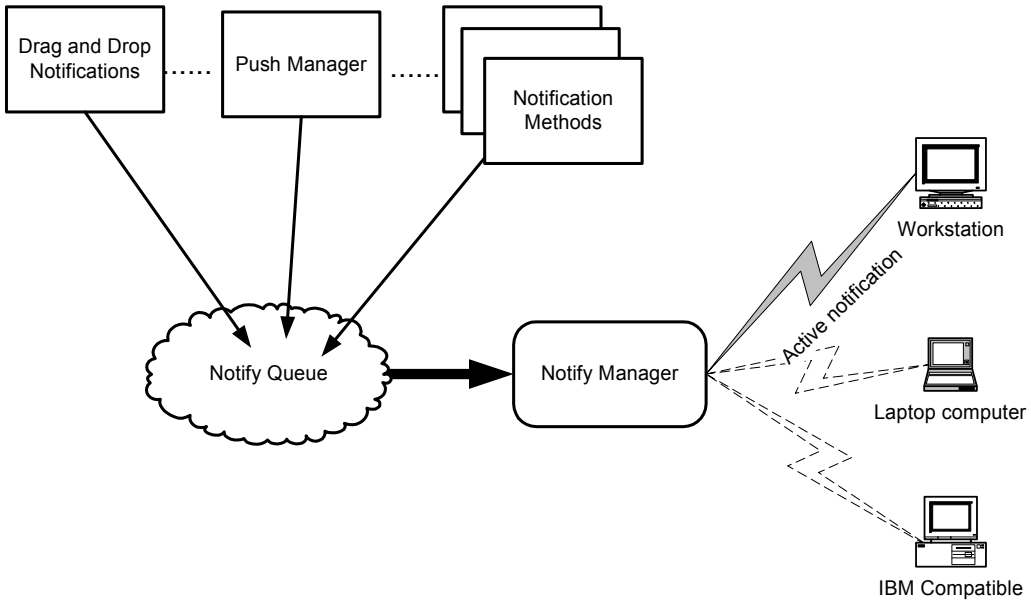
```
ZNFYTM001
ZNFYTM002
ZNFYTM003.
```

Note that the above example is for a TCP/IP Notify Manager. To configure multiple Notify Managers for LU6.2, use the following example:

```
CMD=(znfy6mgr NAME=ZNFY6M001) …
CMD=(znfy6mgr NAME=ZNFY6M002) …
CMD=(znfy6mgr NAME=ZNFY6M003) …
```

Notify Retry Queue

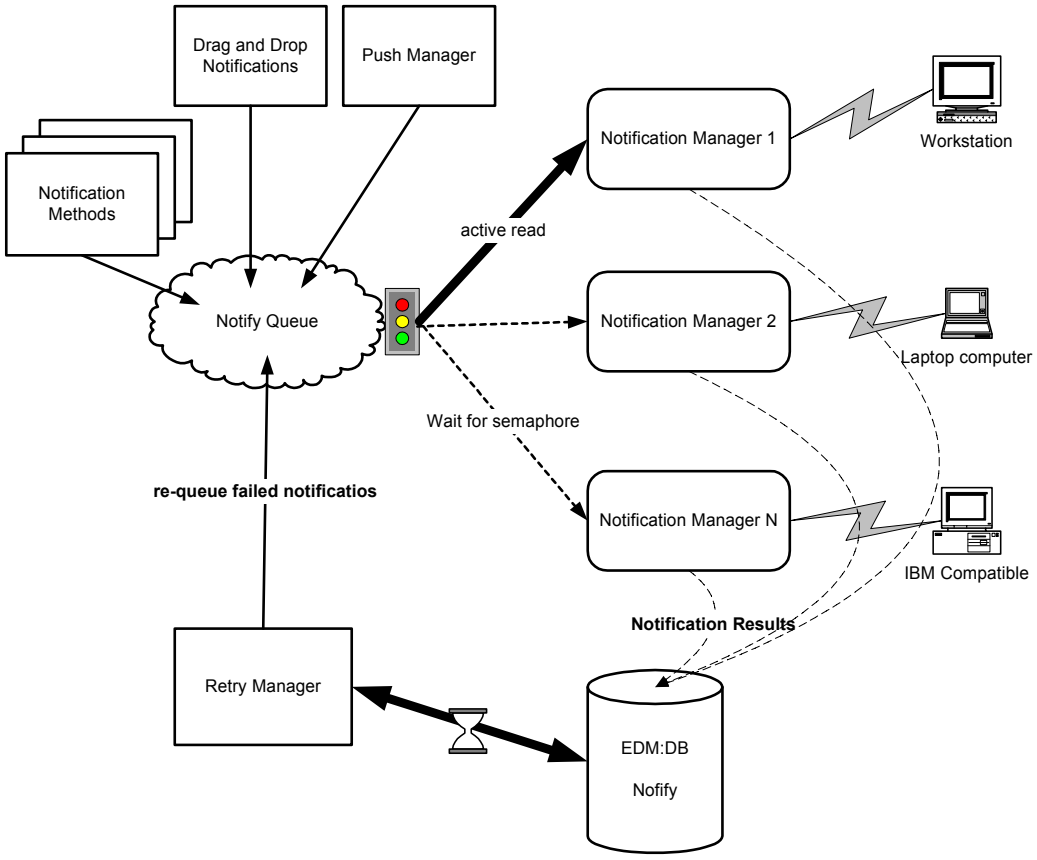
Prior to the release of the version 4.1 Manager, the notification process included the three ways of invoking notification described above. These types of notification put notification requests into a Notify queue. The Notify Manager task would take one request from the queue, process it, and write the results of the processing into the Manager Database NOTIFY file. Then, the Notify Manager would process the next request, and the next, until all requests were processed.



This processing was completely sequential. If, for some reason, one of the notifications would take an inordinate amount of time, all queued requests would be delayed. Also the notification Manager does not retry the notification in case of error.

To allow for retry notifications the Notify Manager now stores failed request information in the RETRY domain of the NOTIFY file. The suitable time for retry is set in the request. The Retry

Manager wakes up every minute and checks all instances of all classes in the RETRY domain. If failed requests exist, the Retry Manager compares the scheduled re-notification time with the current time and re-queues the request if the time is right. The Retry Manager processes failed notifications for all types of communications Managers and re-queues them in the right queue depending on the type. To configure for the Retry Manager, you must add zrtrymgr to the MGR_ATTACH_LIST setting of the Manager Settings file.



Scheduling for Notify

With the version 4.3 release of the Manager, you can use the EDMMPUSH method and the Retry Manager in combination to schedule delayed executions of Notify. To configure this scheduling feature, you must add a NTFYRTIM variable to the inbound EDMMPUSH object and you must have the zrtrymgr task included in the MGR_ATTACH_LIST setting of your Manager Settings file.

Variable	Description
NTFYRTIM	Time in the format of EDM_TIMESTAMP to specify when the transaction should be scheduled. If this variable is absent or set to all blanks, EDMMPUSH will presume that the request should be executed immediately. The format of the time stamp is: **Parameter** **Length** **Options** year 4 month 2 1-12, where 1 = January weekday 1 0-6, where 0 = Sunday day 2 1-31

Variable	Description
	hour 2 0-23 minute 2 0-59 second 2 0-59 millisecond 3 0-999 time zone 4 difference in minutes between GMT and local time Example: 199907509143500000+300 = 1999 07 5 09 14 35 00 000 +300 or: 07/09/1999 14:35:00:000 in NY USA.

If NTFYRTIM is specified, EDMMPUSH does not put the request in the Notify queue. Instead, the request is written to the RETRY domain of NOTIFY file. The zrtrymgr checks the RETRY domain every minute and at the date and time specified by NTFYRTIM, the zrtrymgr schedules the Notify for the appropriate Manager. The scheduling function allows you to retry failed notification. You can also recover those notifications that were scheduled, but where the Manager was stopped and re-started.

Scheduling for Notify can also be configured for Drag-and-Drop Notify. To enable the scheduling function, you must add the following variables to the ZCOMMAND class:

ZCOMMAND Variables Needed for Scheduling

Variable Name	Description	Length
ZCMDHNDL	Domain name created/reused in the Notify file to store the information about notification results. The class name will be created depending on the type of notification. Individual instance names are generated as sequential numbers (e.g., 000000001 for the first request, 00000000s for the second, etc.) If ZCMNDHNDL is not specified, it will be uniquely generated as a function of date and time	32
ZCMDUINF	User information passed to Notify start and stop methods. If ZCMDUINF is not specified it will be uniquely generated as a combination of ZCMNDHNDL and instance name (heap number).	128
ZCMDRMAX	Maximum number of retries in case of Notify failure. The default is 7.	3
ZCMDDLAY	Delay interval (in seconds) before the retry will be scheduled. The default is 300.	4
ZCMDNFYD	Date when the Notify request should be executed for the first time. The format is YYYY/MM/DD. The default is the current date.	10
ZCMDNFYT	Time when the Notify request should be executed for the first time. The format is HH:MM:SS. The default is the current time.	8

Wake-On-LAN

Note: There are Wake-On-LAN configuration issues external to the Manager. The sidebar provides an example in the CISCO environment.

Another version 4.3 Notify feature that takes advantage of the Retry Manager is Wake-On-LAN. If a TCP Notify fails because the destination machine is powered down, the znfytmgr will issue a Wake-On-LAN to wake up the machine. The znfytmgr will then retry the Notify. Please note that zrtrymgr must be active to re-schedule the request. The Wake-On-LAN is issued only once per notify request. The following messages will appear in a manager log:

EDM1977I 17:57:03 [NFYTMGR2 /17B] System Task
Issuing WAKE_ON_LAN call for IP@ = <208.244.225.87>, MAC_addr = <0050DA12437F>.

EDM1987I 17:57:03 [NFYTMGR2 /17B] System Task
EDMWAKE is issued to power up the destination machine, will retry notify in <300> seconds.

The last message will also appear in the corresponding instance of NOTIFY file for the time of delay.

You must add the following entry to the MGR_NOTIFY section of the Manager Settings file to enable or disable support for Wake-On-LAN:

```
ISSUE_WAKE_ON_LAN = yes
```

The default is `NO`.

4

Logs and Messages

This chapter explains the naming conventions and formats used in the activity logs generated by the Manager logging facility, and how trace settings in the file affect logging activity. Also, this chapter explains how to interpret and understand the Manager log and its contents, including the specific messages and their meanings.

This chapter contains commonly seen messages.

Overview of Manager Logging

Note: The Manager Settings file accepts both of the following spellings of threshold: "threshold" and "threshhold".

The Manager maintains an activity log that you can use for informational purposes and for problem determination. These logs are generated for Manager activity.

The Manager log reports a wide range of program activity. This includes everything from Manager start-up to the logging on of each Client, and the establishment of System Explorer sessions. As with the Client logs, you can customize the tracing levels of diagnostic logging for the Manager.

The volume and detail level of the messages that your Manager produces depends on two factors: trace settings and the logging options specified for your specific Manager.

All Manager messages are marked by an alphanumeric character string that begins with the prefix "EDM" or "RAD", followed by the message number (xxxx).

Manager Log Location and Settings

During the installation process, you allocate space for the Manager log. After the installation process, you must specify the location of the log file in the MGR_LOG section of the Manager Settings file. The MGR_LOG section contains six settings that determine where the log is located and how *elastic* it is.

- The **Directory** setting
  tells you the drive and directory to which the log will be written.
- The **FLUSH_SIZE** setting
  determines the buffer size (in bytes) used by the Manager to store messages before actually writing them to the log.
- The **THRESHOLD** setting
  displays how many lines the log file will contain before it is dumped to another file. When the log is dumped, it will be

given a temporary file name in the format of EDMMR*xxx*. A negative threshold value will reuse the same file, overwriting the previous log segment.

- The **PIPESIZE** setting
  determines how many messages (in bytes) the Manager collects before processing and storing them in the log buffers.
- The **MESSAGE_WIDTH** setting
  displays the length (in characters) of each message line.
- The **MESSAGE_PREFIX** setting
  determines whether the messages begin with EDM or RAD.

As with most Novadigm parameters, the Manager log settings can be tuned for maximum efficiency after monitoring your system's performance.

The log will accumulate messages to fill up the allocated space. When this threshold is reached, the Manager will create a new log, designated by a different log ID, or write over the previous log. If this occurs, you can change the trace settings to capture fewer messages, or increase the allocated size of the log. Also, you can change the MGR_LOG settings, if warranted, to maximize memory usage and for other performance considerations. When a new log file is created, it is identified by an EDMMR prefix, and a three-digit extension.

MVS Note: When you installed the MVS Manager, you set up logging data sets to specify where the messages will be logged. The EDM start-up Job Control Language (JCL) requires the presence of two logging data definition names (DDNAMES), EDMLOGA and EDMLOGB. The MGR_LOG section of the Manager Settings file includes a setting that enables you to specify the number of lines EDM processing will write to EDMLOGA before switching to EDMLOGB (and back to EDMLOGA).

You can switch from EDMLOGA to EDMLOGB (or EDMLOGB to EDMLOGA) by using the following MVS Console command:

```
F  jobname,SWITCH
```

or just

```
F  jobname,SWI
```

Here the minimum required is three characters, even though ywo would uniquely identify the command (distinguish it from shutdown).

To flush the log, using the following command:

```
F  jobname,FLUSH
```

or just

```
F  jobname,FLU
```

For details about specifying the MGR_LOG settings, see *Chapter 1: Manager Settings*.

The number of trace settings you specify will influence the number of log messages that are written to the Manager.

The TRACE section of the PROFILE file controls and influences the diagnostic logging for the Manager. This section contains keywords that represent trace settings. All diagnostic output produced by TRACE keywords is written to the active Manager log. To select a trace setting, specify **Yes**. To deselect a trace setting, specify **No**.

The TRACE keywords you specify in the TRACE section are invoked at Manager initialization and remain in effect until you change them by modifying the TRACE section. The trace settings in effect at Manager initialization are displayed at the beginning of the Manager log.

MVS Note: You can refresh your trace options with the following MVS Console command:

```
F  jobname,REFRESH,memname
```

or just

```
F  jobanme,R,memname
```

where *memname* is a member in the parmlib the Manager was started with.

You can refresh your trace options with the following MVS Console command:

```
F  jobname,REFRESH,memname
```

or just

```
F  jobanme,R,memname
```

where *memname* is a member in the parmlib the Manager was started with.

For details about specifying the TRACE keywords, see *Chapter 1: Manager Settings*.

There are other Manager Settings sections that impact the Manager Log. These include the SHOW_VERINFO setting of MGR_STARTUP section. This setting allows you to display version information for each Manager module at the beginning of the Manager Log. MGR_MESSAGE_CONTROL allows you to control the production and destination of Manager Messages.

MVS Note: You can change your version display options with the following MVS Console command:

```
F  jobname,VER
```

This command is useful if you have set SHOW_VERINFO=NO in the MGR_STARTUP section and you need to send a log to Novadigm Technical Support.

You can also refresh your MGR_MESSAGE_CONTROL settings with the following command:

```
F  jobname,R,memname
```

where *memname* is a member in the parmlib with which the Manager was started.

For details about Manager Settings, see *Chapter 1: Manager Settings*.

Viewing the Manager Log

The Manager log file reports on any program activity. You can view the Manager log using your preferred editor. (Two platforms — Windows NT and OS/2 — have GUI interfaces [program groups and icons] for viewing the Manager log.) The Manager log is located in the directory that you specified during installation.

Reading the Manager Log

The Manager logging facility generates an activity log with multiple messages (or line entries). To help you read these entries and understand the information, you should be familiar with the format used. A typical line entry is displayed in the following sample from a Radia Manager for Windows NT log.

Example

```
EDM0999I 13:48:22 [NFYTMGR1 /144]   System Task
    --- TCP Notify Manager <NFYTMGR1> started
```

This message consists of a single line entry with six log fields, as follows:

- The **Message Number** (EDMnnnn or RADnnnn)
  denotes the EDM or Radia Message number. Note that the message number ends with a letter to mark the message as an information (I), error (E), or warning (W) message. You can adjust the prefix to reflect the appropriate type of Manager by changing the MESSAGE_PREFIX setting in the MGR_LOG section of the Manager Profile. See *Chapter 1: Manaeger Settings* for more details.
- The **Time Stamp** (hh:mm:ss)
  indicates the time that the event occurred.
- The [**TaskID** (xxxxxxxx) or communications address (nnn.nnn.nnn.nnn) / Process ID]
  indicates the task for which the activity is being logged, or the communications address currently being used, followed by the process ID. This can be a Manager system task, such as zutilmgr or the communications address of the Manager—204.7.83.74. The process ID represents the Novadigm process number.
- The **Task Type**
  indicates the type of task currently underway. Task types include "System Tasks", "Initializing", Client Type and

version number (e.g., EDMV4 Client"), Admin Task or DMA Task.

- The **Message Type** (---)
  is another way of indicating the type of message being logged. Message types can be informational, (---), warning (--?), or error messages (--!). This should help you to analyze Manager activity quickly and effectively.
- The **Message**
  is the actual message text.

Note: Typically, activity logs contain multiple messages that represent an activity. Most activities that require multiple messages will be framed with an *activity begins* and an *activity ends* message.

You can change the message format to insert a Julian date (YYYYDDD) by adding the MESSAGE_DATE setting to the MGR_LOG section of the Manager Settings file. The value for the MESSAGE_DATE setting is JULIAN.

Example

```
EDM0604I 1999335 [ztoptask /24330]   System Task       ---
   MESSAGE_DATE     ==> JULIAN
```

A Manager Sample Activity Log

A properly sized and configured log will give you appropriate detail on the entire scope of Manager operations. While individual events vary from installation to installation, the general flow of the Manager log consists of five phases:

- Manager start-up
- Client Connect
- Object resolution
- End of Client Connect
- Manager shut-down

The following sections describe the types of messages that result from each phase.

Manager Start-up

Manager start-up is reflected in a series of messages at the beginning of the log, starting with the actual opening of the log itself. Next, the Manager will process the parameters in the PROFILE file and start various Manager tasks. At this point, the Manager is fully configured and operational, waiting for Clients (EDM or Radia) to connect in order to continue processing. Messages associated with this phase or Manager tasks starting, Manager tasks attaching, and Manager Settings section names and values.

Client Connect

The Client Connect process occurs when a client requests a session with the Manager to be resolved to its desired state. Each Client Connect is itself a Manager task. The Manager may defer the session to a time when it is not occupied, or will connect to the Client at the time of the request. Messages associated with client connect can be:

- "Client task has started"

  or

- "Connected to client."

Object Resolution

Object resolution follows a successful Client Connect. The Manager processes client objects, performs methods resolved from those objects and creates resulting objects, transfers data to and from the client, and creates objects containing audit and profile information for that client. Then, when the client has been returned to its desired state, the Manager releases the client and waits for another connect request. Messages associated with the object resolution phase include:

- "Object resolution (object name) begins/ends"

- "Current path (pointing to the methods directory or resource file)"
- warnings that objects could not be found or duplicate objects exist, method identification
- return codes for methods.

End of Client Connect

The end of the Client Connect process occurs when the Client requests that its session with the Manager be ended as a normal logoff. The message associated with end of Client Connect is:

- "Client task has ended"

Manager Shutdown

Manager shutdown occurs when a request to shut down the Manager has been received. Messages associated with this phase include:

- "Shutting down Manager"
- messages halting communications
- error messages that result in the Manager ending operations.

Manager Messages

Note: The following messages display the EDM prefix. The corresponding Radia messages which begin with RAD (not EDM) contain the same meanings.

The following section contains the Manager messages that are generated by the Manager logging facility. The message descriptions explain the meaning of the message, and any system actions that may occur. The descriptions also suggest an action or actions (if any) that can be taken by the system administrator to diagnose and correct a problem.

Message: EDM0000I hh:mm:ss [taskid/processid]
REFRESHING THE TIME

This message is produced every minute that the Manager is operating.

Message: EDM0001I hh:mm:ss [taskid/processid]
managertaskname HAS STARTED

This message tells you that the Manager task displayed in the message has started within the Manager.

Each task provides a specific set of services. The following table lists and describes the possible values of *managertaskname*.

The Manager Task List

EDM Manager Task	Description
AGENT MANAGER TASK	Reserved for future use.
CLIENT TASK	Actually initiates the Client Connect.
CLOCK MANAGER TASK	Wakes up every minute to update the time/date and flush the log.
LOG MANAGER TASK	Flushes the log messages periodically.
REXX MANAGER TASK	Runs REXX programs. An active task that allows you to perform functions that persist in the environment (as opposed to transient methods, which do not persist).
TASK MANAGER TASK	Responsible for timing out tasks and restarting system tasks.
TCP/IP MANAGER EMULATOR TASK	Allows connections from TCP/IP EDM Clients.
TCP/IP NOTIFY MANAGER TASK	Notifies TCP/IP Clients to connect.

Message: EDM0002I hh:mm:ss [taskid/processid]
managertaskname HAS ENDED

This message tells you that the Manager task displayed in the message has ended within the Manager.

Each task provides a specific set of services. See *The Manager Task List* table above for the possible values of *managertaskname*.

Message: EDM0003I hh:mm:ss [taskid/processid]
ATTACHING *managertaskname* [*taskdescription*]

This message tells you that the Manager task displayed in the message is being attached.

This message is written to the message log after the clock Manager task is started, and before the task actually starts.

Message: EDM0005I hh:mm:ss [taskid/processid]
programname HAS STARTED [OBJECT:*objectname*]

This message tells you that a Manager method, program, or process has started.

Message: EDM0006I hh:mm:ss [taskid/processid]
programname HAS ENDED [OBJECT:*objectname*]

This message tells you that a Manager method, program, or process has ended. The method, program, or process displayed is ended, and continues processing.

Message: EDM0007I hh:mm:ss [taskid/processid]
ATTACHING *managertaskname* [*taskdescription*]

This message tells you that the Manager task displayed in the message is being attached.

This message is written to the message log after the clock Manager task is started, and before the task actually starts.

Message: EDM0010E hh:mm:ss [taskid/processid]
objectname ERROR INITIALIZING HEAPS

This message tells you that the Manager was not able to initialize the heaps for the *objectname* object. The reason for this error may be found in prior messages.

Message: EDM0012E hh:mm:ss [taskid/processid]
processname - OBJECT DATABASE ACCESS ERROR

This message tells you that a read/write error occurred to one of the object databases. See accompanying messages for more information. EDM or Radia continues processing, but cannot perform the operation displayed in the message.

Message: EDM0013I hh:mm:ss [taskid/processid]
objectname - OBJECT NOT FOUND

This message tells you that an in-storage object was referenced that does not exist. This message is issued due to a logic or configuration error. EDM or Radia continues processing, but does not retrieve the object displayed in the message.

Message: EDM0013I hh:mm:ss [taskid/processid]
PROMOTE *instancename* [*instance length*]

This message tells you that the Promote process has begun for the instance cited.

Message: EDM0014E hh:mm:ss [taskid/processid]
variablename - VARIABLE NOT FOUND

This message tells you that a variable access was attempted and the requested variable did not exist.

Message: EDM0017E hh:mm:ss [taskid/processid]
parameters - INVALID PARAMETERS

This message tells you that the program had a problem processing the parameters displayed in the message.

Message: EDM0018E hh:mm:ss [taskid/processid]
parameters - NO INPUT PARAMETERS

This message tells you that no parameters were supplied.

Message: EDM0099I hh:mm:ss [taskid/processid]
OBJECT *objectname* HAS ONLY ONE HEAP-SORT NOT DONE

This message tells you that a sort was not required because the object has one heap.

Message: EDM0100I hh:mm:ss [taskid/processid]
PID is *nnnn*

This message tells you that the parent object ID (PID) is the value cited in the message.

Message: EDM0361E hh:mm:ss [taskid/processid]
ERROR INITIALIZING OBJECT objectname

This message tells you that the Manager was not able to initialize the *objectname* object. The reason for this error may be found in prior messages.

Message: EDM0388W hh:mm:ss [taskid/processid]
WARNING - INSTANCE *class.instance* DOES NOT EXIST

This message tells you that the instance *class.instance* was referenced during an object resolution, but the instance

could not be found in the current configuration database (that is, in the Primary or Secondary file).

This could be the result of a missing or incorrectly spelled instance name, or a failure during variable substitution processing.

Message: EDM0512E hh:mm:ss [taskid/processid]
PUT OBJECT ERROR - REQUESTED OBJECTID NOT FOUND

This message tells you that during PUT-OBJECT processing (during which the Manager sends an object to a client), one of the requested object identifiers (OBJECTID) was not available.

A client has requested an instance of an object by its unique object identifier (OJBECTID), but the instance cannot be located.

Message: EDM0516E hh:mm:ss [taskid/processid]
PUT OBJECT - RESOURCE NOT FOUND

This message tells you that the resource requested by the client could not be found.

Message: EDM0516W hh:mm:ss [taskid/processid]
WARNING-RESOURCE SIZE (ZRSCSIZE) NOT NUMERIC

This message tells you that the value specified for the resource is not numeric. This may impede further processing.

Message: EDM0532E hh:mm:ss [taskid/processid]
path NO SUCH FILE OR DIRECTORY

This message tells you that the Manager is not able to find the file or directory listed in the path.

Message: EDM0532I hh:mm:ss [taskid/processid]
CURRENT PATH *path*

This message tells you where the Manager is currently operating or retrieving a file.

Message: EDM0532I hh:mm:ss [taskid/processid]
methodname METHOD ID

This message identifies the method the Manager is currently running.

Message: EDM0537I hh:mm:ss [taskid/processid]
methodname METHOD COMPLETED RC: 0

This message tells you that the method cited was successfully completed, resulting in a return code of 0.

Message: EDM0539E hh:mm:ss [taskid/processid]
SETTING OBJECT RESOLUTION STOP CODE

This message tells you that the Manager is stopping the object resolution process. The cause for the interruption should be detailed by messages preceding this.

Message: EDM0545I hh:mm:ss [taskid/processid]
RESOLVING *objectname*

This message tells you that the Manager is about to resolve the object cited.

Message: EDM0547I hh:mm:ss [taskid/processid]
objectname INHERITENCE COMPLETE

This message tells you that the object being resolved has inherited all the values of its parent.

Message: EDM0559I hh:mm:ss [taskid/processid]
CLASS NOT PROCESSED-BLANK FIELD _ALWAYS_

This message tells you that the class undergoing object resolution was not processed since the default _ALWAYS_ action was not found.

Message: EDM0566E hh:mm:ss [taskid/processid]
classname CLASS DOES NOT CONTAIN CONTROL INFORMATION

This message tells you that the class cited in the message does not have associated control information for the entire class.

Message: EDM0568E hh:mm:ss [taskid/processid]
OBJECT SERVICES INSTANCE NOT FOUND

This message indicates that an error occurred when the Manager was calling object service to obtain the instance for object resolution.

Message: EDM0579E hh:mm:ss [taskid/processid]
OBJECT RESOLUTION FAILURE FOR OBJECT *objectname.variable*

This message tells you that the Manager was not able to resolve the *objectname.varaiable* object. The reason for this error may be found in prior messages.

Message: EDM0580I hh:mm:ss [taskid/processid]
RESOLUTION PROCESS BEGINS: *class.instance* (*message*)

This message tells you that object resolution processing has begun for *class.instance*. If there is a conditional resolution being processed, the message being processed is specified by *message*.

Message: EDM0580I hh:mm:ss [taskid/processid]
RESOLUTION PROCESS ENDS: *class.instance* CRC: crcvalue

This message tells you that object resolution processing has ended for *class.instance*. The object CRC is specified by *crcvalue*. The CRC is used with the date and time to correctly perform object differencing.

Message: EDM0585E hh:mm:ss [taskid/processid]
ERROR RETRIEVING HEAP nn FOR OBJECT objectname

This message tells you that the Manager was not able to initialize the specific heap for the *objectname* object.

Message: EDM0593E hh:mm:ss [taskid/processid]
EDMMGPRO - PROFILE NOT FOUND FOR USER user

This message tells you that the Manager method EDMMGPRO attempted to retrieve an object from the Profile database for user *user*. The requested object did not exist in the database for that user. If the object should exist for that user, check that the object and user name are

correct, or look in the profile data base with the administrator.

Message: EDM0595II hh:mm:ss [taskid/processid]
`ZMETHOD ZPCB AT xxxxxxxx, DIRECTORY AT xxxxxxxx`

This message tells you where the method control block and the method directory is located.

Message: EDM0596E hh:mm:ss [taskid/processid]
`EDMMGPRO PARAMETER ERROR parameter`

This message tells you that the method EDMMGPRO was not completed due to a parameter error.

Message: EDM0597E hh:mm:ss [taskid/processid]
`PROFILE ERROR - ZUSERID NOT FOUND`

This message tells you that a Profile File operation was halted since the ZUSERID of the client was not found.

Message: EDM0604I hh:mm:ss [taskid/processid]
`sectionname` `-> value`

This message tells you the individual values for the settings of the profile, which are found in the Manager Settings file.

Message: EDM0777I hh:mm:ss [taskid/processid]
`OBJECT` *`objectname`* `PROCESSING BEGINS`

This message tells you that the Manager has begun the object resolution process for the *objectname* object.

Message: EDM0777I hh:mm:ss [taskid/processid]
`OBJECT` *`objectname`* `PROCESSING ENDS`

This message tells you that the Manager has finished the object resolution process for the *objectname* object.

Message: EDM0777I hh:mm:ss [taskid/processid]
`CONNECTED TO CLIENT` *`clientname`*

This message tells you that the Manager has connected to the client *clientname*.

Message: EDM0777E hh:mm:ss [taskid/processid]
INSUFFICIENT MEMORY SIZE path

This message tells you that there was not sufficient memory in the path name *path* to perform a specific operation.

Message: EDM0875I hh:mm:ss [taskid/processid]
ZVARSORT - NO NAMES FOUND

This message tells you that the variable names used for the sorting operation were not found.

Message: EDM0888I hh:mm:ss [taskid/processid]
instancename ZPROCESS HEAP QUEUE ALLOCATED AT xxxxxxxx

This message tells you where the ZPROCESS heap queue is located.

Message: EDM0937E hh:mm:ss [taskid/processid]
ZT0104 GET-OBJECT ERROR

This message indicates that an error occurred when the EDM: Manager was performing GET-OBJECT processing (Manager receiving object from the Client), and a lower level service encountered an error in the processing. See messages in the log prior to see if the cause has been indicated.

Message: EDM0998E hh:mm:ss [taskid/processid]
INSTANCE *instancename* (1) AT xxxxxxxx

This message tells you the location of the instance cited.

Note: A number of different messages are identified by the message number EDM0999I. These messages will be re-assigned in subsequent releases.

Message: EDM0999I hh:mm:ss [taskid/processid]
RETURNED RC:99 END OF SESSION

This message tells you that the Manager has received the return code that terminates communications between the Manager and a client. This ends a client connect.

Message: EDM0999E hh:mm:ss [taskid/processid]
OBJECT NOT FOUND [address]

This message tells you that the Manager was not able to find an expected object at the address cited.

Message: EDM1009E hh:mm:ss [taskid/processid]
objectname COPY OBJECT - INVALID NAME

This message tells you that the object cited could not be copied because the object name was invalid.

Message: EDM1010E hh:mm:ss [taskid/processid]
objectname COPY OBJECT - NEW NAME EXISTS

This message tells you that the object cited could not be copied because the new object name already exists.

Message: EDM1120I hh:mm:ss [taskid/processid]
EDMMTUCH UPDATED RESOURCE *pathname*

This message tells you that the method EDMMTUCH updated the date/time for the resource file cited.

Message: EDM1121E hh:mm:ss [taskid/processid]
EDMMTUCH NO INPUT PARAMETERS

This message tells you that the method EDMMTUCH could not be performed since there were no input parameters to identify the object to be updated.

Message: EDM1121I hh:mm:ss [taskid/processid]
PROFILE SECTION *sectionname*

This message tells you the section of the manager Settings file that the Manager is currently processing.

Message: EDM1161I hh:mm:ss [taskid/processid]
EDMMOLOG FOR OBJECT *objectname*

This message tells you that the object cited will be written to the log.

Message: EDM1163I hh:mm:ss [taskid/processid]
EDMMUPSZ TOTAL MATCHED LENGTHS xxxx

This message tells you the number of resources that matched the parameters given for the method EDMMUPSZ.

Message: EDM1163I hh:mm:ss [taskid/processid]
EDMMUPSZ TOTAL UNMATCHED LENGTHS xxxx

This message tells you the number of resources that were not matched by the parameters given for the method EDMMUPSZ.

Message: EDM1163I hh:mm:ss [taskid/processid]
EDMMUPSZ TOTAL LENGTHS UPDATED xxxx

This message tells you the number of resource file lengths that were updated by the method EDMMUPSZ.

Message: EDM1173E hh:mm:ss [taskid/processid]
EDMMUPSZ NO INPUT PARAMETERS

This message tells you that there were no input parameters given to perform the method EDMMUPSZ.

Message: EDM1175E hh:mm:ss [taskid/processid]
NO ZRSCSIZE FIELD IN CLASS

This message tells you that there is no ZRSCSIZE field in the class template for the instance. Therefore, the method EDMMUPSZ could not be performed.

Message: EDM1175E hh:mm:ss [taskid/processid]
EDMMUPSZ FIELD NOT 8 CHARACTERS

This message tells you that the field cited for the method EDMMUPSZ did not contain the required eight characters. Therefore, EDMMUPSZ could not be performed.

Message: EDM1180I hh:mm:ss [taskid/processid]
EDMMEXIS RECORD WAS FOUND *pathname*

This message tells you that the record requested by the method EDMMEXIS was found in the path cited.

Message: EDM1181I hh:mm:ss [taskid/processid]
EDMMEXIS RECORD NOT FOUND *pathname*

This message tells you that the record requested by the method EDMMEXIS was not found.

Message: EDM1209E hh:mm:ss [taskid/processid]
GET-RECORD DECOMPRESSION ERROR

This message indicates that an error occurred when the Manager was performing a decompression of the received object from the Client. An error was detected in the process of decompressing the record.

Message: EDM1365E hh:mm:ss [taskid/processid]
PROFILE ERROR - ZUSERID NOT FOUND

This message tells you that a Manager Settings file operation could not be done because the ZUSERID was not found.

Message: EDM1367E hh:mm:ss [taskid/processid]
PROFILE - 'FROM-TO' INVALID

This message tells you that a Manager Settings file operation could not be done because the FROM-TO parameters used were not valid.

Message: EDM1370E hh:mm:ss [taskid/processid]
PROFILE DATABASE ERROR

This message tells you that a Profile file database error occurred. The cause of the error can be found in prior messages.

Message: EDM1370E hh:mm:ss [taskid/processid]
PROFILE - DIRECTORY NOT FOUND

This message tells you that the directory cited for a Profile database operation could not be found. The requested process will not be performed.

Message: EDM1370E hh:mm:ss [taskid/processid]
PROFILE - ERROR OPENING FILE

This message tells you that an error occurred during a Profile database error operation. The requested process will not be performed.

Message: EDM1370E hh:mm:ss [taskid/processid]
PROFILE - ERROR UNLINKING FILE

This message tells you that an error occurred during a Profile File operation.

Message: EDM1371I hh:mm:ss [taskid/processid]
PROFILE DATABASE DELETED

This message tells you that a Profile File database was deleted at the conclusion of a Profile database operation.

Message: EDM1977I hh:mm:ss [taskid/processid]
ISSUING WAKE_ON_LAN CALL FOR IP @ <IPAADDRESS>, MAC-addr =<MACADDRESS>

This message tells you that a Wake_On_LAN notify was issued to the IP address cited.

Message: EDM1987I hh:mm:ss [taskid/processid]
EDMWAKE IS ISSUED TO POWER UP THE DESTINAATION MACHINE, WILL RETRY NOTIFY IN <XXX> SECONDS

This message tells you that an EDMWAKE command was issued to the client desktop and the Notify will be attempt again in xxx seconds.

Message: EDM2500E hh:mm:ss [taskid/processid]
UNABLE TO OBTAIN STORAGE

This message tells you that storage is unavailable for the process displayed in the message. EDM continues processing, but cannot perform the operation displayed.

Message: EDM2500E hh:mm:ss [taskid/processid]
INVALID NUMBER OF ARGUMENTS PASSED

This message tells you that too many arguments were used in the operation cited in the message.

Message: EDM2500E hh:mm:ss [taskid/processid]
UNABLE TO PACK OBJECTID

This message lets you know that the Manager was not able to pack the unique object identifier (OBJECTID) for a specific object.

Message: EDM2502E hh:mm:ss [taskid/processid]
methodname METHOD ARGUMENT ERROR

This message tells you that the argument used in the method cited in the message was not correct for that method.

Message: EDM2504E hh:mm:ss [taskid/processid]
ERROR GETTING TEMPLATE

This message tells you that an error occurred in retrieving a class template.

Message: EDM2505E hh:mm:ss [taskid/processid]
ZADMCLAS NOT FOUND IN HEAP *instancename*

This message tells you that the ZADMCLAS variable was not found in any occurrences of the instance cited in the message.

Message: EDM2506E hh:mm:ss [taskid/processid]
FILE OPEN FAILED *filepath [filename]*

This message tells you that the Manager was not able to open a file required for the object resolution process.

Message: EDM2507E hh:mm:ss [taskid/processid]
DIRECTORY OPEN FAILED *classpath*

This message tells you that the Manager was not able to open a directory containing files required for the object resolution process.

Message: EDM2509E hh:mm:ss [taskid/processid]
FILE NOT FOUND *filename*

This message tells you that the Manager was not able to find the expected *filename* file.

Message: EDM2515E hh:mm:ss [taskid/processid]
ERROR READING TEMPLATE ENTRY

This message tells you that the Manager was not able to read an entry for a class template. The class update operation was not completed.

Message: EDM2522E hh:mm:ss [taskid/processid]
TEMPLATE NOT FOUND *classname*

This message tells you that the template for the class cited in the message was not found.

Message: EDM2529E hh:mm:ss [taskid/processid]
SUBSTITUTION ERROR

This message tells you that an error occurred while attempting variable substitution. The cause of this error may be found in prior messages.

Message: EDM2533E hh:mm:ss [taskid/processid]
ERROR ALLOCATING HEAP *objectname*

This message tells you that the Manager was not able to allocate a heap for the *objectname* object.

Message: EDM2536E hh:mm:ss [taskid/processid]
OBJECT TEMPLATE NOT FOUND *objectname*

This message tells you that the Manager was not able to find the template for the object cited in the message.

Message: EDM2539E hh:mm:ss [taskid/processid]
OBJECT HEAP NOT FOUND *objectname* HEAP

This message tells you that no occurrences of the object cited in the message were found.

Message: EDM3000E hh:mm:ss [taskid]
NO HEAPS WERE ALLOCATED FOR OBJECT *objectname*

This message tells you that the Manager did not allocate any occurrences for the *objectname* object.

Message: EDM7777E hh:mm:ss [taskid]
ZINST PARAMETER ERROR

This message indicates that an error was detected on the parameters passed to obtain class.instance. One of the following parameters was not present to allow processing: DOMAIN, CLASS, or INSTANCE. Diagnostic dump of the passed parameters may be present to determine which of the parameter(s) was in error.

Part III

Managing the Database

Part III, Managing the Database, provides information about the 'care and feeding' of your Manager Database. When you installed your Manager, a sample database was created. As you customized the database for your specific environment, you added users, policy, and applications. From time to time, you will need to modify your database either to maintain it efficiently or to reflect changing conditions. Depending on your specific requirement, you can modify your database using either the Database Utilities (Chapter 5) or the EDM Access Method Services utilities (Chapter 6).

Please note that as with most database maintenance, it is recommended that you back up your database before using these utilities.

Part III: Managing the Database, contains two chapters:

- *Chapter 5: Database Utilities*
- *Chapter 6: EDMAMS*

5

Database Utilities

This chapter shows you how to manage the Manager Database using the Database utilities.

> **Note:** Novadigm strongly recommends that you shut down your Manager to ensure that the Manager Database contents are not changing during the execution of the database import and export utilities.
>
> Furthermore, Novadigm strongly recommends that you back up your database prior to running any of the import utilities against a pre-existing database.

Manager Database Utility Programs

The Manager Database utilities are initially used by the installation program to install the Manager's Database. You can then use these utilities to take a test environment database to the production environment, overlay an existing database, or move a database configuration from one operating system to another. The database utilities are located in the BIN subdirectory of the EDMMGR directory (Intel platforms),in the exe subdirectory of the Unix Manager installation and in the MVS LOAD dataset.

Note: If you do not specify optional command line parameters for the utilities in this chapter, please be advised that the default value for the optional parameter will be used.

The Manager Database Utility List

Program Name	Description
EDMMDBSP	Reads a platform-independent binary file produced by the execution of EDMMDBJO and recreates the appropriate class and instance data, based on the contents of the input file. The EDMMDBSP utility is not available for MVS.
EDMMRSSP	Reads a platform-independent binary file produced by the execution of EDMMRSJO and recreates the appropriate RESOURCE data based on the contents of the input file. The EDMMRSSP utility is not available for MVS.
EDMMDBJO	Exports class templates and database instance data into a platform-independent binary file. The EDMMDBJO utility is not available for MVS.
EDMMEXPC	Exports class data from the Manager Database and generates a comma-separated variable file that can be imported into another Manager Database (using EDMMIMPC). Contents include class template and base instance values.
EDMMEXPI	Exports instance data (objects) from the database as a comma-separated variable file for input into a third party tool for reporting purposes or for input into another database (using EDMMIMPR).

Program Name	Description
EDMMEXPR	Exports resources from the Manager resource database, and generates a binary file that can be imported into another Manager Database (using EDMMIMPR).
EDMMIMPC	Imports class data and its base instance from a comma-separated variable file produced when using EDMMEXPC. Imported files or records are updated with a new object ID and time and date stamp if the TIME=NEW parameter is used.
EDMMIMPI	Reads a comma-separated variable list and creates instances from the list created by EDMMEXPI.
EDMMIMPR	Imports resources into the Manager Database. Instances with the same names may be replaced with resource data contained in the input file.

Warning: It is imperative that you periodically back up your production database when working with mission-critical production applications and prior to making changes to the database.

Furthermore, to avoid changes to the database while you are working with its contents, do not invoke these utilities while the Manager is running.

EDMMDBSP

EDMMDBSP is a utility that splits and organizes files containing multiple platform neutral records into the database class and instance equivalents. The utility converts data from EBCDIC to ASCII format.

The content of the input file (database_file) is converted to database objects and instance data.

Syntax

```
EDMMDBSP database_file [record_size] [directory]
```

The default `[record_size]` is 4000.

The `[directory]` should be the directory where the Manager's PRIMARY file is located. To specify a different location, make sure that you have write access to that directory.

Specifying the EDMMDBSP Utility

The following table describes each of the command parameters necessary to run the EDMMDBSP utility.

EDMMDBSP Command Parameters

Parameter	What To Specify	Required or Optional
database_file	The database_file you are converting. In most cases, the name of the file is FBCONF.XXX. The extension of this file varies.	Required
[directory]	The name of the directory in which you are storing the new file, typically, the directory of your EDM database. The current directory will be used if none is defined.	Optional

Parameter	What To Specify	Required or Optional
[record_size]	This size of the file you are creating from the conversion. The default is 4000, or 4 KB.	Optional

Examples

The following examples show some of the ways you can specify the EDMMDBSP command options.

Command Explanation
`EDMMDBSP FBCONF.BIN 4000 directory_path_to_the_database` Separates the database file FBCONF.BIN and stores the output in the specified directory.
`EDMMDBSP FBCONF. BIN` Separates the database file FBCONF.BIN and stores the output in the current directory.

EDMMRSSP

EDMMRSSP is a utility that splits and organizes files containing platform neutral resource data records into separate resource files in binary format. The utility converts data from EBCDIC to ASCII format.

A resource contains one or more records. The multiple records for a resource are joined to create a single file for each resource.

Syntax

```
EDMMRSSP database_file [record_size] [directory]
```

The default `[record_size]` is 3000.

The `[directory]` should be the directory where the Manager's resource file is located. To specify a different directory location, make sure that you have write access to that directory.

Specifying the EDMMRSSP Utility

The following table describes each of the command parameters necessary to run the EDMMRSSP utility.

EDMMRSSP Command Parameters

Parameter	What To Specify	Required or Optional
database_file	The database file you are converting. In most cases, the name of the file is FBRESO.XXX. The extension of this file varies.	Required
[directory]	The name of the directory in which you are storing the new file, typically, the directory of the database. The current directory will be used if none is defined.	Optional

Parameter	What To Specify	Required or Optional
[record_size]	This size of the file you are creating from the conversion. If you do not specify a size, the default is 3000.	Optional

Warning: You should run both the ZINIDMCT utility and the Year 2000 Fix database utility after importing data into a database that is already Year 2000 compliant.

Examples

The following examples show some of the ways you can specify the EDMMRSSP command options.

Command Explanation
`EDMMRSSP FBRESO.BIN 3000 directory_path_to_the_database` Separates the database file FBRESO.BIN and stores its output in database directory path specified.
`EDMMRSSP FBRESO.BIN` Separates the database file FBRESO.BIN and stores the output in the current directory.

EDMMDBJO

This utility groups the objects and instances based on the domains and creates a single binary platform neutral configuration file which can be used by Managers on similar or different platforms. Data is converted from ASCII format to EBCDIC format. Parameters and values on non-Unix platforms are specified by /*character*=*value*. For Unix platforms, the format is –[space] *value*.

Syntax

```
EDMMDBJO [/p] /f=EDM_file /d=EDM_domain
[/c=EDM_class] [/i=EDM_instance] /o=output_file
[/r=record_size]
```

The default `[record_size]` is 4000.

Note: You can use prefixes and wildcards (*) for class and instance names.

You must specify an `[output_file]`. If you do not specify the `[record_size]`, the program prints the maximum record size needed to pack the database into a single file. This data can be subsequently used to create the configuration file.

The `[/p]` option presents a preview list of the exported object displays on the screen. In addition to the exported objects, the `[/p]` option provides statistics on the number of records, classes, and instances, as well as the recommended record size and the database size (in bytes).

Specifying the EDMMDBJO Utility

The following table describes the command parameters for the EDMMDBJO utility.

EDMMDBJO Command Parameters

Parameter	What To Specify	Required or Optional
/c=EDM_class	Specify the name of the source class, such as USER, DEPT, WORKGRP or others under the SYSTEMX domain. Allows wild cards - asterisk (*) only.	Optional
/d=EDM_domain	Specify the name of the source domain, such as SYSTEMX under the PRIMARY file. Allows wild cards -asterisk (*) only.	Required
/f=EDM_file	Specify the name of the source Manager database by name. PRIMARY, PROFILE and HISTORY are the only valid options. Does not allow wild cards.	Required
/i = EDM_instance	Specify the name of the source instance, such as ZCONFIG in the PROFILE File. Allows prefixes and wild cards - asterisk (*) only.	Optional
/o =output_file	Specify the name of the destination output file (with extension) where you want the exported data to reside. The default is the current window (stdout).	Required
/p	This parameter will provide you with a preview of the database file.	Optional
[/r]=	Record size specification.	Optional

Examples

The following examples show some of the ways you can specify the EDMMDBJO command options.

Command Explanation
`EDMMDBJO /f =PRIMARY /d=SYSTEMX /c=* /o=FBCONF.BIN` Exports all the objects under the SYSTEMX domain, PRIMARY file.
`EDMMDBJO /f=PRIMARY /d=SYSTEMX /c=USER /i=I DIFF* /o=FBCONF.BIN` Exports all the objects that begin with a prefix of DIFF under the USER class, SYSTEMX domain, and PRIMARY file.
`EDMMDBJO /f=PRIMARY /o=FBCONF.BIN` Exports all objects under the PRIMARY file to the file FBCONF.BIN.

EDMMEXPI

EDMMEXPI is a utility that is used to export objects from the database. This program generates a modified comma-separated variable (CSV) file of an object. The EDMMEXPI database tool is an extension of the functionality of the original EDMMEXPO tool.

Note: Do not modify the exported CSV file prior to importing it.

This allows you to restrict the attribute data retrieved by the EDMMEXPI tool or it may retrieve all attribute data contained within the instances that you select at the command line.

Note: EDMMEXPI exports the objects (instances) only. It does not export the classes or any underlying resources for the objects. To export the resources, refer to the EDMMEXPR program.

The resulting modified CSV file can be imported into another Manager database using the import facility EDMMIMPI, or used to generate reports by using third party vendor software (for example, Microsoft Access, Microsoft Excel).

To format the modified CSV output file for import into third party vendor software, you can run a Novadigm-provided REXX program, called EXPORT.REX, against the output file of EDMMEXPI. This REXX program will reformat the output of EDMMEXPI into a true CSV file.

In generating a CSV output file, EDMMEXPI will create unique names for connects and methods in a class. Connects can be identified by the CONN*nnnn* format, and methods by METH*nnnn*.

This is only when the **Report=YES** switch is selected. Select **Report=NO** as an option, if you intend to import to another Novadigm database.

You will need to specify certain parameters when executing EDMMEXPI. The literal parameters, such as FILE=, must be specified in upper case, as well as the variable parameters, such as PRIMARY.

For Intel Managers, be sure that the Manager Settings file is in the same directory from which the program is executed. The database path is specified in the DBPATH parameter.

For Unix Managers, be sure that the Manager Settings file (~/.edmprof) has the DBPATH parameter set to the appropriate database path.

For MVS Managers, the DBPATH parameter in the Manager Settings file (PARMLIB) will determine the database path.

Syntax

```
EDMMEXPI PREVIEW=YES/NO,FILE=file_name,DOMAIN=
domain_name,CLASS=class_name,INSTANCE=instance_name,
FROMINST=from_inst,TOINST=to_inst,BASE=YES/NO,
FROMDATE=from_date,FROMTIME=from_time,TODATE=to_date,
TOTIME=to_time,KEEP=keep_list_file,DROP=drop_list_file,
ORDER=YES/NO,OUTPUT=output_file,COMMENT=comment,
CSVL=YES/NO,INPUT=inputfile,HEADER=YES/NO,PHEX=YES/NO
```

Specifying the EDMMEXPI Utility

The following table describes the command parameters for the EDMMEXPI utility.

EDMMEXPI Command Parameters

Parameter	What To Specify	Required or Optional
BASE=	Specify this parameter if you want to include all the attributes that are predefined from the base instance.	Optional
CLASS=	Specify the name of the source class, such as USER, DEPT, WORKGRP, or others under the respective domain. Allows wild cards – asterisk (*) only.	Required
COMMENT=	Specify this parameter if you want to add a comment to the optional output file header.	Optional

Parameter	What To Specify	Required or Optional
CSVL=	YES: Specify this parameter if you want to produce a CSV listing for all attribute values. Currently the output file you specify with the CSVL=YES option would be created in a sub-directory of the current working directory. NO: If you do not want a CSV output file. The default value is NO.	Optional
DOMAIN=	Specify the name of the source domain, such as SYSTEMX under the PRIMARY file. Allows wild cards – asterisk (*) only.	Required
DROP=	Specify the instance attributes that you do not want to retain in the resulting file.	Optional
FILE=	Specify the name of the source Manager Database by name. PRIMARY, PROFILE, or HISTORY are the only valid options. Does not allow wild cards.	Required
FROMDATE=	Specify the start of a range of instances (by date) to export. The correct date format is YYYYMMDD.	Optional
FROMINST=	Specify the start of a range of instances (by instance name) to export. Allows wild cards.	Optional
FROMTIME=	Specify the start of a range of instances (by time) to export. The correct time format is HH:MM:SS.	Optional
HEADER=	YES: Specify this parameter if you want to produce an output header file. NO: If you do not want an output header file. The default value is NO.	Optional

Parameter	What To Specify	Required or Optional
INPUT=	Specify a file which contains the instances in the following format FILE=file_name,DOMAIN=domain_name, CLASS=class_name, INSTANCE=instance_name.	Optional
INSTANCE=	Specify the name of the source instance, such as ZCONFIG under the PROFILE file. Allows prefixes and wild cards – asterisk (*) only.	Required
KEEP=	Specify the instance attributes that you want to retain in the resulting file.	Optional
ORDER=	Specify if you want the resulting file to be ordered by attribute names.	Optional
OUTPUT=	Specify the name of the destination output file (with extension) where you want the exported data to reside. The default is the current window (stdout).	Required
PHEX=	YES: Specify this parameter if you want to output the data portion of variables in printable hex. NO: If you do not want to output the data portion of variables in printable hex. (default is NO.)	Optional
PREVIEW=	YES: Specify this parameter if you want a preview listing of the exported object instances in the input file. NO: Performs the export. The default value is YES.	Optional
TODATE=	Specify the end of a range of instances (by date) to export. . The correct date format is YYYYMMDD.	Optional
TOINST=	Specify the end of a range of instances (by instance name) to export. Allows wild cards.	Optional

Parameter	What To Specify	Required or Optional
TOTIME=	Specify the end of a range of instances (by time) to export. The correct time format is HH:MM:SS.	Optional

MVS User's Note: The following is a sample JCL for EDMMEXPI:

```
//*
//IMPIPRIM EXEC PGM=EDMMEXPI,
//PARM='PREVIEW:NO,FILE:PRIMARY,OUTPUT://DSN:YOUR.PDS(IPRIM
)'
//             COND=(0,NE)
//STEPLIB  DD DSN=YOUR.EDM.LOAD,DISP=SHR
//         DD DISP=SHR,DSN=SASC.C650.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//PREVIEW  DD SYSOUT=*
//LOG      DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//STGRPT   DD SYSOUT=*
//PARMLIB  DD DSN=YOUR.EDM.PARMLIB(STARTUP),DISP=SHR
//PRIMARY  DD DISP=SHR,DSN=YOUR.EDM.PRIMARY,
//   AMP=('STRNO=10,BUFNI=30,BUFND=100')
//*
```

Examples

The following examples show some of the possible ways you can specify the EDMMEXPI command options.

Command Explanation
`EDMMEXPI PREVIEW=NO,0 FILE=PRIMARY,0 OUTPUT=report.fil` Exports all objects under the PRIMARY file placing the result of the export in the file "report.fil".
`EDMMEXPI PREVIEW=NO,FILE=PRIMARY,DOMAIN=SYSTEMX, CLASS=USER,FROMINST="DIFF*",TOINST="DIFF*", OUTPUT=report.fil` Exports all the objects that begin with a prefix of DIFF under the USER class, SYSTEMX domain, and PRIMARY file.

Command Explanation
`EDMMEXPI PREVIEW=NO,BASE=YES,FILE= PRIMARY,DOMAIN=SYSTEMX, CLASS=USER,OUTPUT= report.fil` Exports all the objects under the USER class, SYSTEMX, and PRIMARY file, inheriting the predefined values from the base instance.
`EDMMEXPI PREVIEW=YES,FILE=PRIMARY,DOMAIN=SYSTEMX, CLASS=USER,INSTANCE="*"` Exports all the objects under the USER class, SYSTEMX, and PRIMARY file, displaying the expected results to the edmmexpi.log file.

EDMMEXPR

EDMMEXPR is a utility that exports resources from the Manager resource database. The resulting file can be imported into another Manager database using the EDMMIMPR import facility.

You will need to specify certain parameters when executing EDMMEXPR. The literal parameters, such as FILE=, must be specified in upper case, as well as the variable parameters, such as PRIMARY.

For Intel Managers, be sure that the Manager PROFILE File is in the same directory from which the program is executed. The database path is specified in the DBPATH parameter.

For Unix Managers, be sure that the Manager PROFILE (~/.edmprof) has the DBPATH parameter set to the appropriate database path.

For MVS Managers, the DBPATH parameter in the PARMLIB will determine the database path.

EDMMEXPR is used to join resources data from the Novadigm configuration file (e.g., PRIMARY) into a single platform-independent binary file that may be used between Manager platforms.

Syntax

```
EDMMEXPR PREVIEW=YES/NO,FILE=file_name,DOMAIN=
domain_name,CLASS=class_name,FROMINST=from_inst,
TOINST=to_inst,FROMDATE=from_date,FROMTIME=from_time,
TODATE=to_date,TOTIME=to_time,OUTPUT=output_file,
COMMENT=comment,INPUT=input_file,HEADER=YES/NO
```

Specifying the EDMMEXPR Utility

The following table describes the command parameters for the EDMMEXPR utility.

EDMMEXPR Command Parameters

Parameter	What To Specify	Required or Optional
PREVIEW=	YES: Specify this parameter if you want a preview listing of the exported resources in the input file. NO: Performs the export. The default value is YES.	Optional
FILE=	Specify the name of the source Manager Database by name. PRIMARY and RESOURCE are the only valid options. Does not allow wild cards.	Required
DOMAIN=	Specify the name of the source domain, such as SYSTEMX under the PRIMARY file. Allows wild cards – asterisk (*) only.	Required
CLASS=	Specify the name of the source class, such as the ZRSOURCE. Allows wild cards – asterisk (*) only.	Required
FROMINST=	Specify the start of a range of resources (by instance name) to export. Allows wild cards. (Parameters are not valid when FILE=RESOURCE.)	Optional
TOINST=	Specify the end of a range of resources (by instance name) to export. Allows wild cards. (Parameters are not valid when FILE=RESOURCE.)	Optional
FROMDATE=	Specify the start of a range of resources (by date) to export. The correct date format is YYYYMMDD.	Optional
FROMTIME=	Specify the start of a range of resources (by time) to export. The correct time format is HH:MM:SS.	Optional

Parameter	What To Specify	Required or Optional
TODATE=	Specify the end of a range of resources (by date) to export. The correct date format is YYYYMMDD.	Optional
TOTIME=	Specify the end of a range of resources (by time) to export. The correct time format is HH:MM:SS.	Optional
OUTPUT=	Specify the name of the destination output file (with extension) where you want the exported data to reside. The default is the current window (stdout).	Required
COMMENT=	Specify this parameter if you want to add a comment to the output file header.	Optional
INPUT=	Specify a file which contains the instances in the following format FILE=file_name,DOMAIN=domain_name, CLASS=class_name, FROMINST= instance_name, TOINST=instance_name.	Optional
HEADER=	YES: Specify this parameter if you want to produce an output header file. NO: If you do not want an output header file. The default value is NO.	Optional

Warning: You should run both the ZINIDMCT utility and the Year 2000 Fix database utility after importing data into a database that is already Year 2000 compliant.

MVS User's Note: The following is sample JCL for EDMMEXPR:

```
//*
//IMPR   EXEC PGM=EDMMEXPR,
//PARM='PREVIEW:NO,FILE:RESOURCE,OUTPUT://DSN:YOUR.PDS(RRES
)',
//            COND=(0,NE)
//STEPLIB  DD DSN=YOUR.EDM.LOAD,DISP=SHR
//         DD DISP=SHR,DSN=SASC.C650.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//PREVIEW  DD SYSOUT=*
//LOG      DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//STGRPT   DD SYSOUT=*
//PARMLIB  DD DSN=YOUR.EDM.PARMLIB(STARTUP),DISP=SHR
//PRIMARY  DD DISP=SHR,DSN=YOUR.EDM.PRIMARY,
//   AMP=('STRNO=10,BUFNI=30,BUFND=100')
//RESOURCE DD DISP=SHR,DSN=YOUR.EDM.RESOURCE,
//   AMP=('STRNO=10,BUFNI=30,BUFND=30')
//
```

Examples

The following examples show some of the possible ways you can specify the EDMMEXPR command options.

Command Explanation
`EDMMEXPR PREVIEW=NO,FILE=PRIMARY,OUTPUT=MYFILE.bin` Exports all RESOURCE data for the instances under the PRIMARY file.
`EDMMEXPR PREVIEW=NO,FILE=PRIMARY,FROMINST="TSO*", TOINST="TSO*",OUTPUT=MYFILE.bin` Exports all the RESOURCE data that begin with a prefix of TSO under the PRIMARY file, and produces an export file called "MYFILE.bin".
`EDMMEXPR PREVIEW=NO,FILE=PRIMARY,DOMAIN=SYSTEMX, CLASS=ZRSOURCE,OUTPUT=MYFILE.bin` Exports all the RESOURCE data under the PRIMARY file, SYSTEMX domain, ZRSOURCE class.
`EDMMEXPR PREVIEW=YES,FILE=PRIMARY,OUTPUT=MYFILE.bin` Creates a preview of the expected export. The preview results will be shown to the edmmexpr.log file.

Command Explanation
`EDMMEXPR PREVIEW=YES,FILE=PRIMARY,OUTPUT=MYFILE.bin` Creates a preview of the expected export into the edmmexpr.log file.
`EDMMEXPR PREVIEW=NO,FILE=RESOURCE,OUTPUT=MYFILE.bin` Exports all RESOURCE data for the instances under the RESOURCE file.
`EDMMEXPR PREVIEW=NO,FILE=RESOURCE,DOMAIN=SYSTEMX, CLASS=ZRSOURCE,OUTPUT=MYFILE.bin` Exports all the RESOURCE data under the RESOURCE file, SYSTEMX domain, ZRSOURCE class.
`EDMMEXPR PREVIEW=YES,FILE=RESOURCE,OUTPUT=MYFILE.bin` Creates a preview of the expected export. The preview results will be shown to the edmmexpr.log file.

EDMMEXPC

EDMMEXPC is a utility that exports classes from the Manager Database. The resulting file contains the exported class templates and base instances, and can be imported into another Manager Database using the EDMMIMPC import facility.

You will need to specify certain parameters when executing EDMMEXPC. The literal parameters, such as FILE=, must be specified in upper case, as well as the variable parameters, such as PRIMARY.

For Intel Managers, be sure that the Manager PROFILE file is in the same directory from which the program is executed. The database path is specified in the DBPATH parameter.

For Unix Managers, be sure that the Manager PROFILE (~/.edmprof) has the DBPATH parameter set to the appropriate database path.

For MVS Managers, the DBPATH parameter in the PARMLIB will determine the database path.

Syntax

```
EDMMEXPC FILE=file_name,DOMAIN=domain_name,
CLASS=class_name,PREVIEW=YES/NO,OUTPUT=output_file,
COMMENT=comment,INPUT=input_file,HEADER=YES/NO
```

Specifying the EDMMEXPC Utility

The following table describes the command parameters for the EDMMEXPC utility.

EDMMEXPC Command Parameters

Parameter	What To Specify	Required or Optional
FILE=	Specify the name of the source Manager Database by name. PRIMARY is the only valid option. Does not allow wild cards.	Required
DOMAIN=	Specify the name of the source domain, such as SYSTEMX under the PRIMARY file. Allows wild cards – asterisk (*) only.	Required
CLASS=	Specify the name of the source class, such as USER, DEPT, WORKGRP, or others under the SYSTEMX domain. Allows wild cards – asterisk (*) only.	Required
PREVIEW=	YES: Specify this parameter if you want a preview listing of the exported classes in the input file. NO: Performs the export. The default value is YES .	Optional
OUTPUT=	Specify the name of the destination output file (with extension) where you want the exported data to reside. The default is the current window (stdout). Ignored if PREVIEW=YES.	Required
COMMENT=	Specify this parameter if you want to add a comment to the output file header.	Optional
INPUT=	Specify a file which contains the classes in the following format FILE=file_name,DOMAIN=domain_name, CLASS=,class_name, INSTANCE=.	Optional

Parameter	What To Specify	Required or Optional
HEADER=	YES: Specify this parameter if you want to produce an output header file. NO: If you do not want an output header file. The default value is NO.	Optional

MVS User's Note: The following is sample JCL for EDMMEXPC:

```
//*
//IMPCPRIM EXEC PGM=EDMMIMPC,
//            PARM='PREVIEW:NO,FILE://DSN:YOUR.PDS(CPRIM)',
//            COND=(0,NE)
//STEPLIB  DD DSN=YOUR.EDM.LOAD,DISP=SHR
//         DD DISP=SHR,DSN=SASC.C650.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//PREVIEW  DD SYSOUT=*
//LOG      DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//STGRPT   DD SYSOUT=*
//PARMLIB  DD DSN=YOUR.EDM.PARMLIB(STARTUP),DISP=SHR
//PRIMARY  DD DISP=SHR,DSN=YOUR.EDM.PRIMARY,
//   AMP=('STRNO=10,BUFNI=30,BUFND=100')
//*
```

Examples

The following examples show some of the possible ways you can specify the EDMMEXPC command options.

Command Explanation
`EDMMEXPC PREVIEW=NO,FILE=PRIMARY,OUTPUT=MYFILE.bin` Exports all instances under the PRIMARY file.
`EDMMEXPC PREVIEW=NO,FILE=PRIMARY,DOMAIN=SYSTEMX, CLASS=ZRSOURCE,OUTPUT=MYFILE.bin` Exports the ZRSOURCE class template and base instance attribute values under the PRIMARY file, SYSTEMX domain.
`EDMMEXPC PREVIEW=YES,FILE=PRIMARY,OUTPUT=MYFILE.bin` Creates a preview of the expected export. The preview results will be shown to the edmmexpc.log file.
`EDMMEXPC PREVIEW=NO,FILE=PRIMARY,COMMENT="My comment here"` Inserts the comment specified at the head of the output file.

EDMMIMPI

Note: Prior to importing instance data into the Manger Database the class template that corresponds to the instance data, being imported, must already exist with in the context of the Manager Database.

EDMMIMPI is a utility that reads a comma-separated variable list and database objects from the input file.

You will need to specify certain parameters when executing EDMMIMPI. The literal parameters, such as FILE=, must be specified in upper case, as well as the variable parameters, such as MYFILE.

For Intel Managers, be sure that the Manager PROFILE File is in the same directory in which the program is executed. The database path is specified in the DBPATH parameter.

For Unix Managers, be sure that the Manager PROFILE (~/.edmprof) has the DBPATH parameter set to the appropriate database path.

For MVS Managers, the DBPATH parameter in the PARMLIB will determine the database path.

Syntax

```
EDMMIMPI PREVIEW=YES/NO,FILE=input_file,
TIME=OLD/NEW/MOD,REPLACE=YES/NO,Y2K=YES/NO,
FROMMDOMA=,TODOMA=,CHGCONS=YES/NO
```

Specifying the EDMMIMPI Utility

The following table describes the command parameters for the EDMMIMPI utility.

EDMMIMPI Command Parameters

Parameter	What To Specify	Required or Optional
FILE=	Specify the file containing the input information; that is, the objects you want to create or update.	Required
FROMDOMA=	Domain from which to import the data.	
PREVIEW=	meter if you listing of le contains. port.	Optional
TIME=	ject time and object ID. tance time and the on the time ort. tance time *only* leaving ct ID. This nended in centric he correct SS.	Optional

Caution: The use of the TIME=NEW option should be used with great caution. Introducing new instance object IDs for previously deployed instance data, will cause a redeployment of updated instance data on the next client connect (i.e., ZSERVICE, ZAUDITS, ZAUDITR). This option should not be used when importing class instance data that has underlying physical resource data (For example: ZRSOURCE). Instead the TIME=MOD option should be specified.

Note: Variable names that are present in the input data file but not defined in the class template will be dropped when the object is created.

Parameter	What To Specify	Required or Optional
TODOMA=	Domain in which to import the data. (**Note**: This must be specified if FROMDOMA= is specified.)	
CHGCONS	Change the connect values to the new domain (TODOMA) If FROMDOMA= and TODOMA= is specified then CHGCONS must be specified.	
Y2K=	YES: Converts YY/MM/DD or MM/DD/YY to Y2K format. . Please note that the Y2K date format is YYYYMMDD. NO: Retains the YY/MM/DD or MM/DD/YY format. The default is YES.	Optional

MVS User's Note: The following is sample JCL for EDMMIMPI:

```
//*
//IMPIPRIM EXEC PGM=EDMMIMPI,
//            PARM='PREVIEW:NO,FILE://DSN:YOUR.PDS(IPRIM)',
//            COND=(0,NE)
//STEPLIB  DD DSN=YOUR.EDM.LOAD,DISP=SHR
//         DD DISP=SHR,DSN=SASC.C650.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//PREVIEW  DD SYSOUT=*
//LOG      DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//STGRPT   DD SYSOUT=*
//PARMLIB  DD DSN=YOUR.EDM.PARMLIB(STARTUP),DISP=SHR
//PRIMARY  DD DISP=SHR,DSN=YOUR.EDM.PRIMARY,
//   AMP=('STRNO=10,BUFNI=30,BUFND=100')
```

Examples

The following examples show some of the ways you can specify the EDMMIMPI command options.

Command Explanation
`EDMMIMPI FILE=myfile` Imports a file into an EDM Manager.
`EDMMIMPI PREVIEW=YES,FILE=myfile` Creates a preview listing of input file contents and expected results may be viewed in the edmmimpi.log file.
`EDMMIMPI PREVIEW=YES,FILE=MYFILE` Creates a preview listing of the input file contents into edmmimpi.log for printing.
`EDMMIMPI FILE=MYFILE,TIME=OLD,REPLACE=YES,PREVIEW=NO` Imports the instance data in the file MYFILE, and all object IDs, time and date information would be retained for the imported instance data, and pre-exiting instance data would be replaced.
`EDMMIMPI FILE=MYFILE,TIME=NEW,PREVIEW=NO,Y2K=NO` Imports the instance data in the file MYFILE and all new object IDs, time and date information would be created for the newly imported instance data. In addition, the old, non-Y2K format is retained or already converted.

EDMMIMPR

Note: Prior to importing resource data into a Manager RESOURCE file the instance data that corresponds to the physical resource data must already exist with in the context of the Manager database.

EDMMIMPR is a utility that is used to import resources into a Manager Database. All existing resources with the same name could be replaced by the input file.

You will need to specify certain parameters when executing EDMMIMPR. The literal parameters, such as FILE=, must be specified in upper case, as well as the variable parameters, such as MYFILE.

For Intel Managers, be sure that the Manager PROFILE file is in the same directory from which this program is executed. The database path is specified in the DBPATH parameter.

For Unix Managers, be sure that the Manager PROFILE (~/.edmprof) has the DBPATH parameter set to the appropriate database path.

For MVS Managers, the DBPATH parameter in the PARMLIB will determine the database path.

Syntax

```
EDMMIMPR
PREVIEW=YES/NO,FILE=input_file,REPLACE=YES/NO,
FROMDOMA=,TODOMA=
```

Specifying the EDMMIMPR Utility

The following table describes the command parameters for the EDMMIMPR utility.

EDMMIMPR Command Parameters

Parameter	What To Specify	Required or Optional
FILE=	Specify the file containing the input information; that is, the RESOURCE data you want to create or update.	Required
FROMDOMA=	Domain from which to import the data	
PREVIEW=	YES: Specify this parameter if you want a preview listing of the resources in the input file. NO: Performs the import. The default is YES.	Optional
REPLACE=	YES: Replaces the RESOURCE data if it already exists. NO: If the RESOURCE data already exists, it will not be modified. The default is NO.	
TODOMA=	Domain in which to import the data. (**Note**: This must be specified if FROMDOMA= is specified.)	

Warning: Caution should be used with the REPLACE = YES option. If you choose this option, all existing instances in the file being imported to will be deleted.

MVS User's Note: The following is sample JCL for EDMMIMPR:

```
//*
//IMPR   EXEC PGM=EDMMIMPR,
//            PARM='PREVIEW:NO,FILE://DSN:YOUR.PDS(RRES)',
//            COND=(0,NE)
//STEPLIB  DD DSN=YOUR.EDM.LOAD,DISP=SHR
//         DD DISP=SHR,DSN=SASC.C650.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//PREVIEW  DD SYSOUT=*
//LOG      DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//STGRPT   DD SYSOUT=*
//PARMLIB  DD DSN=YOUR.EDM.PARMLIB(STARTUP),DISP=SHR
//PRIMARY  DD DISP=SHR,DSN=YOUR.EDM.PRIMARY,
//   AMP=('STRNO=10,BUFNI=30,BUFND=100')
//RESOURCE DD DISP=SHR,DSN=YOUR.EDM.RESOURCE,
//   AMP=('STRNO=10,BUFNI=30,BUFND=30')
//
```

Examples

The following examples show some of the ways you can specify the EDMMIMPR command options.

Command Explanation
`EDMMIMPR PREVIEW=NO,FILE=myfile.bin` Imports a file into the Manager. For any of the parameters not specified the default values will be used.
`EDMMIMPR PREVIEW=YES,FILE=myfile.bin` Preview the expected results to the edmmimpr.log file, does not perform the import.
`EDMMIMPR PREVIEW=YES,FILE=myfile.bin,REPLACE=YES` Preview the expected results to the edmmimpr.log file, does not perform the import, though the preview will show what resources would be replaced.

EDMMIMPC

EDMMIMPC is a utility that is used to import classes into a Manager Database. Existing classes with the same name will be replaced by the input file.

You will need to specify certain parameters when executing EDMMIMPC. The literal parameters, such as FILE=, must be specified in upper case, as well as the variable parameters, such as MYFILE.

For Intel Managers, be sure that the Manager PROFILE file is in the same from which directory this program is executed. The database path is specified in the DBPATH parameter.

For Unix Managers, be sure that the Manager PROFILE (~/.edmprof) has the DBPATH parameter set to the appropriate database path.

For MVS Managers, the DBPATH parameter in the PARMLIB will determine the database path.

Warning: When running EDMMIMPC, if the command line parameter REPLACE=YES is specified (or it defaults to YES), an import of a class that already exists will result in the deletion of all pre-existing instances and class template information within the context of the class in the destination Manager Database.

Syntax

```
EDMMIMPC PREVIEW=YES/NO,FILE=input_file,
TIME=OLD/NEW,REPLACE=YES/NO,XDF=YES/NO,Y2K=YES/NO,
FROMDOMA=,TODOMA=
```

Specifying the EDMMIMPC Utility

The following table describes the command parameters for the EDMMIMPC utility.

EDMMIMPC Command Parameters

Parameter	What To Specify
FILE=	The keyword file should be followed by the filename that contains the output from the execution of the EDMMEXPC command.
FROMDOMA=	Domain from which to import the data.
PREVIEW=	YES: Previews the expected results in the edmmimpc.log. No import will be performed. NO: Applies the database changes contained in the input_file_name. The default is YES.
REPLACE=	YES: Replaces the class template if it already exists. The old class template and All instances will be deleted. NO: If the class already exists, it will not be modified. The default is NO.
TIME=	OLD: Retains class object time stamps and object ID. NEW: Updates the class time stamp and the object ID based on the import. The default is OLD. The correct time format is HH:MM:SS.
TODOMA=	Domain in which to import the data. (**Note**: This must be specified if FROMDOMA= is specified.)
XDF=	Effective only when creating a Primary file. The default is YES.

Warning: Caution should be used with the REPLACE = YES option. If you choose this option, all existing instances also in the file being imported to will be deleted.

Caution: The use of the TIME=NEW option should be used with great caution. Introducing new class object IDs for previously deployed class and instance data will cause a redeployment of previously deployed instance data on the next Client Connect (i.e., ZSERVICE, ZRSOURCE).

Parameter	What To Specify
Y2K=	YES: Converts YY/MM/DD or MM/DD/YY to Y2K format. . Please note that the Y2K date format is YYYYMMDD. NO: Retains the YY/MM/DD or MM/DD/YY format. The default is YES.

MVS User's Note: The following is sample JCL for EDMMIMPC:

```
//*
//IMPCPRIM EXEC PGM=EDMMIMPC,
//            PARM='PREVIEW:NO,FILE://DSN:YOUR.PDS(CPRIM)',
//            COND=(0,NE)
//STEPLIB  DD DSN=YOUR.EDM.LOAD,DISP=SHR
//         DD DISP=SHR,DSN=SASC.C650.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//PREVIEW  DD SYSOUT=*
//LOG      DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//STGRPT   DD SYSOUT=*
//PARMLIB  DD DSN=YOUR.EDM.PARMLIB(STARTUP),DISP=SHR
//PRIMARY  DD DISP=SHR,DSN=YOUR.EDM.PRIMARY,
//   AMP=('STRNO=10,BUFNI=30,BUFND=100')
```

Examples

The following examples show some of the ways you can specify the EDMMIMPC command options.

Command Explanation
`EDMMIMPC FILE=myfile,REPLACE=NO,PREVIEW=NO` Imports the class templates and base instance data contained in the file MYFILE into the Manager Database. It will not overwrite a class should the class in the input file already exist in the database.
`EDMMIMPC PREVIEW=YES,FILE=MYFILE,REPLACE=NO` Creates a preview listing of input file contents and expected results may be viewed in the edmmimpc.log file.
`EDMMIMPC PREVIEW=NO,FILE=MYFILE,TIME=OLD,REPLACE=NO` Imports the class templates and base instance data from the file MYFILE, and all object IDs, time and date information would be retained for the newly imported class and instance data. The import would not overwrite pre-existing class templates and the instances within the pre-existing class.
`EDMMIMPC PREVIEW=NO,FILE=MYFILE,TIME=OLD,REPLACE=NO` Imports the class template and base instance data from the file MYFILE, and all object IDs, time and date information would be retained for those classes that existed prior to the import.
`EDMMIMPC FILE=MYFILE,TIME=NEW,PREVIEW=NO,REPLACE=NO` Imports the class template data from the file MYFILE and all new object IDs, time and date information would be created for the newly imported class and instance data.

6

EDM Access Method Services (EDMAMS)

This chapter shows you how to manage your Manager database using the EDM Access Method Services utilities.

> **Note:** Novadigm strongly recommends that you shut down your Manager to ensure that the Manager Database contents are not changing during the execution of the EDMAMS utilities. *However, ZEDMAMS may be run as a method, in which case the Manager must be running.*
>
> Furthermore, Novadigm strongly recommends that you back up your database prior to running any of the EDMAMS utilities that update the database.

EDMAMS

Note: The ZEDMAMS utility is subject to further development. Subsequent printings of *The Manager Guide* will document new or enhanced functionality.

EDMAMS is a core set of utilities that can be used to create, delete, copy, change, and list objects in the Manager Database. Functions are called by the module (ZEDMAMS). Individual ZEDMAMS utilities are invoked using one of the 37 verb names. Available ZEDMAMS verb functions are displayed by entering "ZEDMAMS" on the command line as shown below. For a display of a particular function keyword, enter the following: "ZEDMAMS VERB=FUNCTION". The functions required and optional keywords will be displayed. Optional keywords are enclosed in parentheses.

You will need to specify certain parameters when executing ZEDMAMS. Verbs, keywords, and data may be typed in upper, lower, or mixed case, however the value of string keywords, such as FROMDATA=, TODATA= and STRING= are case-sensitive where indicated. A comma must follow the function name as well as each keyword value, except for the last value. The default value for FILE is PRIMARY.

Output is written to ZEDMAMS.LOG, with the exception of error conditions, which are written to STDERR.

Syntax

```
ZEDMAMS VERB=
```

One way to run the EDMAMS utilities is to contain the commands in a file that's been edited by a text editor. In this case the KEYWORDS may be specified on one or more lines for a single function. Each KEYWORD must be followed by a single comma, except for the last KEYWORD, which must have a space following it. More than one verb may be specified in one file, for example you may want to execute the COPY_DOMAIN verb, followed by the DELETE_CLASS verb, followed by the LIST_CONNECTS verb. In other words, any combination of verbs may be included in one run, as long as the data sets being accessed do not conflict with each other. If an

asterisk (*) is placed in the first column it is considered a comment, no action is taken.

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

```
   ZEDMAMS ZFILE "DRIVE:\PATH\FILE NAME"
```

Example of multiple VERBS executed from one file:

```
   VERB=COPY_DOMAIN,FROMDOMA=SYSTEMX,TODOMA
   IN=SYSTEMA,
   BASEONLY=YES,REPLACE=NO
*
   VERB=DELETE_CLASS,DOMAIN=SYSTEMA,CLASS=T
   ESTCLAS
*
   VERB=LIST_CONS_VAR,DOMAIN=SYSTEMX,CLASS=
   ZRSOURCE, INSTANCE=CICS
```

Note: ZEDMAMS is the executable. ZFILE is the second argument and must in upper case.

Wildcards

EDMAMS supports two types of wildcarding, implicit and explicit. Implicit wildcarding is available for the COPY_INSTANCE, DELETE_INSTANCE, and LIST_INSTANCE verbs. You can specify any portion of the parameter to select all occurrences that contain the portion of the parameter specified. For example, you can specify FROMINST =DIFF to specify all the fields that contain DIFF as any part of the string.

Explicit wildcarding is available for the CHANGE_INST_FIELD, COPY_NEW_SUFFIX, COPY_RESOURCE, LIST_CONS_VAR, LIST_INST_DATA, LIST_ZRSC_FIELDS and SEARCH_INSTANCES verbs. Explicit wildcarding takes the form of PARAMETER= (wildcard string)*. For example, you can specify FROMINST =EDM* to select all the instances that contain EDM as the first part of the string.

Specifying the ZEDMAMS Utility

The table below describes the verbs for the ZEDMAMS utility.

ZEDMAMS Verbs

Verb	Description (bold denotes the default value)
ADD_FIELD **Syntax:**	Adds a variable at the end of a template, including an automatic README file. (FILE=),DOMAIN=,CLASS=,FLDNAME=,LENGTH=,TYPE=,(MFLAGS=),(CFLAGS=),(KEEPDATE=)
CHANGE_FIELDNAME **Syntax:**	Changes variable names in class templates. (FILE=),DOMAIN=,CLASS=,FROMNAME=,TONAME=
CHANGE_FLD_VALUE **Syntax:**	Changes a template's variable length, type, Manager, and Client flags. (FILE=),DOMAIN=,CLASS=,(DESC=),(FLDNAME=),(LENGTH=),(TYPE=),(MFLAGS=),(CFLAGS=),(INDEX=),(KEEPDATE=)
CHANGE_INS_FIELD **Syntax:**	Changes one field in each instance of a class and verifies connects. (FILE=),DOMAIN=,CLASS=,(PREVIEW=YES/NO),(PREFIX=),FIELD=,TODATA=,(INDEX=),(VERIFY=YES/NO),(KEEPDATE=YES/NO)
CHANGE_INST_DATA **Syntax:**	Changes data globally in instance records by class. (FILE=),DOMAIN=,CLASS=,(FIELD=),FROMDATA=,(TODATA=),(PREVIEW=**YES**/NO)
CHECK_RESOURCES **Syntax:**	Verifies the size of resources against ZRSCSIZE and ZCMPSIZE. (LISTALL=YES/NO)
CLONE_INSTANCE **Syntax:**	Clones an instance with a four-digit suffix (max. 2000). DOMAIN=,CLASS=,INSTANCE=,COUNT=nnnn
COPY_CLASS **Syntax:**	Copies a class, its instances and its resource data if it exists. (FILE=),FROMDOMA=,FROMCLAS=,TODOMAIN=,(TOCLASS=),(REPLACE=YES/**NO**),(BASEONLY=YES/**NO**)
COPY_DOMAIN **Syntax:**	Copies domains. (FILE=),FROMDOMA=,TODOMAIN=,(BASEONLY=YES/**NO**),(REPLACE=YES/**NO**)
COPY_FIELD **Syntax:**	Copies attribute data to a new attribute or to an existing attribute. (FILE=),DOMAIN=,CLASS=,FROMFLD=,TOFLD=,(INDEX=),(PREVIEW=),(REPLACE=YES/NO)

Verb	Description
COPY_INSTANCE **Syntax:**	Copies an instance or a range of instances. (PREVIEW=**YES**/NO),(FILE=),FROMDOMA=,FROMCLAS=, TODOMAIN=,FROMINST=,(TOINST=),(REPLACE=YES/**NO**), (KEEPDATE=YES/**NO**)
COPY_NEW_SUFFIX **Syntax:**	Copies and re-names the suffix for ZRSOURCE class and mated instances. (PREVIEW=**YES**/NO),FROMDOMA=,FROMCLAS=,TODOMAIN=, FROMINST,OLDSUFF=,NEWSUFF=
COPY_RESOURCE **Syntax:**	Copies an instance or instance group and their mated resource.records. (PREVIEW=**YES**/NO),FROMDOMA=,FROMCLAS=,TODOMAIN=, FROMINST=
CREATE_INSTANCES **Syntax:**	Writes and populates instances from data in an edited text file. INFILE=,DOMAIN=,CLASS=
DELETE_CLASS **Syntax:**	Deletes class and instances. (FILE=),DOMAIN=,CLASS=
DELETE_DOMAIN **Syntax:**	Deletes (1) domain to a range of domains. (FILE=),(PREVIEW=**YES**/NO),FROMDOMA=,(TODOMAIN=)
DELETE_FIELD **Syntax:**	Deletes an attribute from a template. DOMAIN=,CLASS=,FIELD=,INDEX=(1-9)
DELETE_INSTANCE **Syntax:**	Deletes/displays a range of instances within a class and its resource data if it exists. (PREVIEW=**YES**/NO),(FILE=),DOMAIN=,CLASS=,FROMINST=, (TOINST=),(SUFFIX=)
DELETE_ORPHANS **Syntax:**	Deletes unmated (orphans) resource records from the RESOURCE file. (PREVIEW=)
LIST_CONS_VARS **Syntax:**	Displays *connect* and *variables* field (types C and V) values from instance records and stores output in ZEDMAMS.LOG. (FILE=),DOMAIN=,CLASS=,(INSTANCE=),(VTYPE=YES/**NO**)
LIST_DOMAINS **Syntax:**	Display a list of domains in the log of a specified file. (FILE=),(FROMDOMA=),(TODOMAIN=)
LIST_FLAGS **Syntax:**	Lists Manager and Client flags in selected class record. DOMAIN=,CLASS=,(README=YES/**NO**)

Verb	Description
LIST_INSTANCE **Syntax:**	Lists instance names by domain, class, and instance (prefix). `DOMAIN=,CLASS=,(FROMINST=),(TOINST=)`
LIST_INST_DATA **Syntax:**	Lists instance data including variable names to ZEDMAMS.LOG `DOMAIN=,CLASS=,(INSTANCE=)`
LIST_PREFIX **Syntax:**	Lists the DMA prefix by file, domain and class. `(FILE=),(DOMAIN=),(CLASS=)`
LIST_RESOURCES **Syntax:**	Lists RESOURCE prefix information (promote date, instance, and so forth). `DOMAIN=`
LIST_ZRSC_FIELDS **Syntax:**	Lists field values of fields beginning with ZRSC. `DOMAIN=,CLASS=,(INSTANCE=)`
MATCH_RESOURCES **Syntax:**	Matches RESOURCE records against ZRSOURCE class records. `(PREVIEW=`**`YES`**`/NO),DOMAIN=`
REFRESH_DMA **Syntax:**	Recounts the instances and classes in a file and updates the DMA prefix. `(PREVIEW=YES/NO),(DOMAIN=),(CLASS=)`
RENAME_INSTANCE **Syntax:**	Renames or displays instances by full name or prefix. `DOMAIN=,CLASS=,OLDPREFIX=,NEWPREFIX=,(KEEP=YES/NO),(PREVIEW=YES/NO)`
SEARCH_INSTANCES **Syntax:**	Searches selected instances by domain and class for specified string. `DOMAIN=,CLASS=,(FROMINST=),STRING=`
SORT_OBJECT_ID **Syntax:**	Sorts object IDs in ascending or descending order by file or domain. `(FILE=),DOMAIN=(ALL/DOMAIN),ORDER=(ASCEND/DESCEND/NOSORT)`
UPDATE_INSTANCES **Syntax:**	Updates existing instance fields from data in an edited text file. `INFILE=,DOMAIN=,CLASS=`
UPDATE_MGRIDS **Syntax:**	Updates the Manager ID and Manager Name by file, domain, and class. `(FILE=),(DOMAIN=),(CLASS=),(MID=),(MMID=),(MNAME=),(MMNAME=)`
VERIFY_CLASS **Syntax:**	Verifies class templates for gaps, overlaps, and other anomalies. `DOMAIN=(ALL/DOMAIN),CLASS=(ALL/CLASS)`
ZRSOURCE_UNMATES	Matches ZRSOURCE instances with the RESOURCE.

```
(PREVIEW=YES/NO),DOMAIN=,(CLASS=),(DEBUG=YES/NO)
```

The following sections describe each ZEDMAMS verb.

ADD_FIELD

This EDMAMS verb adds a variable (attribute) to the end of a class template and unconditionally includes a README attribute. The length of the README attribute is 35 characters. The LENGTH of the specified attribute may not be greater than 255 characters or less than one character. The Manager and Client 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 may be any one- to eight-character name and may be changed with the CHANGE_FIELDNAME function below.

Syntax: `(FILE=),DOMAIN=,CLASS=,FLDNAME=,LENGTH=,TYPE=,(MFLAGS=),(CFLAGS=)`

Example: Add an 'M' type attribute, NEWFIELD, to the specified class, giving it a length of 165 characters, assigning default Manager and client flags, and updating the object date and time:

```
DOMAIN=SYSTEMX,CLASS=USER,FLDNAME=NEWFIELD,
LENGTH=165,TYPE=M
```

Tip: Run LIST_FLAGS against the class before and after running this to view the old and new template.

CHANGE_FIELDNAME

This EDMAMS verb changes the specified class template field name (FROMNAME) to the new field name (TONAME). If there are multiple-named field names (e.g., 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=`

Example: To change the template field name in the specified class from GROUPID to GROUP type:

```
DOMAIN=SYSTEMX,CLASS=USER,FROMNAME=GROUPID,
TONAME=GROUP
```

Tip: Run LIST_INST_DATA to display template field names with INSTANCE=_BASE.

CHANGE_FLD_VALUE

This EDMAMS verb changes a variable's LENGTH, TYPE, Manager and Client flags by FLDNAME. The LENGTH must be in the range of 1 to 255 characters. TYPE= is required. The Manager and Client flags (MFLAGS and CFLAGS) are optional and will retain their current settings if omitted. INDEX changes the nth occurrence of variable FLDNAME=, when omitted, the first variable with a matching FLDNAME will be changed.

Syntax: `(FILE=),DOMAIN=,CLASS=,FLDNAME=,(LENGTH=),TYPE=,(MFLAGS=),(INDEX= )`

Example: To change the TYPE value of the third attribute named EDMSETUP in the template specified from 'V' to 'C'. (The length, and Manager and Client flags retain their current value.)

```
DOMAIN=SYSTEMX,CLASS=ZSERVICE,
FLDNAME=EDMSETUP,TYPE=C,INDEX=3
```

Tip: Run LIST_FLAGS against the class before and after running this to view the old and new template.

CHANGE_INS_FIELD

This EDMAMS verb changes the instance data of a specific field (FIELD=) in specific instances (PREFIX=, SUFFIX=) to TODATA=. INDEX is the relative number (1– 99) of multiple same-named fields (e.g., EDMSETUP or README). VERIFY=YES verifies that a connection value in TODATA exists. If the verify fails, the program aborts. KEEPDATE=YES will prevent the OBJDATE and OBJTIME from being updated. TODATA may be entered in any case, the data is placed in the field as entered. If TODATA contains imbedded spaces the string must be enclosed in double quotes. 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.

This verb allows wildcarding for the PREFIX parameter. For example, you can specify PREFIX=EDM* to view all the prefixes that contain EDM as the first part of the string.

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.EDM _RESOURCE_FILE", verify the existence of the connection, and update the OBJDATE and OBJTIME.

```
DOMAIN=SYSTEMX,CLASS=USER,PREFIX=TSO,
FIELD=EDMSETUP,
TODATA=ZLOCMGR.EDM_RESOURCE_FILE,
VERIFY=YES,INDEX=3
```

CHANGE_INST_DATA

This EDMAMS verb changes instance data from FROMDATA to TODATA. A match occurs only when FROMDATA begins the instance field, and is not imbedded or part of other data in the field. Omitting TODATA will set the field to blanks. FROMDATA and TODATA may be entered in any case, however FROMDATA is case sensitive. If either FROMDATA or TODATA contain imbedded spaces, the string must be enclosed in double quotes.

Syntax: `(FILE=),DOMAIN=,CLASS=,FROMDATA=,(TODATA=),(PREVIEW=)`

Example: To change all instance data in the specified class from JohnPublic to John Q. Doe type:

```
DOMAIN=SYSTEMX,CLASS=USER,
FROMDATA=JohnPublic,TODATA="John Q. Doe"
```

Tip: Run LIST_INST_DATA to get a display of instance data and the class field names to which the data belongs.

Tip: Run first with PREVIEW=YES to display the current variable data values.

CHECK_RESOURCES

This EDMAMS 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, then if ZCMPSIZE contains a value it is compared against the actual size. If ZCMPSIZE is zero or blank then 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 on. If LISTALL=YES, then all objects checked are listed, and objects with resources are listed with their respective sizes.

Syntax: `(LISTALL=YES/NO)`

Example: List all resources that have a mismatch.

```
CHECK_RESOURCES
```

CLONE_INSTANCE

This EDMAMS verb clones the instance specified (*nnnn*) times and each new instance name is suffixed with one to four digits: 0000 through *nnnn*-1. The cloned instance name plus its suffix may not exceed 32 characters: one to 28 digits for the instance name, plus one to four digits for the suffix. Therefore the

instance name length = 32 – the length of the suffix.

A new object ID, OBJDATE, and time are generated for the cloned objects.

Syntax: `DOMAIN=,CLASS=,INSTANCE=,COUNT=nnnn`

Example: To clone 100 instances in the specified class naming the cloned instances DIFF80000 through DIFF80099 type:

```
DOMAIN=SYSTEMX,CLASS=USER,INSTANCE=DIFF8,
COUNT=100
```

Tip: Run LIST_INSTANCE to display the instance names before and after the cloning.

COPY_CLASS

This EDMAMS verb copies a class template, instances, and data from FROMDOMA to TODOMAIN. If TOCLASS is omitted, the FROMCLAS will be the name of the destination class, in which case TODOMAIN and FROMDOMA must be different. BASEONLY=YES copies the class template and base instance only. 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) in order to create a valid domain.

Syntax: `(FILE=),FROMDOMA=,FROMCLAS=,TODOMAIN=,(TOCLASS=),(REPLACE=YES/NO),(BASEONLY=YES/NO)`

Example: To copy the specified class template and only the base instance from SYSTEMX to SYSTEMA type:

```
FROMDOMA=SYSTEMX,FROMCLAS=USER,
TODOMAIN=SYSTEMA,BASEONLY=YES
```

Tip: Run once with REPLACE=NO to determine the pre-existence of the class in the destination domain.

COPY_DOMAIN

This EDMAMS verb copies a domain (classes and instances) from FROMDOMA to TODOMAIN. BASEONLY=YES copies only the class template and base instance for each class. A new object ID, OBJDATE, and time are generated for the copied objects. The ZRSOURCE class will not be copied. Use COPY_CLASS, COPY_RESOURCE, or COPY_NEW_SUFFIX to copy the ZRSOURCE class and/or instances.

Syntax: `(FILE=),FROMDOMA=,TODOMAIN=,`
`(BASEONLY=YES/NO),(REPLACE=YES/NO)`

Example: To copy SYSTEMX to SYSTEMA type:

```
FROMDOMA=SYSTEMX,TODOMAIN=SYSTEMA
```

Tip: Run once with REPLACE=NO to determine the pre-existence of the destination domain.

COPY_FIELD

This EDMAMS 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 multiply 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.

Syntax: `(FILE=),DOMAIN=,CLASS=,FROMFLD=,TOFLD=,(INDEX=),(PREVIEW=),(REPLACE=YES/NO)`

Example: To copy an attribute named OLDFLD and the associated data in the specified Class to a new attribute named NEWFLD:

```
DOMAIN=SYSTEMX,CLASS=USER,FROMFLD=OLDFLD,
TOFLD=NEWFLD
```

Tip: Run LIST_INST_DATA against the Class to view the contents of each attribute.

COPY_INSTANCE

This EDMAMS verb copies a range of specified instances (FROMINST to TOINST) from one domain and class to another domain and class. The function assumes that the destination class has the same name as the source class and that the templates are identical. The FROMINST may be partially specified. The TOINST may be partially specified or omitted. If omitted, only the FROMINST(s) are copied. A new object ID is generated for the copied objects. A new OBJDATE and OBJTIME will be generated unless KEEPDATE=YES is specified. PREVIEW=YES will display a list of instance names that would be copied. PREVIEW=NO will display instances that are copied. If REPLACE=NO is specified, the function will abort if an existing instance is found. The existing instance name will be written to STDERR and instances that have been copied will be listed in the log. If the destination domain does not contain FROMCLAS, then the source class (FROMCLAS) and its BASE_INSTANCE will be copied to the destination domain (TODOMAIN), if PREVIEW=NO. ZRSOURCE instances will not be copied using COPY_INSTANCE. Use COPY_RESOURCE or COPY_NEW_SUFFIX.

This verb allows wildcarding for the FROMINST and TOINST parameters. For example, you can specify FROMINST=DIFF to specify all the instances that contain DIFF as any part of the string.

Syntax:
```
(PREVIEW=YES/NO),(FILE=),FROMDOMA=,
FROMCLAS=,TODOMAIN=,FROMINST=,(TOINST=),
(REPLACE=YES/NO),(KEEPDATE=YES/NO)
```

Example: To copy instances prefixed with the name DIFF from SYSTEMX to SYSTEMA, and do not replace existing instances.

```
PREVIEW=NO,FROMDOMA=SYSTEMX,FROMCLAS=USER,
TODOMAIN=SYSTEMA,FROMINST=DIFF
```

Tip: Run once with PREVIEW=YES to get a list of instances to be copied and REPLACE=NO to

determine the pre-existence of the specified instances.

COPY_NEW_SUFFIX

This EDMAMS verb copies the specified ZRSOURCE instances and resource data from FROMDOMA to TODOMAIN in the PRIMARY and RESOURCE files and re-suffixes the destination instances with the new suffix. The FROMINST may be partially specified. If the NEWSUFF length is longer than the OLDSUFF length, the new instance name must not exceed 32 characters. 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 ZRSOURCE class does not exist in the TODOMAIN, the ZRSOURCE class and BASE_INSTANCE will be copied from the source domain (FROMDOMA). Instance names must match both the prefix (FROMINST) and the suffix (OLDSUFF) in order to be copied and renamed.

This verb allows wildcarding for the FROMINST parameter. For example, you can specify FROMINST =EDM* to select all the instances that contain EDM as the first part of the string.

Syntax: `(PREVIEW=YES-NO),FROMDOMA=,TODOMAIN=, FROMINST=,OLDSUFF=,NEWSUFF=`

Example: To copy instances and RESOURCE data from and to the specified domains, to replace the "ISO" suffix of instances having the prefix "TSO" with the suffix "NEW_ICO," and to display only those that would be copied, type:

```
FROMDOMA=SYSTEMX,TODOMAIN=SYSTEMB,
FROMINST=TSO,OLDSUFF=ICO,NEWSUFF=NEW_ICO
```

Tip Run once with PREVIEW=YES to verify that the new instance names will not exceed 32 characters 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 and pre-exist will be deleted.

COPY_RESOURCE

This EDMAMS verb copies the specified ZRSOURCE instances and resource data from FROMDOMA to TODOMAIN in the PRIMARY and RESOURCE files. The FROMINST may be partially specified. If SUFFIX is specified, then the instance names must match both the prefix (FROMINST) and the suffix (SUFFIX) in order to be copied. PREVIEW=YES will display the instances that will be copied from the source class. If the ZRSOURCE class does not exist in the TODOMAIN, the ZRSOURCE class and BASE_INSTANCE will be copied from the source domain (FROMDOMA).

This verb allows wildcarding for the FROMINST parameter. For example, you can specify FROMINST =EDM* to select all the instances that contain EDM as the first part of the string.

Syntax: `(PREVIEW=YES/NO),FROMDOMA=,TODOMAIN=, FROMCLAS=,FROMINST=`

Example: To copy instances and RESOURCE data from the specified domain with instance prefixes CICS and instance suffixes ICO type:

```
FROMDOMA=SYSTEMX,TODOMAIN=SYSTEMC,
FROMCLAS=ZRSOURCE,FROMINST=CICS
```

Tip: Run once with PREVIEW=YES to view which instances will be copied. Also run LIST_RESOURCES against the source domain to display a list of existing resources and instance names that may be copied and then against the destination domain to see what was copied. Destination domain instance names that pre-exist will be deleted.

CREATE_INSTANCES

The following example updates 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 starting in column 1 for a maximum of 32 characters.

The next line contains the variable field name beginning in column 2 for a maximum of eight characters, followed by data to be populated at creation time beginning in column 11.

Syntax: `INFILE=,DOMAIN=,CLASS=`

Example: Update the instances in the specified text file into:

```
CLASS=INFILE=MYINPUT.TXT,
DOMAIN=SYSTEMX,CLASS=USER
```

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

DELETE_CLASS

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

Syntax: `(FILE=),DOMAIN=,CLASS=`

Example: To delete class USERTEST from domain SYSTEST type:

```
DOMAIN=SYSTEST,CLASS=USERTEST
```

DELETE_DOMAIN

This EDMAMS verb deletes a domain or an alphabetical range of domains (class templates and instances) from the file specified. Omitting TODOMAIN results in deletion of the FROMDOMA only.

Syntax: `FILE=,FROMDOMA=,(TODOMAIN=),(PREVIEW=)`

Examples: To delete SYSTEMB only from the PRIMARY file type:

```
FILE=PRIMARY,FROMDOMA=SYSTEMB
```

To delete USERA through USERF from the PROFILE file type:

```
FILE=PROFILE,FROMDOMA=USERA,TODOMAIN=USERF
```

Tip: Use caution when specifying a range of domains. Be certain of the range specified, there is no confirmation requested. Run with PREVIEW=YES first.

DELETE_FIELD

This EDMAMS verb deletes an attribute from a template along with its README field and re-organizes the class accordingly. FIELD= must refer to an attribute and 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.

Syntax: `DOMAIN=,CLASS=,FIELD=,INDEX=(1-9)`

Examples: Delete the attribute USERATTR from the USER class. Type:

```
DOMAIN=SYSTEMA,CLASS=USER,FIELD=USERATTR
```

Tip: Use with discretion because deletion of an attribute causes a re-structuring and re-write of all the instances in the class.

DELETE_INSTANCE

This EDMAMS verb deletes an instance or a range of alphabetical instances within a specific class. The FROMINST and TOINST may be omitted or be partially specified. Omitting TOINST results in deletion of FROMINST instances only. No confirmation is requested before the delete is executed. PREVIEW=YES displays a list of instances that would be deleted with PREVIEW=NO.

This verb allows wildcarding for the FROMINST and TOINST parameters. For example, you can specify FROMINST=DIFF to select all the instances that contain DIFF as any part of the string.

Syntax: `(PREVIEW=YES/NO),(FILE=),(PREVIEW=),`
`DOMAIN=,CLASS=,FROMINST=,(TOINST=),`
`(SUFFIX=)`

Example: `DOMAIN=SYSTEMX,CLASS=USER,FROMINST=OLD_USER`

DELETE_ORPHANS

This EDMAMS verb will delete all orphans in all domains. Orphans are defined as resource file data that have no mated instance in the PRIMARY file. PREVIEW=YES is the default.

Example: `ZEDMAMS VERB=DELETE_ORPHANS,PREVIEW=`

LIST_CONS_VARS

This EDMAMS verb displays a list of *connect-to* data (type C) and variable data (type V) for the specified instance(s). Omit INSTANCE= to list data for all type Cs and type Vs in all instances of the specified class. Wildcards (*) may be specified in INSTANCE=, for example, EDM* or EDM*SOL*. To display one instance only, the entire name may be required.

This verb allows wildcarding for the INSTANCE parameter. For example, you can specify EDM* to select all the instances that contain EDM as the first part of the string.

Syntax: `(FILE=),DOMAIN=,CLASS=,(INSTANCE=),(VTYPE=YES/NO)`

Example: To list *connect to* values and variable values in the USER class only for instances prefixed with DIFF type:

```
DOMAIN=SYSTEMX,CLASS=USER,INSTANCE=DIFF*
```

LIST_DOMAINS

This EDMAMS verb displays a list of domains for a specified file (default PRIMARY). Omit FROMDOMA= and TODOMAIN= to get a list of all the domains in the file.

Syntax: `(FILE=),(FROMDOMA=),(TODOMAIN=)`

Example: `FILE=PROFILE,FROMDOMA=NAMEA,TODOMAIN=NAMEZ`

LIST_FLAGS

This EDMAMS verb displays the attribute name, length, type, and Manager and Client flags for a specific class template.

Syntax: `DOMAIN=,CLASS=,(README=YES/NO)`

Example: `DOMAIN=SYSTEMX,CLASS=ZSERVICE`

LIST_INST_DATA

This EDMAMS verb displays the variable name and data of all data types for the instances specified.

This verb allows wildcarding for the INSTANCE parameter. For example, you can specify INSTANCE=EDM* to select all the instances that contain EDM as the first part of the string.

Syntax: `DOMAIN=,CLASS=,(INSTANCE=)`

Example: `DOMAIN=SYSTEMX,CLASS=ZRSOURCE,INSTANCE=TSO*`

LIST_INSTANCE

This EDMAMS verb displays a list of instance names and object IDs for the specified class. If FROMINST and TOINST are omitted, all instances are listed. If just TOINST is omitted, only the FROMINST groups will be listed. FROMINST and TOINST may be fully or partially specified to display a range of instances.

This verb allows wildcarding for the FROMINST and TOINST parameters. For example, you can specify FROMINST =DIFF to select all the instances that contain DIFF as any part of the string.

Syntax: `DOMAIN=,CLASS=,(FROMINST=),(TOINST=)`

Example: `DOMAIN=SYSTEMX,CLASS=USER,FROMINST=DIFF`

LIST_PREFIX

This EDMAMS verb displays data from the DMA prefix. If DOMAIN and CLASS are omitted all class prefixes will be displayed. All keywords are optional thereby requiring (FILE=) to display usage.

Syntax: `(FILE=),(DOMAIN=),(CLASS=)`

Example: List the DMA prefixes for all classes in SYSTEMX domain of the PRIMARY file.

```
DOMAIN=SYSTEMX
```

LIST_RESOURCE

This EDMAMS verb goes against the RESOURCE and the PRIMARY files to display a list of names with there promote dates, times, data sizes, and object IDs. The domain entered should be the domain where the ZRSOURCE class is resident. If there is no mated instance in the PRIMARY file, a message will indicate this.

Syntax: `DOMAIN=`

Example: `DOMAIN=SYSTEMX`

LIST_ZRSC_FIELDS

This EDMAMS verb displays the data in all fields that begin with "ZRSC" such as ZRSCSIZE and ZRSCVRFY. The specifics for INSTANCE= are the same as described in LIST_CONNECTS.

This verb allows wildcarding for the INSTANCE parameter. For example, you can specify INSTANCE=EDM* to select all the instances that contain EDM as the first part of the string.

Syntax: `DOMAIN=,CLASS=,(INSTANCE=)*`

Example: `DOMAIN=SYSTEMX,CLASS=ZRSOURCE,INSTANCE=CICS`

MATCH_RESOURCES

This EDMAMS verb will compare resource data from the Resource file against instance names from the Primary file to determine and display those resources that are or are not 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 instance. If PREVIEW=NO is specified, resource data determined to be orphaned will cause a search of the Primary file instances to find a matching instance object ID in the resource file. If a match is made the instance name is placed in the resource data prefix and the resource is updated. Additionally, the time stamp for the resource data will be updated with the ZRSCDATE and ZRSCTIME. Totals at completion include resources found, total mated and total orphaned resources.

> **Caution:** When using PREVIEW=NO, a great deal of I/O may occur, if a great many orphans are detected with PREVIEW=YES, then using the verb ZRSOURCE_UNMATES detailed below will be a more efficient way to update the Resource data file.

Syntax: `(PREVIEW=YES/NO),DOMAIN=`

Example: Match and display Resource data names in the Resource File under domain SYSTEMX against Instance names in the Primary file under domain SYSTEMX.

```
PREVIEW=YES,DOMAIN=SYSTEMX
```

REFRESH_DMA

This EDMAMS verb will re-count all the instances and classes and domains in the Primary file, and refresh the TOTAL INSTANCE COUNTS, TOTAL CLASS COUNTS, and TOTAL DOMAIN COUNTS in the appropriate DMA prefix areas and display the updated output in the log. All keywords are optional except PREVIEW=YES/NO, which must be specified. If YES is specified, 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 "count" calculated and the "current count value" in the DMA prefix. If the newly calculated count differs from the "current count" the "current count" will be flagged with an asterisk. Additionally, the LastUpdateDate, the LastInstanceUpdateDate, and the LastClassUpdateDate for each applicable DMA Prefix will be computed and updated (PREVIEW=NO). Each class for each domain will be previewed separately, and at the end of each domain a domain summary will be previewed. After the last domain, a file summary will be previewed by listing the ZBASE.ZBASE template information.

Syntax: `(PREVIEW=YES/NO),(DOMAIN=),(CLASS=)`

Example: Preview the counts and update dates in the DMA prefix for the entire Primary file.

```
PREVIEW=YES
```

RENAME_INSTANCE

This EDMAMS verb will rename instances and the internal name of any mated resource data. In the syntax indicated below, KEEP indicates that the old instance will not be erased.

Syntax: `DOMAIN=,CLASS=,OLDPREFIX=,NEWPREFIX=,`
`(KEEP=YES/NO),(PREVIEW=YES/NO)`

SEARCH_INSTANCES

This EDMAMS verb will search the specified instances for the data contained in STRING=. The data specified is not case-sensitive, but must be enclosed in double quotes if it contains embedded spaces. The output log will contain the name of the Instance, attribute and the STRING= value. If no string is found in the instances specified, this will be reported in the log.

This verb allows wildcarding for the FROMINST parameter. For example, you can specify FROMINST=EDM* to select all the instances that contain EDM as the first part of the string.

Syntax: `DOMAIN=,CLASS=,(FROMINST=),STRING=`

Example: `DOMAIN=X,CLASS=USER,STRING="J.Q. Public"`

SORT_OBJECT_ID

This EDMAMS verb will sort object IDs in a domain. NOSORT means that the object IDs will be listed in class.instance groups. Duplicate object IDs will be flagged in ascending or descending orders.

Syntax:
```
(FILE=),DOMAIN=(ALL/DOMAIN),
ORDER=(ASCEND/DESCEND/NOSORT)
```

UPDATE_INSTANCES

This EDMAMS 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 starting in column one for a maximum of 32 characters.

The next line contains the variable field name beginning in column 2 for a maximum of eight characters, followed by data to be populated at creation time beginning in column 11.

Syntax: `INFILE=,DOMAIN=,CLASS=`

Example: Update the instances in the specified text file into:

```
CLASS=INFILE=MYINPUT.TXT,
DOMAIN=SYSTEMX,CLASS=USER
```

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

UPDATE_MGRIDS

This EDMAMS verb updates the specified Manager ID, Manager name, managing Manager ID, and managing Manager name in the DMA prefix. If DOMAIN and CLASS are omitted, the entire file will be updated. All keywords are optional, however at least one keyword other than DOMAIN or CLASS must be specified or a usage error will be displayed to STDERR.

Syntax: `(FILE=),(DOMAIN=),(CLASS=),(MID=),(MMID=),(MNAME=),(MMNAME=)`

Example: Update the Manager IDs, Manager names, managing Manager IDs, and managing Manager names in the PRIMARY file.

```
MID=XXX,MMID=XXX,MNAME=XXX,MMNAME=XXX
```

VERIFY_CLASS

This EDMAMS verb will display class templates as specified to determine if there are any gaps, overlays or other anomalies in the template. The output log will consist of a template entry number, the attribute name, and the attribute's displacement in the heap and the length of the attribute. If an error is found it will be indicated in the appropriate place.

Syntax: `DOMAIN=(ALL/DOMAIN),CLASS=(ALL/CLASS)`

ZRSOURCE_UNMATES

This EDMAMS verb will match instances in the Primary file with the resource data in the Resource file based on the domain and class specified, 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, this is reported in the log as an inconsistency. If PREVIEW=NO is specified, 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. The domain specified is the domain that contains resource data, usually SYSTEMX. If CLASS= is omitted, the default is the ZRSOURCE class. If DEBUG=YES is specified, then all instances are listed as they are verified to have mated or unmated resource data.

Syntax: `PREVIEW=YES/NO,DOMAIN=,(CLASS=),`
`(DEBUG=YES/NO)`

JCL Examples for Running EDMAMS (MVS Only)

The following are JCL examples for running EDMAMS.

JCL Example Number 1

For running EDMAMS on an MVS operating system using the KEYWORD DD statement method.

```
//JOBNAME1 JOB CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//EDMAMS   EXEC PGM=ZEDMAMS,COND=(0,NE),PARM=(ZFILE)
//STEPLIB  DD DSN=YOUR.EDM.LOAD.LIBRARY,DISP=SHR
//         DD DISP=SHR,DSN=SASC.C650.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//PREVIEW  DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//STGRPT   DD SYSOUT=*
//LOG      DD SYSOUT=*
//INPDS    DD DSN=YOUR.INPUT.PDS.(PDSINP),DISP=SHR
//KEYWORD  DD DSN=YOUR.KEYWORD.PDS(KW1),DISP=SHR
//PARMLIB  DD DSN=YOUR.EDM.PARMLIB(STARTUP),DISP=SHR
//PRIMARY  DD DISP=SHR,DSN=YOUR.EDM.PRIMARY.FILE,
//   AMP=('STRNO=10,BUFNI=30,BUFND=100')
//RESOURCE DD DISP=SHR,DSN=YOUR.EDM.RESOURCE.FILE,
//   AMP=('STRNO=10,BUFNI=30,BUFND=30')
//*
```

Note: The KEYWORD DD statement contains the name of a dataset member with EDMAMS KEYWORDS as the following example indicates:

```
*
 VERB=CHANGE_INST_DATA,DOMAIN=SYSTEMA,CLASS=USER,
 FROMDATA="J. FILBERT",TODATA="JOHN B. MCDONALD"
*
 VERB=CHANGE_INS_FIELD,DOMAIN=SYSTEMA,CLASS=USER,PREFIX=DIF,SUFFIX=F8,
 FIELD=NAME,TODATA="H. Kapelcro",PREVIEW=NO
*
 VERB=LIST_INST_DATA,DOMAIN=SYSTEMa,CLASS=USER,INSTANCE=DIFF
*
 VERB=SEARCH_INSTANCES,DOMAIN=SYSTEMA,CLASS=USER,STRING=
 "J. ASTRANIS"
*
 VERB=CHECK_RESOURCES,LISTALL=NO
*
 VERB=COPY_CLASS,FROMDOMA=SYSTEMX,TODOMAIN=SYSTEMC,
 FROMCLAS=USER,REPLACE=yes
*
 VERB=COPY_INSTANCE,FROMDOMA=SYSTEMX,TODOMAIN=SYSTEMA,
 FROMCLAS=USER,FROMINST=DIFF2
*
 VERB=DELETE_DOMAIN,FROMDOMA=SYSTEMC,PREVIEW=NO
*
 VERB=CREATE_INSTANCES,INFILE=INPDS,DOMAIN=SYSTEMA,CLASS=USER
*
 VERB=DELETE_CLASS,DOMAIN=SYSTEMC,CLASS=ZSERVICE
*
 VERB=DELETE_INSTANCE,DOMAIN=SYSTEMA,CLASS=USER,FROMINST=DIFF
*
 VERB=LIST_CONS_VARS,DOMAIN=SYSTEMX,CLASS=ZRSOURCE,INSTANCE=CICS
*
 VERB=LIST_DOMAINS
*
 VERB=LIST_FLAGS,DOMAIN=SYSTEMX,CLASS=ZRSOURCE
*
 VERB=LIST_INSTANCE,DOMAIN=SYSTEMA,CLASS=USER
*
 VERB=LIST_RESOURCES,DOMAIN=SYSTEMX
*
 VERB=LIST_ZRSC_FIELDS,DOMAIN=SYSTEMX,CLASS=ZRSOURCE,
 INSTANCE=EDM
```

JCL Example Number 2

For running EDMAMS on an MVS operating system using the PARM= method.

This method is the least convenient, because the syntax for entering the keywords is very exacting.

```
//JOBNAME1 JOB CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//EDMAMS   EXEC PGM=ZEDMAMS,COND=(0,NE),
//         PARM=('VERB=COPY_CLASS,FROMDOMA=SYSTEMX,FROMCLAS=USER,',
//         'TODOMAIN=SYSTEMB,REPLACE=YES')
//STEPLIB  DD DSN=YOUR.EDM.LOAD.LIBRARY,DISP=SHR
//         DD DISP=SHR,DSN=SASC.C650.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//PREVIEW  DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//STGRPT   DD SYSOUT=*
//LOG      DD SYSOUT=*
//PARMLIB  DD DSN=YOUR.EDM.PARMLIB(STARTUP),DISP=SHR
//PRIMARY  DD DISP=SHR,DSN=YOUR.EDM.PRIMARY.FILE
//RESOURCE DD DISP=SHR,DSN=YOUR.EDM.RESOURCE.FILE
//*
```

JCL Example Number 3

For running EDMAMS on an MVS operating system using the *re-direction* method. This method is convenient when running single EDMAMS utilities, but your PARMIN line is limited to 80 characters with no intervening spaces.

```
//JOBNAME1 JOB CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//EDMAMS   EXEC PGM=ZEDMAMS,COND=(0,NE),
//  PARM='=<//DDN:PARMIN'        REDIRECT INLINE - 80 CHARS MAX
//PARMIN   DD  *
  VERB=LIST_INSTANCE,DOMAIN=SYSTEMX,CLASS=USER
//*
//STEPLIB  DD DSN=YOUR.EDM.LOAD.LIBRARY,DISP=SHR
//         DD DISP=SHR,DSN=SASC.C650.LINKLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//PREVIEW  DD SYSOUT=*
//SYSTERM  DD SYSOUT=*
//STGRPT   DD SYSOUT=*
//LOG      DD SYSOUT=*
//INPDS    DD DSN=YOUR.INPUT.DATA.SET(PDSINP),DISP=SHR
//PARMLIB  DD DSN=YOUR.EDM.PARMLIB(STARTUP),DISP=SHR
//PRIMARY  DD DISP=SHR,DSN=YOUR.EDM.PRIMARY.FILE,
//   AMP=('STRNO=10,BUFNI=30,BUFND=100')
//RESOURCE DD DISP=SHR,DSN=YOUR.EDM.RESOURCE.FILE,
//   AMP=('STRNO=10,BUFNI=30,BUFND=30')
//*
```

Part IV

Optimizing the Manager

7

Performance

This chapter discusses performance aspects for the Manager.

An Overview of Performance Issues

The purpose of this chapter is to discuss system performance issues as they relate to the Manager. The following chapter, *Chapter 8: Troubleshooting*, explores some problem determination issues.

The distinction between the two is that performance issues are associated with enhancing the efficiency of a working system while troubleshooting deals with features, functions and components that are not operating in a satisfactory way. It is useful to note that addressing Performance and Usage Considerations may forestall many of the conditions that require troubleshooting.

The Manager is a multi-platform, multi-processing server framework for:

- Policy Management
- Component Management
- Network Management
- Version Management
- Asset Management
- State Management.

There are many aspects to Manager performance. In addition, there are specific phases of Manager operations. Each phase has different performance characteristics and requirements. Because of these factors, there can be no easy 'cook book' approach to Manager performance.

General Performance and Usage Considerations

There are at least three important Performance and Usage Considerations that must be taken into account before beginning any discussion of Manager 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 should include average performance levels, as well as high and low levels.
- What are the workload parameters?
  These should include average demand, peak load requirements and idle times

You should be familiar with these considerations as they apply to your Manager before undertaking any further performance initiatives.

How this Chapter is Organized

This chapter is organized into three performance topic areas:

- CPU
- Memory
- Network.

CPU

There are minimum CPU requirements specified for each Manager platform at installation. It must be noted, however, that these are minimum values. The real requirements for CPU utilization can only be determined by workload. Workload in this case is typified by the number of resolutions that the Manager can process.

The CPU and Object Resolution

Most of the processor time required by the Manager is used up by the object resolution process. As each Client connects to the Manager an identifier object (ZMASTER) is sent from the Client to the Manager triggering the dynamic construction of an object model for that Client. This process is known as *object resolution*. When trying to determine how many object resolutions can take place in a given time, a relatively simple formula can be applied. It is as follows:

Total Number of Available CPU Seconds

Number of CPU Seconds Required Per User = Z
Total Number of User Resolutions Possible

Total Number of Available CPU Seconds

This value is obtained by multiplying the number of processors (CPUs) by the number of seconds in the connection window of opportunity. The window of opportunity is the timeframe in which your Clients need to be connected in. An example where 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 = 43,200 seconds).

Total Number of CPU Seconds Required Per User

This value is a little more complicated to obtain. Some benchmarks for object resolution speed have been defined by Novadigm running a Manager on an HP/K200 system with 85 MHz processors. We have determined that approximately 860 objects can be resolved in one second. With this benchmark we can make an approximation of how many CPU seconds will be required to resolve a user's object model. By taking the average number of ZRSOURCEs per user (ZRSOURCE is the most common object in a user's model) and multiplying it by three (estimation of how many objects get resolved to end up with a fully resolved ZRSOURCE) we get the average number of objects to be resolved for a user. We then divide that by number of objects the Manager can resolve in a CPU second we get the number of CPU seconds required to resolve the average client's object model. For example let's say the average number of ZRSOURCEs per user in your environment is 1000. We multiply that by 3 to get 3000 objects per user, divided by 860 (the average number of objects resolved per second by the Manager) you get about 3.5 seconds of CPU time required to resolve the average user's model.

Total Number of User Resolution Possible

This value is obtained by dividing the Total Number of Available CPU Seconds by the Total Number of CPU Seconds Required per User. Using our above example 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 User Resolutions Possible of about 12,000.

Note that this calculation by itself 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 and from Clients, the platform's network card capability (for concurrent conversations between the manager and clients), 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 Manager.

Memory

There are two features available that deal with memory usage, content caching, and index caching. Both are established in the MGR_CACHE section of the Manager Settings file.

Content caching refers to the loading of a portion of the Manager Database (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 Manager wide resolution boost.

Index caching is used to keep in memory all names of the instances of the particular cached class. By caching the names of all instances of cached classes, it eliminates the need to read the directory to determine all of the instances that begin with the specified prefix. Index caching makes a significant performance improvement when the generic resolution feature is utilized. Generic resolutions are those which 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 defines the values that determine how much virtual storage is set aside for content cache. The two values are CACHE_SEGMENTS that determine the number of separate memory areas of CACHE_SIZE to be allocated at startup exclusively for content cache use. The product of CACHE_SEGMENTS and CACHE_SIZE is the amount of memory that will not be available for resolution purposes during connection. In the NT environment, the virtual storage available to a single process (i.e., the NT Manager) is up to 2 Giga-bytes

(unless the MS NT Enterprise Server is used which provides up to 3 Giga-bytes of virtual memory per process), but may be constrained by the size of real memory and the size of the page space available. The NT Manager will attempt to use all of the virtual storage available to it and may cause all (up to 2 Giga-bytes) of the defined page space to be in use. The AVERAGE_OBJECT_SIZE in this section should be set to the largest instance size of those classes being cached. This value defaults to 2K if not explicitly specified, and should have a value of the ZRSOURCE instance size that is represented as about 3660 below.

A parameter called ICACHE is available for index caching. This is activated by specifying a value for the ICACHE_SIZE keyword. The easiest means to correctly size this is to take the total of all instances to be cached and multiply by 100 and place the result as the ICACHE value size. ICACHE is of benefit when generic resolution is active, that is, any connection to ZRSOURCE.PREFIX_* which requires the Manager to process all of the potential instances prefixed by the string. While ICACHE is most important for ZRSOURCE, 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 during resolution of each Client for in-storage objects (as differentiated from database classes and instances where the class name may be the same as the in-storage object name as is the case for ZSERVICE and ZRSOURCE). 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 purposes, the contents of MGR_CLASS are processed in the order in which they are

presented, so the first DOMAIN.CLASS is processed completely (i.e., index cache is loaded and content cache is loaded) before the second, and so on.

For each DOMAIN.CLASS (i.e., SYSTEMX.ZRSOURCE) identified in this section, four parameters are specified. The first two control the caching behavior, and the second two are used exclusively for persistent object virtual storage allocation during resolution.

The first of the four parameters allows one to specify if the class template and base instance are to be cached. A value of **Y** is recommended since it is generally necessary to load the class template and base instance and this eliminates a tremendous amount of disk I/O for classes which are commonly involved in resolution.

The second of the four parameters determines if all of the instances of DOMAIN.CLASS are to be loaded at initialization (**Y**) or are to be cached at first reference (**N**). A value of **Y** is recommended for those classes that are most heavily referenced during resolution, such as ZLOCMGR, ZLOCCLNT. For other classes, the recommendation is to allow the first references to cause the instance to be cached. This has two benefits- Manager startup is much faster, and content cache (CACHE_SEGMENTS * CACHE_SIZE) can generally be reduced to much less (~20%) of the total amount of space which might otherwise be required if all of ZRSOURCE were to be cached without seriously impacting resolution performance. The assumption is that those instances that are most frequently referenced will become cached, and those less frequently referenced will be fetched as needed. And since they are cached by resolution reference, it is possible for the manager to adapt after each startup or cache refresh to the then current reference pattern.

The third of the four parameters is important only for persistent instorage objects. These are generally ZSERVICE, ZRSOURCE, ZAUDITS, AND ZAUDITR. The value for heap size represented here is the resolved size and is best determined by examining the actual size of each object (each heap size, not

the file size as seen by a DIR command) using the Client Browser (the Object Editor) tool on a Client that contains a resolved object of the appropriate type. The reason for specifying this is that during the course of resolution, the size of an object may change, and the performance hit from having to expand a persistent object during resolution can be very considerable. Being able to specify this size before resolution finds the largest correct size benefits performance overall. In those cases where the resolved object size is less than the database instance size (i.e., ZRSOURCE) this also simplifies right-sizing virtual storage that would otherwise be lost during the resolution process and cause unnecessary demands on virtual storage during the period of peak usage.

The fourth of the four parameters controls how mane heaps should be allocated on the initial allocation and each subsequent request for object expansion. So if the maximum number of instances in any client under maintenance is 10,000 and this is represented by 10000 ZRSOURCE instances each of 2K in size, if the MGR_CLASS section specification were:

```
SYSTEMX.ZRSOURCE = Y,N,2048,10000
```

Then the first time any Client requested storage for a ZRSOURCE object, an allocation for the full 10000 instances of 2K each would be made to the virtual storage subsystem as a single malloc (2048*10000 ~=20Meg storage request). If all Clients have this requirement, this is the right thing to do. (I'd be inclined to check TASKLIM and the amount of page space available however.frequently). However, if there is a range of files under management ranging from 500 to 10,000 with the larger population being somewhere on the lower end of this range, then a specification which allocates a more likely increment would be recommended. For example, see below.

```
SYSTEMX.ZRSOURCE = Y,N,2048,500
```

The specification above allocates only one megabyte (500 * 2048) of memory initially for the ZRSOURCE object, and if more than 500 heaps are required, it will chain together additional 500 heap chucks of memory until the storage demands

or resolution are satisfied. While this would require 19 secondary allocations for the 10,000-heap Client, the treatment of virtual storage is easier on the system than handling requests for 10 MB of contiguous storage even when it wasn't being used would be.

Network

Bandwidth throttling is a new feature. It was designed and developed to help you better use your network resources while running the Manager. Bandwidth throttling refers to the reservation of a certain percentage of the available TCP/IP bandwidth for use by other processes on the desktop.

Bandwidth throttling is configured in the Manager using the SEND_THROTTLE setting of the MGR_TIMEOUT section of the Manager Settings file. This setting specifies the number of milliseconds that the Manager waits 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 Client's ZMASTER object that also have an impact on bandwidth throttling, ZBWMGR, ZBWMAX and ZBWPCT. ZBWMGR is the switch to determine if the Manager will be controlling the bandwidth (YES indicates that the manager will be managing the bandwidth). ZBWMAX is the maximum speed (in bytes per second) of the sends (0 -4 GB). ZBWPCT is the percentage of the maximum to use (0 - 100).

Please note that ZBWMGR, ZBWMAX and ZBWPCT values will override the SEND_THROTTLE setting.

8

Troubleshooting the Manager

This chapter discusses basic troubleshooting issues and provides three operational scenarios, their associated problems, possible causes, and recommended actions.

Troubleshooting Issues

The purpose of this chapter is to explore some problem determination issues as they relate to the Manager. *Chapter 7: Performance* discusses system performance issues.

The distinction between the two is that performance issues are associated with enhancing the efficiency of a working system while troubleshooting deals with features, functions and components that are not operating in a satisfactory way. It is useful to note that addressing Performance and Usage Considerations may forestall many of the conditions that require troubleshooting.

General Troubleshooting Considerations

There are several things that you should consider before attempting to troubleshoot a specific problem:

- **What specifically is the problem?**
  In some cases there are similar symptoms for different problems. For example, if a Client resolution does not complete due to timing out, the timeout could be based on either a Manager value or a Client setting.
- **At what point did the problem occur?**
  If you can determine at what point a process failed, you may be able to eliminate prior steps.
- **How is the problem reflected in the Manager log?**
  You can use the Manager log and the search tools provided by Novadigm Customer Support to isolate exactly where the problem is reported in the Manager log.
- **Are there external causes for the problem?**
  You may be able to determine if a cause unrelated to the Manager is responsible for the problem.

How this Chapter is Organized

This chapter is organized into three general scenarios:

- The Manager does not start.
- The Manager does not process tasks as expected.
- The Manager does not respond.

Each scenario contains individual conditions, possible causes and recommended actions.

The Manager Does Not Start

Condition	Possible Cause	Recommended Action
The Manager does not start.	The Database did not verify correctly.	Reset DB_VERIFY settings in the Manager Settings file.
	There is insufficient disk space.	Free up sufficient disk space.
	The Manager Settings file is not processed.	Ensure that the Manager Settings file is in the same directory as ztoptask.
The Manager does not start (NT specific).	The Manager was installed under a User Account that is not part of the NT Admin Group.	Reinstall the Manager under a User Account that is part of the NT Admin Group.
The Manager does not appear in the NT Services List (NT specific).	The Manager was not installed as a service.	Reinstall the Manager as a service.
The Manager does not start automatically when rebooted (NT specific).	The Manager Service in the NT Services List is set to Manual.	Set the Manager Service to Automatic. Then, reboot the Manager.

The Manager Does Not Process Tasks as Expected

There are two aspects to this scenario: either the Manager does not perform the process correctly or that the data received is not correct.

Condition	Possible Cause	Recommended Action
No Console or Admin functions.	Incorrect MGR_ACCESS values.	Change MGR_ACCESS values.
Manager tasks not starting.	Manager tasks not listed in MGR_ATTACH_LIST section.	Add more slots in MGR_ATTACH_LIST section.
Manager does not accept Client tasks.	No communications Manager tasks specified in MGR_ATTACH_LIST section.	Specify appropriate communications Manager tasks in MGR_ATTACH_LIST section.
Manager does not accept additional Client tasks.	Setting in MGR_TASKLIM is too low.	Increase MGR_TASKLIM setting.
Too much processor time required to load commonly used classes.	Classes not listed in MGR_CLASS section.	Add classes to MGR_CLASS section.
Methods not executing properly.	Time-out setting in MGR_METHODS section is too low.	Increase time-out setting in MGR_METHODS section.
Too many messages in Manager log.	Tracing is set to YES.	Set tracing to **NO** for unnecessary trace settings.

Condition	Possible Cause	Recommended Action
Manager Log is slow to respond.	FLUSH_SIZE is set too low.	Raise FLUSH_SIZE in MGR_LOG section.
Lost portions of Manager log.	Log has been reused.	Change THRESHOLD setting in MGR_LOG section to a positive value.

The Manager Does Not Respond

Condition	Possible Cause	Recommended Action
The Manager does not respond to communications requests.	Communication Manager tasks are not enabled.	Add appropriate communications Manager tasks in MGR_ATTACH_ LIST section.
The Manager does not respond to Client task requests.	Other Clients have a RETRY value that is too low.	Raise RETRY value to at least 1.

Part V

Special Purpose Managers

9

Multi-Mode Manager

This chapter describes the usage of a single Manager for both EDM and Radia Clients.

> **Note:** You must have proper licensing for both EDM and Radia to operate the Multi-Mode Manager.

Introduction

The Manager is fully capable of processing EDM, Radia and other transactions (clients) during a single Manager session. There are some features in the Manager that apply only to Radia as well as some features that apply only to EDM. If EDM is seen as a multi-service batch type of environment, then Radia represents single-service processing. Since Radia is more oriented to interactive processing, the database for it contains some additional features to allow for easier and friendlier usage. The EDM configurations are more oriented to complicated, multidimensional processes that fully define the states of desktops and/or servers.

Single Manager Approach

The Manager can work in a multi-function mode serving two types of clients — e.g., EDM and Radia. The EDM and Radia clients have different settings that direct the paths of resolution that are stored in the database. Generally, even the definitions may be shared as long as the object resolution start point is defined separately for each type of client.

In prior versions of the Manager, the resolution's start point was defined in the MGR_STARTUP section of the Profile file. The START_DOMAIN and START_CLASS defined the start domain and class for any inbound object received by the Manager. The multi- mode manager uses EDM_STARTUP section to define start domain and start class for all the EDM clients and RADIA_STARTUP section for all the Radia clients. This way the resolution start point may be different depending on the type of the connection.

Old Version

```
[MGR_STARTUP]
MANAGER_TYPE      = DISTRIBUTED
TCP PORT= 1955
SHOW_VERINFO      = NO
MGR_NAME          = VFH

  ...
/'Resolution startpoints:'/
START DOMAIN      = ZSYSTEM
START CLASS       = ZPROCESS

/'Resolution default domains and classes:'/
DOMAIN            = ZSYSTEM
CLASS             = ZPROCESS


MGR ACCESS
ADMIN = deny
CONSOLE = deny
   ...
```

Multi-mode Version

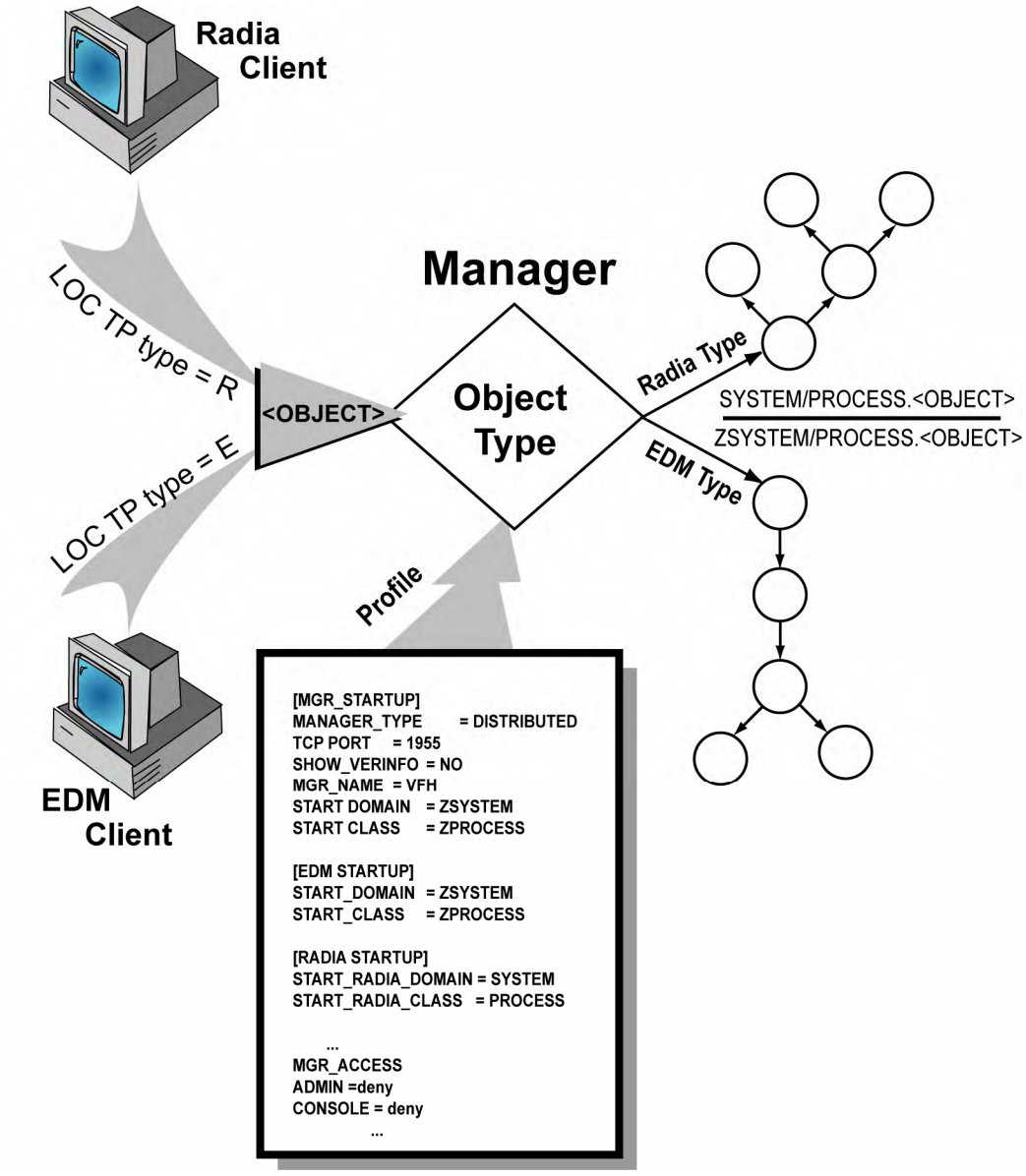
```
[MGR_STARTUP]
MANAGER_TYPE      = DISTRIBUTED
TCP PORT= 1955
SHOW_VERINFO      = NO
MGR_NAME          = VFH
START DOMAIN      = ZSYSTEM
START CLASS       = ZPROCESS

[EDM STARTUP]
START_DOMAIN      = ZSYSTEM
START_CLASS       = ZPROCESS

[RADIA STARTUP]
START_RADIA_DOMAIN = SYSTEM
START_RADIA_CLASS  = PROCESS

            ...
MGR_ACCESS
ADMIN =deny
CONSOLE = deny
            ...
```

In the example above, the standard start point of the resolution is:

ZSYSTEM.ZPROCESS.<OBJECT_NAME> is used for non-Radia clients and

SYSTEM.PROCESS.<OBJECT_NAME> for Radia clients.

In this case the multi-mode Manager's processing may look like:



How does the Manager distinguish between EDM and Radia Clients? The type of the client is defined in the initial handshake transactions called EDMLOCTP that every client sends to the manager immediately after the connection is established. The EDMLOCTP contains the client product field that sets the client type (E or R). The Manager Settings define the start point for each type of resolution. Then for any inbound object, the resolution starts in the specified domain and class depending on the type of the client.

10

HTTP, SSL and Multicast Managers

This chapter describes the configuration and usage of the HTTP Manager, the SSL Manager, and the Multicast Manager.

> **Note:** You must have proper licensing to operate the HTTP Manager, the SSL Manager or the Multicast Manager.

Introduction

The Manager is a powerful resource for the storage and dissemination of information. We have increased the versatility of the Manager by introducing several new capabilities. These changes increase security in the information exchange, allow you to configure the Manager as a Multicast initiator, and enable the Manager to act as a Web-server.

The new features are the HTTP Manager, Multicast Manager, and SSL Manager. To enable these new Manager features, a new task must be entered into the MGR_ATTACH_LIST section. Instructions on how to do this, as well as database specifications for each Manager utility, are covered in the sections that follow.

HTTP Manager

The HTTP protocol has been added in order to enable the Manager to act as a web-server. A new section, MGR_HTTP, has been added to the Manager startup file (edmprof.dat) to support HTTP. The new Manager task is called zhttpmgr.

Example

```
[MGR_HTTP]
HTTP_PORT = 3466
RESPONSE_HTML_FILE = /test/reject.html
```

In the example above, the HTTP_PORT command specifies the port to which the zhttpmgr should listen for connections from clients. The default port number is 80. The RESPONSE_HTML_FILE command specifies the name of the HTML file that will be sent to unauthorized client web-browsers should they accidentally connect to the zhttpmgr. If a response file has not been specified, the zhttpmgr will send back a status code of 400 (HTTP Bad Request).

How to Configure an HTTP Manager

In the MGR_ATTACH_LIST section, add the HTTP Manager:

```
CMD_LINE=(zhttpmgr) RESTART=YES
```

FOR UNIX ONLY:

1. Start up the Manager (ztoptask). You should see the zhttpmgr as one of the tasks in the process lists.

2. Using a client Web-browser, connect to the zhttpmgr, as below:

   http://208.244.225.25:3466

It should display the RESPONSE_HTML_FILE message specified in the MGR_HTTP section, or the "Bad Request" status code (400), if no response file has been specified.

VIRTUAL IP ADDRESSES IN UNIX

With virtual IP addresses, a machine with a single NIC card can have multiple IP addresses. This is especially useful when multiple server programs have to listen on the same port. For example, our zhttpmgr and a standard web server both listening on port 80. To resolve the port conflicts, machines are set up with Virtual IP addresses, whereby multiple IP addresses are assigned to a machine.

The ztcpmgr and zhttpmgr have been changed so that they support Virtual IP addresses. They accept the IP address and the port number on the command line, as in the following:

```
CMD_LINE=(zhttpmgr addr=1.1.1.94,port=4438)
RESTART=YES
CMD_LINE=(ztcpmgr addr=1.1.1.10,port=4438)
RESTART=YES
```

You would have to modify your entry in the [MGR_ATTACH_LIST] section of .edmprof. If the address is not specified, it uses the machine address.

To configure Virtual IP addresses, use the ifconfig command. This command has to be run under root privileges, as shown in the example below:

Note: *hme0* is the device name. This can be *le0* on other systems. You can get the proper device name to use by typing:
`ifconfig -a`

```
/usr/sbin/ifconfig hme0:1 inet 208.244.225.163
netmask 0xffffff00 broadcast +
/usr/sbin/ifconfig hme0:2 inet 208.244.225.175
netmask 0xffffff00 broadcast +

/usr/sbin/ifconfig hme0:1 up
/usr/sbin/ifconfig hme0:2 up
```

STARTING THE MANAGER WITH ROOT PRIVILEGES ON UNIX SYSTEMS

When a Radia Client has to connect to the Manager using an intermediary Web-server, it uses TCP/IP tunneling. The TCP/IP

tunneling works by blindly funneling requests between the Client and the Manager. It is important that the Manager's port number be properly set.

For using ports below 1024, which are reserved ports, you would have to start the Manager as root, either using the rc scripts or logging in as root. In addition, you would have to set the LD_LIBRARY_PATH to the manager-executable directory before you run ztoptask, as in the example below:

```
LD_LIBRARY_PATH=/mgrbuild/V4.11/exe:/usr/lib:lib
export LD_LIBRARY_PATH
./ztoptask
```

Note: For Microsoft proxy servers, the port number has to be 443, which is the secure HTTP port. This requires that the Manager run on port 443, so that the proxy can contact it; otherwise, the proxy won't let the clients establish the tunnel.

Multicast Manager

Multicasting enables you to program the Manager to transmit files to multiple Clients, simultaneously, by notifying them to attend at a predetermined time. An advantage of this method of broadcasting is that each Client takes from the one-time transmission, only what it requires, or is authorized to receive.

Note: All the parameters needed for multicasting are stored in the Database. The zmcaster does an object resolution in order to get the multicast parameters.

The Multicast Manager is comprised of the main multicast manager (zmcstmgr), and several collector/multicaster tasks (zmcaster). The zmcstmgr task waits on a queue for a Client task (ztermtsk) to deposit a filename. This filename contains the list of files that a client needs in order to be multicasted. The zmcstmgr passes the filename over to the zmcaster thread, which builds a list of files, and waits for further requests from the Client. At a predetermined time, the zmcaster thread stops the collection process and starts multicasting the list of files collected from the clients.

Manager Setup

Modify the Manager startup file.

1. In the [MGR_ATTACH_LIST] section, specify zmcstmgr as one of the tasks.

2. Create a new section called [MGR_MULTICAST] and specify:

   - the number of concurrent multicast tasks (zmcaster) to start on the Manager, with each task catering to a different audience (multicast host group)
   - the working directory for multicast (for client tasks to create a list of files needed by the client)

```
[MGR_MULTICAST]
NUMBER_OF_MULTICAST_TASKS = 5
MULTICAST_WORK_DIR = c:\temp\multicast\
```

3 Create a new class in the EDM Database that will contain the multicast instances. Each multicast instance can store the information about a multicast host group. By connecting or disconnecting users or workgroups to or from one of these instances, the administrator can enable or disable multicasting.

Note: We recommend you add these values to the base instance so that all the instances inherit these values.

There are certain variables needed by the EDM Client in order to receive the files from the Manager. These variables need to have the global flag turned on so they can flow back to the ZMASTER object on the Client. The DOMAIN, CLASS, and INSTANCE variables should always contain the values described in this document.

The important variables in the multicast class are presented in the following table.

Multicast Class Variables

Variable	Flags	Description
ADDRESS	Global	Specifies a valid multicast/broadcast address
CGMTDATE	Global	The collection start date specified in YYYYMMDD format
CGMTTIME	Global	The collection start time specified in HH:MM:SS format
CLASS	Resolve, Substitute, Global	Specifies class name
DELAYBP		Delay, in milliseconds, between packets
DELAYFP		Delay, in milliseconds, after the first packet is sent
DOMAIN	Resolve, Substitute, Global	Specifies domain name

Variable	Flags	Description
EGMTDATE	Global	The collection end date specified in YYYYMMDD format
EGMTTIME	Global	The collection end time specified in HH:MM:SS format.
INSTANCE	Resolve, Substitute, Global	Specified instance name
MCAST	Global	A flag to indicate if multicast is enabled (Y) or disabled (N)
MGMTDATE	Global	The multicast date specified in YYYYMMDD format
MGMTTIME	Global	The multicast time specified in HH:MM:SS format.
MODE	Global	Broadcast (B) or Multicast (M)
PORT	Global	UDP port
RESENDS		The number of resends
STORE	Global	Number of packets, to store as backup, that would be used for re-sends
TIMEOUT	Global	Default time-out in minutes (how long the receiver should wait).
TTL		The number of router hops

Note: If the MODE variable is set to multicast, then the ADDRESS should be specified in the Internet dotted-decimal format, and should be between 224.0.0.255 and 239.255.255.255.

The CGMTDATE, CGMTTIME, EGMTDATE, EGMTTIME, MGMTDATE and MGMTTIME are specified in GMT format, therefore, before you add these values to the instance, you should consider the offset from GMT, which varies from one time zone to another. The Client should send up multicast requests to the Manager if it is between the collection start time and the collection end time.

The time-out specified in the instance variable cannot be calculated until all the file requests have been collected. Once this happens, the Multicast Manager calculates the total multicast

time based on the number of packets to send out, the delay after the first packet, and the delay between packets. This time is sent in a special session information packet to all the clients, before the first file is transmitted. This is done so that each client can calculate the time-out by adding network latency to the time-out in the information packet. If the Client fails to receive the session-information packet, it uses the default time-out specified in the instance.

SSL Manager

Enabling SSL in Manager and Client

Secure Socket Layer (SSL) capability has been added to increase security in the exchange of information with the Manager. It is a communication DLL (shared library), similar to our TCP/IP DLL. The SSL protocol is actually an extension of our existing TCP/IP DLL, and is called nvdtcps.dll. SSL is used by the Client and the Manager, and is implemented using the public domain OPENSSL.

Note: The only occasion when a Client would require a Client certificate, is when the Manager needs to verify a Client. Typically, the Client does a self-verification.

To use SSL, the Client and the Manager each need a Certificate of Authority (CA certificate). These certificates enable the Manager-Client handshake, so the two can communicate. The Manager also needs a Server certificate, whereas an optional Client certificate is available for the Client.

To enable SSL on the Manager, add a new task to the MGR_ATTACH_LIST section, e.g.,

```
[MGR_ATTACH_LIST]
CMD_LINE=(zsslmgr) RESTART=YES
```

Manager Changes

With the SSL Manager, there are two new components: nvdtcps.dll, which is the SSL DLL, and zsslmgr, shown above, which is the SSL Manager Task. In addition, a new section has been added to the Manager Settings file to store the SSL parameters. The section name is MGR_SSL, and it contains the following keywords:

MGR_SSL Keywords

Keyword	Function
CERTIFICATE_FILE	Use to set the Manager, or the Server, certificate. The value should be a valid and existing certificate file. An expired or corrupt certificate would prevent the SSL Manager from starting up.
KEY_FILE	Use to set the private key. The value should be a valid and existing key file. Usually, the private key is stored in the same file as the Server certificate. If so, you don't have to specify any value for the key file.
KEY_PASSWORD	Use to specify the password used to encrypt the private key, the one specified in KEY_FILE. This is typically needed if the private key is encoded. If the private key is not encoded, you don't need to specify the password.
DH_FILE	Use to set the **D**iffie-**H**ellman parameters. These parameters are used to exchange keys, using the ephemeral Diffie-Hellman method, during the SSL handshake.
SSL_PORT	Use to set the port number to which the SSL Manager should listen for Client Connects.
CA_FILE	Use to set the Certificate Authority's certificate. The SSL Manager needs a CA certificate to start up. An expired or corrupt CA certificate would prevent the SSL Manager from starting up.
VERIFY_CLIENT	Use to specify if the Manager should verify the Client by requesting a certificate. If the Client doesn't have a certificate, the connection would be dropped.

Example

```
[MGR_SSL]
CERTIFICATE_FILE = w:\openssl\ms\srvcert.pem
KEY_FILE = w:\openssl\ms\srvprvk.pem
KEY_PASSWORD=violin
DH_FILE = w:\openssl\ms\dhparams.pem
SSL_PORT = 3456
CA_FILE = w:\openssl\ms\cacert.pem
VERIFY_CLIENT=Y
```

Client Changes

To enable SSL on the Client, you need the following parameters in the ZMASTER object.

Note: The Client should have at least a CAFILE specified in order to use SSL. It's optional for the Client to have a certificate and private key, as these are needed only when the Manager or Server needs to verify the Client.

ZMASTER Object Parameters

Parameter	Function
CAFILE	Use to specify the Certificate Authority certificate file.
CERTFILE	Use to specify the Client certificate file.
DHFILE	Use to specify the Diffie-Hellman parameters file.
KEYFILE	Use to specify the private key-file.
ZDEVICEN	Use to specify the device number for SSL (094).

Radia-Specific Changes

The Certificate Authority certificates are stored in the CACERTIFICATES directory, under IDMSYS. The Client should store all the CA certificates in this directory. If there are multiple CAs, they should be stored with unique names. The default certificate file is CACERT.PEM. Client certificates are stored in the CERTIFICATES directory under IDMSYS.

In Radia, most of the configuration is done using the EMBED tags in an HTML file. The following is a list of SSL-related embed tags.

```
sslmanager=sun-dbcs
sslport=443
```

The sslmanager tag is used to specify the IP address of the Manager, and the sslport is used to specify the port number to which the sslmanager listens for Client Connects. Specifying these parameters enables SSL for all the Client Connects from the HTML page, except for the Self-Maintenance. Self-Maintenance is done without SSL. The embed tag's resolutionmanager and resolutionport are used for Self-Maintenance. The Self-Maintenance can be used to download the CA certificate from the Manager.

The Radia Dispatcher (radpinit.exe) sets up the proper variables in the ZMASTER object using the values from the embed tags for enabling SSL.

The NVDTCPS.DLL can also be used to connect to the Manager using a proxy server. The proxy server is specified using the proxy embed tag in the HTML file. This causes the DLL to create a secure tunnel, via the proxy, to the Manager. When connecting using a proxy, the Manager would have to listen on port 443, the standard SSL port.

A

Getting Help

This appendix presents you with the information you need to contact and get help from Novadigm Technical Support. It covers the variety of methods offered to communicate with Technical Support, and details on what information you should have available prior to contacting them.

In addition to contact information, this appendix provides diagnostic information, maintenance notes (with *fixes* and *alerts*), and information regarding our documentation. Each of these is a valuable resource, and should be consulted prior to contacting Technical Support.

Contacting Tech Support

The following table presents the various methods by which you can contact the Technical Support department.

Support Type	Access Method
Telephone support	(201) 512-7800
e-mail submission of initial call	You must submit the required template as text of an e-mail to csmail@novadigm.com. Do *not* send it as an attachment.
e-mail correspondence	support@novadigm.com
Internet/Web access	http://techsupport.novadigm.com
FTP	ftp.novadigm.com
BBS	(201) 512-1316

Telephone Support

Note: See *Problem Reporting* on page 233 for a list of the information that the Technical Support representative will require.

If all support lines are busy, your call will be answered by voice-mail. Please leave a message and your call will be returned within one hour. A support staff member will take all initial calls to establish the required information. The aim of this initial contact is to:

- Gather necessary information regarding your site configuration and EDM components.
- Obtain detailed description on the problem.
- Assign a call tracking number, or obtain an already existing tracking number.

E-mail Support

Initial Call

The CSCALLS#.TXT template is available to enable submission of an *initial* problem directly into the Novadigm Technical Support problem tracking system. You will automatically receive the assigned tracking number by return e-mail if the appropriate information is complete and it contains your e-mail address.

Instructions on the template are available in the CSCALLS#.DOC file. These files are located within the TEMPLATE.ZIP file on the FTP/BBS in the EDMUTILS library. If you have any questions about using the template, call Technical Support.

E-mail Correspondence

E-mail correspondence can be sent to Technical Support via the support e-mail address, support@novadigm.com. Reference the assigned tracking number in your e-mail subject.

FTP/BBS/WEB Support

One standard sign-on ID and password is available for each customer. Any FTP software can be used to access this site. It can also be accessed through the web, at http://ftp.novadigm.com/library. Customers are granted access to libraries. These libraries will include information on all platforms for which the customer is licensed to use the Novadigm product, as well as three additional libraries that are customer specific. For example, a customer with an ID of ABCINC would see the following libraries:

- **ABCINC**
  This is the *log* library. The customer has the ability to upload and download files to this library.
- **ABCINC2**
  The *fix* library always has the number "2" appended to the name. The customer only has the ability to download files from this library. Any fixes provided to the customer will be placed in this library.
- **ABCINC3**
  All license strings are filed in the *license* library with the number "3" appended to the name. Permissions for the license library are identical to those of the fix library.

Problem Reporting

Before contacting Technical Support, consult the available EDM documentation for review of your questions or problems. On the FTP/BBS site, published EDM documentation is available in the EDM_PDF library. Also, a comprehensive problem-reporting procedure document is available on the FTP/BBS site in the CS_INFO library.

The following site/user information *will be requested* by Technical Support.

User Information

Please have this information available prior to calling:

- Company Name
- Time Zone
- Customer Contact Name
- Customer Callback information (phone number, e-mail address, fax number, and/or pager number [if applicable])
- Client operating system and version level
- Manager operating system and version level
- EDM Client version
- EDM Manager version
- Level of EDM maintenance or patches applied (if unknown run Clientmt)
- Protocols being used
- Failing EDM component
- Failing EDM module
- Error messages received on screen or in logs

- EDM function in use at time of error

Problem Description

- Can you recreate the problem? If yes, please note the steps to recreate the problem.
- Describe in detail what you were attempting to accomplish when problem occurred.

Documentation

Electronic access to EDM documentation is available in two areas. Consult the table below to locate these files.

Documentation	Location
Published EDM documentation and manuals	EDM_PDF library on the FTP/BBS site, and http://techsupport.novadigm.com.document/EDM/index.html
Technical Support Notes, FAQ's, and Tips and Tricks	url: http://techsupport.novadigm.com

Diagnostic Information

Problems that can be recreated require logs set at maximum logging levels to assist in problem determination. Technical Support will require you to submit Client logs, local EDM objects, and Manager logs. When communicating a problem description to Technical Support, give precise steps used to reproduce the problem and, if available, provide all materials required to reproduce the problem.

Setting Trace Levels

ZTRACEL

The ZTRACEL value of ZMASTER should be set to "999" before running the Client Connect. Depending on your environment, you may need to set the USER INSTANCE value in the database to preserve this setting.

ZTRACE

The ZTRACE value on the desktop ZMASTER should be set to "Y". This will cause tracing of each data-buffer transferred from the Client to the Manager. Depending on your environment, you may need to set the USER INSTANCE value in the database to preserve this setting.

Logs and Objects

EDMLIB Contents

All *.edm, *.log, (and if Packager is in use, *.dat) files should be collected for subsequent analysis and included in the files sent to Technical Support.

Manager Logs

The Manager log should display the beginning of the log so those module levels currently executing are visualized.

Additional Diagnostic Tools

Utilities are available to assist in diagnosing your problems, and are located on Novadigm's FTP and Technical Support Web site.

EDMSTATE

EDMSTATE is designed to give an overview of the current state of the EDM Client. It is available with EDM version 3.1 and higher. EDMSTATE reports information about the following:

- Object statistics for ZMASTER, ZSERVICE and ZRSOURCE.
- Environment settings
- Emulator settings
- Duplicate resources
- ZSERVICE/ZRSOURCE instances and status codes (includes Text version of status code)
- Service Totals
- EDM Timer schedule information
- Elapsed time values for available log files

CLIENTMT

The EDM Client Maintenance process exists for limited client platforms. It is designed to provide Novadigm Technical Support with information on your current working set of code for comparison with the latest EDM release. This helps to prevent conflicts or back-leveling of any applied fixes.

MODLIST

MODLIST.EXT is an extended batch that is available in the EDM Manager for the MVS environment. It is executed from your EDMSYS directory. MODLIST.EXT creates an EDM object, RCONMODL.EDM, which contains information pertaining to currently loaded manager modules. MODLIST displays the version level, assembly date and time, load address, and length of each loaded EDM Manager module.

EDM Maintenance

EDM maintenance is in the form of HIPER fixes or Service Packs, and is available through the Technical Support FTP/BBS and Web site.

HIPER Fixes

The HIPER (**Hi**gh **P**rofile **E**rror **R**esolution*)* fixes are high visibility fixes identified after a GA release and prior to the next release. HIPER fixes are available only for EDM Versions 3.2.1 and earlier. The Version 4.0 maintenance is referred to as Service Packs.

The HIPER.ZIP or Hiper.TAR file contains a grouping of individual fixes and a Hiper.txt file, which provide details on the fixes. Not all fixes are identified as a HIPER. It is our recommendation that all HIPER fixes be applied.

Note: Unix HIPER fixes are provided as TAR files.

Naming Convention: OS_Hi##.TXT,

- OS = an operating system acronym
- Hi = identifies as HIPER
- ## = the incremental version number of the HIPER text

The HIPERs, as well as the naming of the fix index (OS_Hi##.TXT) file, are incremental. Therefore, when a fix is added, the HIPER and index are incremented. HIPER fixes *must* be applied on top of a designated level of EDM. This will be detailed in the first line of the OS_Hi##.TXT.

Fixes

Fixes are released to customers based upon specific problems. They are placed within customer-specific libraries on the FTP/BBS site. It is recommended that the CLIENTMT or

MODLIST always be run prior to applying maintenance. If a fix is made available to all customers, it is identified as a HIPER fix.

Service Packs

Service Packs are only released for EDM V4.0 and higher. They resolve all known issues discovered since the previous release, including HIPERs and regular fixes. The Service Pack compressed file contains a grouping of individual fixes and a readme.txt file, which provide details on the fixes. The Service Pack is incremental, therefore, all fixes in the previous Service Pack will be included.

Naming Convention: OS4_S##C.ZIP, where

- OS = an operating system acronym
- 4 = identifies version as a V4.0
- S = identifies as Service Pack
- ## = the incremental version number of the Service Pack
- C = identifies the Component, either Client or Administrator

Example: NT4_S02C.ZIP identifies the file as NT, Version 4, Service Pack 2, Client.

Note: Unix HIPER fixes are provided as TAR files.

EDM Alerts

An EDMALERT is a notification on a bug, behavior or problem that runs any risk of corrupting the system or database. It would be experienced across a wide range of customers, not specific to one customer's individual architecture or configuration but might involve a consistent software/hardware portion of any enterprise.

This information is sent via e-mail to the specific customers affected, and is also available through the Technical Support Web site.

Although Alerts were designed with EDM fixes in mind, they may occasionally be used as an effective emergency communication tool to the EDM customer base.

FTP/BBS Libraries and Sign-on Maintenance Library Naming Convention

The FTP/BBS Library naming convention is subject to change.

XX ABC Z MT where:

- XX = Level EDM maintenance. Base release version, e.g., 3.2 or 3.1.
- ABC = The operating system 3 letter designation, see table.
- Z = (C) client, (S) stager, (M) manager

  Note: Admin found with Client.

- MT = maintenance

For example, maintenance for MVS Manager version 3.X can be found in 31MVSMMT. In this example, XX = 31, ABC = MVS, Z = M, MT =MT.

EDM Components and Available Operating Systems 3-Letter Designation

Operating System	Client	Admin	Stager	Manager
MVS				MVS
Windows 3.X	WIN	WIN		
Windows 95	W95	W95		
Windows NT	NT	NT	NT	NT
Windows 32-bit (95 & NT)	W32	W32		
OS/2	OS2	OS2	OS2	
Novell NetWare	NVL		NVL	
Solaris	SOL	SOL	SOL	SOL
HPUNIX	HP	HP	HP	HP
GIS/NCR	NCR	NCR	NCR	NCR
AIX	AIX	AIX	AIX	AIX
SUNOS	SUN			
UnixWare	UXW			
Macintosh	MAC	MAC		
DOS	DOS			

Maintenance Library Content: Visible Libraries

Maintenance libraries, those operating systems for which a customer is licensed, will be available on the FTP/BBS. If you are denied access to a library, e.g., a particular operating system is not displayed to you on the FTP/BBS and your license agreement indicates that it should be, please call Technical Support.

Each Novadigm customer receives two unique company-specific libraries. One write-access for sending information to Novadigm, the other read-access for pulling fixes from Novadigm.

Additional customer public libraries include:

- EDMALERT
- EDM_PDF
- EDMUTILS
- MAIN

The MAIN library contains the GUI Worldgroup Client ver 2.02 BBS software in use at Novadigm. You can download it if necessary.

Sending Files to Technical Support

Send Compressed File to Technical Support at Novadigm

Note: If you do not have or know your ID and password, call Technical Support for assignment.

Compress, or tar, the files into one *.zip, *.trz or *.tar file and reference the *tracking number* you were given by Novadigm Technical Support by naming the file to include the tracking number, preceded by L, (for example, LXXXXXA.*, [call number represented by XXXXX]). Follow it with an A for an alphabetic sequence for additional files submitted on the same call number. Place the file onto the Novadigm FTP (ftp.novadigm.com) or BBS ((201) 512-1316) site under your company-specific library. You will not be able to overwrite existing files.

Notify Technical Support of Sent Files

You must contact Technical Support by phone at (201) 512-7800 or e-mail support@novadigm.com, notifying them that you have placed a file onto the site. It would be beneficial if you included

a .TXT file, explaining in detail the problem you are experiencing and your environment. This is especially useful if you are located in a time zone greater than 8 hours from EST, and your contact with support is limited to e-mail or voice-mail.

B

Manager Methods

This appendix is a reference for Manager methods. For information on configuring and using Manager Methods, see *Chapter 2: Managing Processing.*

Manager Methods

Method Name	Alias Name	Description
EDMMAILQ	NEW	Deposits e-mail in the mail queue so it can be sent to a remote system user.
EDMMALLO		Allocates an external data set (MVS only).
EDMMCACH	NEW	Refreshes or disables cache.
EDMMCMPR	ZOBJCMPR	Compresses an in-storage object.
EDMMCOPY	ZOBJCOPY	Copies an in-storage object.
EDMMDCLA	ZDCLASS	Deletes a class from the database.
EDMMDALO	NEW	De-allocates an external data set (MVS only).
EDMMDB	NEW	Locks and unlocks the database against all components except DMA and Console.
EDMMDELI	ZOBJDELI	Deletes and instance from an in-storage object.
EDMMDELV	ZOBJDELV	Deletes a variable from all instances of an in-storage object.
EDMMDINS	ZDELINS	Deletes an instance or instances from within a database class.
EDMMDOBJ	ZDELOBJS	Deletes an in-storage object.
EDMMDPRO	ZDELPROF	Deletes an object in the PROFILE database.
EDMMEXIS	ZEXIST	Verifies the existence of a given class or instance in the database.
EDMMGPRO	ZGETPROF	Creates an in-storage object from the PROFILE database.
EDMMNFYT	ZNOTIFY	Executes a TCP/IP notification on a client.

Method Name	Alias Name	Description
EDMMOLOG	ZVARLOG	Displays the contents of an in-storage object.
EDMMPHIS	ZPUTHIST	Puts an in-storage object into the HISTORY database.
EDMMPPRO	ZPUTPROF	Puts an in-storage object into the PROFILE database.
EDMMPROM	ZPROMANY	Adds or updates an instance to the database.
EDMMPUSH	NEW	Puts an inbound object into a Notify queue.
EDMMRESO	ZSIMRESO	Resolves specified objects.
EDMMRPRO	NEW	Adds, updates, or deletes instances in PRIMARY database based on the variables of an in-storage object.
EDMMSIGN	EDMSIGN	Authenticates users against the database.
EDMMSGNR	EDMSIGNR	Authenticates users against external security systems.
EDMMSORT	ZOBJSORT	Sorts instances, by stems, of in-storage objects.
EDMMTUCH	ZTOUCH	Updates the date/time stamp of an instance.
EDMMULOG		Used to write to the user log file.
EDMMVDEL	ZVARDEL	Deletes all in-storage objects.
EDMMVGBL	ZVARGBL	Migrates values from one in-storage object to another and deletes the source object.
EDMMXREF	ZXREF	Cross-references class and instance usage during the object resolution process.

EDMMAILQ

Method Name EDMMAILQ

Description Deposits e-mail in the mail queue (outbox). The [MGR_SMTP_MAIL] section must be added to the Manager PROFILE file for this method to execute correctly.

Alias Name New

Parameters `FROM,MESSAGE,TO,ATTACH,MESGFILE, SUBJECT.`

Note: Arguments are expected to be in the `KEYWORD=VALUE` format delimited by commas.

Parameter Name	Description	
ATTACH	Optional.	Specifies attachment files. Multiple attachments can be listed by using a semi-colon (;) as the delimiter between each attachment (c:\config.sys;c:\autoexec.bat). Attachments are sent using MIME. Attachments can be in binary.
FROM	Required.	The sender's address (user1@company1.com).
MESGFILE	Optional.	Specifies the file that contains the message. Is used in place of the MESSAGE parameter if the message is greater than 255 characters.
MESSAGE	Required.	Specifies a brief message (limited to 255 characters).
SUBJECT	Optional.	Specifies the subject of the e-mail.
TO	Required.	Specifies the e-mail recipients. Multiple users can be listed by using a semi-colon (;) as the delimiter between each recipient. (user1@company1.com;user2@company b.com).

Parameters are used as keywords and are not case sensitive.

Example

In the example below mail is sent from *user1@company1.com* to *user2@company.com* with a brief message.

```
EDMMAILQ
from=user1@company.com,to=user2@company2.com,
Message="This is a brief message"
```

Note: The double quote character is needed when a value contains embedded blanks or commas.

Example

In the example below, a text file (*c:\report.txt*) is sent from *user1* to *user2* with a subject.

```
EDMMAILQ from=user1@com1.com,
to=user2@com2.com,Mesgfile=
c:\report.txt,Subject= "My report"
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMALLO (MVS Only)

Method Name EDMMALLO

Description Enables you to allocate a file dynamically for use in a REXX executable. The file will be freed automatically after the EXECIO command is issued with the FINIS option, or when the EDMMALLO method is issued. The maximum number of files a single REXX executable can allocate is 50. Use this method to set aside storage for the resulting output of a REXX method or program.

Alias Name ZIRXALOC

Parameters ALLOPARM

Parameter Name	Data Set Name to be Allocated
ALLOPARM	One string with the three former parameters (DDN || "," || DISP || "," || MEM;) passed with commas separating the values.

Example

```
/************************ REXX ********************/
SAY 'SAMPLE FILE ALLOCATION'
/********** ALLOCATE THE DATA SET *****************/
DSN = 'DSN=USER1.EDM.TESTLIB';
DISP= 'DISP=SHR';
MEM= "MEMBER=TESTMEM"
ALLOPARM = DDN || "," || DISP || "," || MEM;
/**********************************************/
/* ALLOCATE THE FILE                          */
/**********************************************/
DDN = EDMMALLO(ALLOPARM)
/**********************************************/
/* READ ALL THE RECORDS FROM THE FILE INTO A  */
/* STRUCTURE WITH RECS. AS THE STEM VARIABLE FOR  */
/* EACH RECORD.  THE FINIS OPTION WILL CAUSE THE  */
```

```
/* DATA SET TO BE CLOSED AND DEALLOCATED.  IF THE  */
/* FINIS OPTION IS NOT SPECIFIED REXX KEEPS THE    */
/* DATA SET OPEN AND THE "EXECIO 0" COMMAND MUST   */
/* BE USED TO CLOSE THE FILE.                      */
/*************************************************/
"EXECIO * DISKR" DDN "(STEM RECS. FINIS"
SAY 'ALLOCATED   DDN = ' DDN;

/* "EXECIO * DISKR" DDN "(STEM RECS. " ←-THIS     */
/* WOULD NOT CLOSE/FREE                            */
/*************************************************/
/* DE-ALLOCATE THE FILE. THIS IS NOT NECESSARY AND */
/* WILL FAIL IF THE DATA SET ALREADY FREED.  IT    */
/* SHOULD BE USED IN CASE AN ERROR IN THE REXX     */
/* EXEC DOES NOT EXECUTE THE EXECIO COMMAND AT ALL,*/
/* NEVER OPENING/CLOSING AND FREEING THE FILE.  IF */
/* THE DATA SET IS NOT FREED OTHER USERS MAY NOT BE*/
/* ABLE TO ACCESS THE DATA SET UNTIL THE MANAGER   */
/* IS TERMINATED.                                  */
/*************************************************/
DDNF = EDMMDALO(DDN);
/*************************************************/
/* ALLOCATE THE FILE                               */
/*************************************************/
  SAY 'DEALLOCATED DDN = ' DDF;
END
```

Possible Return Codes

Return Code	Description
0	The method was successful.

EDMMCACH

Method Name EDMMCACH

Description Refreshes or disables caching.

Alias Name

Parameters `OPTION=`

Parameter Name	Description
OPTION =	Action to be taken. Values are ENABLE or /DISABLE.

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMCMPR

Method Name EDMMCMPR

Description Compresses an in-storage object.

Alias Name ZOBJCMPR

Parameters `object`

Parameter Name	Description
object	The name of the in-storage object to be compressed.

Example

```
ADDRESS EDMLINK EDMMCMPR 'ZTEST' ;
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMCOPY

Method Name EDMMCOPY

Description Copies an in-storage object. The resulting object has the same variables and number of heaps as the original object.

Alias Name ZOBJCOPY

Parameters `fromobject,toobject`

Parameter Name	Description
fromobject	The name of the existing in-storage object to be copied.
toobject	The name of the new in-storage object to be created.

Example

```
ADDRESS EDMLINK EDMMCOPY 'OBJECT1,OBJECT2' ;
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMDALO (MVS Only)

Method Name EDMMDALO

Description Enables you to dynamically de-allocate files. This method should be used only in case an error in the REXX exec does not execute the EXECIO command at all. Use this method to free the data set, and terminate the Manager, so other users will not be denied access to the data set.

Note: If the data set is already freed, this method is not necessary, and, if specified, a 'fail' will result.

Alias Name None – new method

Parameters DDNAME

Parameter Name	Description
DDNAME	Data set name (allocated by EDMMALLO) to be de-allocated.

Example

Possible Return Codes

Return Code	Description
0	The method was successful.

EDMMDB

Method Name EDMMDB

Description Locks and unlocks the database to all tasks except DMA and Console

Alias Name

Parameters `OPTION=`

Parameter Name	Description
OPTION =	

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMDCLA

Method Name EDMMDCLA

Description Deletes a class and all associated instances from the database. Note that EDMMDCLA will not delete ZRSOURCE instances.

Alias Name ZDCLASS

Parameters `(file),domain,class`

Parameter Name	Description
domain	The name of the domain that contains the class to be deleted. A maximum of eight characters.
class	The name of the class to be deleted. A maximum of eight characters.
file (optional)	The name of the file that contains the class to be deleted.

Example

```
/***************************** REXX ***************/

 DOMAIN = 'SYSTEMX ';
 CLASS  = 'TESTCLAS';

PARM = SUBSTR(DOMAIN,1,8)  ||  SUBSTR(CLASS,1,8);
 SAY 'PARM STRING IS ' PARM;
 ADDRESS EDMLINK EDMMDCLA PARM;
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMDELI

Method Name EDMMDELI

Description Deletes a heap of an in-storage object.

Alias Name ZOBJDELI

Parameters `object,instance#`

Parameter Name	Description
object	The name of the in-storage object to delete an heap from.
instance#	The heap number to delete.

Example

```
/************************ REXX ********************/
      DPARM = 'TESTOBJ,'||1|| ' ';
      ADDRESS EDMLINK EDMMDELI DPARM;
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMDELV

Method Name EDMMDELV

Description Deletes a variable from all heaps of an in-storage object. The EDMMDELV method verifies the existence of the specifed object and finds the specified variable in that in-storage object. The variable value is then removed from each heap in the in-storage object.

Alias Name ZOBJDELV

Parameters `object,variable`

Parameter Name	Description
object	The object that contains the variable to be deleted.
variable	The name of the variable to be deleted.

Example

```
/************************ REXX ********************/
ADDRESS EDMLINK EDMMDELV 'TESTOBJ,VAR00001' ;
SAY 'QAREXX ****** VAR00001 DELETED FROM OBJECT
     TESTOBJ' ;
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMDINS

Method Name EDMMDINS

Description Deletes or displays an instance, or range of instances, within a class from the database. This method permits the use of wildcards (*). Note that the displayed instances will be written to the Manager log even if all other TRACE settings are off.

Alias Name ZDELINS

Parameters `file,domain,class,option, frominst,toinst`

Parameter Name	Description
file	The file of the instance(s) to be displayed/deleted.
domain	The domain of the instance(s) to be displayed/deleted.
class	The class of the instance(s) to be displayed/deleted.
option	The word "DISPLAY" if instance(s) is to be displayed, "DELETE" if instance(s) is to be deleted.
frominst	The instance name or starting name to be deleted or displayed.
toinst	The instance name to be deleted or displayed. Blanks in this field would indicate that it is a single instance to display or delete and not a range.

Example

```
/********************** REXX ********************/
FILE = 'PRIMARY'
 DOMAIN = 'SYSTEMX'
 CLASS = 'ZRSOURCE';
 FROMIN = 'TSO_                ;
 TOINS = '                ;
 OPTION = 'DISPLAY';
 PARM = FILE||DOMAIN||CLASS||OPTION||FROMIN||TOINS;
 SAY 'PARM STRING IS 'PARM;
 ADDRESS EDMLINK EDMMDELI PARM;
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMDOBJ

Method Name EDMMDOBJ

Description Deletes an in-storage object.

Alias Name ZDELOBJS

Parameters `object`

Parameter Name	Description
object	The name of the in-storage object to be deleted.

Example

```
ADDRESS EDMLINK EDMMDOBJ 'ZTEST';
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMDPRO

Method Name EDMMDPRO

Description Deletes a class in the PROFILE database.

Alias Name ZDELPROF

Parameters `domain,class`

Parameter Name	Description
domain	The domain the object to be deleted is located.
class	The class in which the object to be deleted is located.

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMENCR

Method Name EDMMENCR

Description Enables you to activate encryption. For security purposes this makes data unreadable until it has been decrypted. Use to encrypt/decrypt a variable, such as a password, within an object.

Alias Name None – new method

Parameters `action,fromobject,toobject,fromvar, tovar`

Parameter Name	Description
action	Specifies the action to be taken.
fromobject	The object from which to import the data.
toobject	The object in which to import the data.
fromvar	The variable from which to import the data.
tovar	The variable in which to import the data.

Example

```
ACTION = 'ENCRYPT'
FROMOBJECT = 'object_name1' ;
TOOBJECT = 'object_name2' ;
FROMVAR = 'var_name1' ;
TOVAR = 'var_name2' ;
```

Syntax

TO ENCRYPT:

```
EDMMENCR
ACTION=ENCRYPT,FROMOBJECT=obj_name1,TOOBJECT=obj_name
2, FROMVAR=var_name1,TOVAR=var_name2
```

TO DECRYPT:

```
EDMMENCR
ACTION=DECRYPT,FROMOBJECT=obj_name1,TOOBJECT=obj_name
2, FROMVAR=var_name1,TOVAR=var_name2
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMEXIS

Method Name EDMMEXIS

Description Verifies the existence of a given class or instance in the object database.

Alias Name ZEXIST

Parameters `file.domain.class.instance`

Parameter Name	Description
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 = 'SYSTEMX ' ;
CLASS = 'USER' ;
INST = 'USER1' ;
PARM = FILE  || '.' || DOMAIN || '.' || CLASS || '.'
|| INST ;
ADDRESS EDMLINK EDMMEXIS PARM ;
IF RC = 0 THEN
    SAY 'QAREXX ****** OBJECT ' INST ' EXISTS;
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMGPRO

Method Name EDMMGPRO

Description Creates an in-storage object from a PROFILE database object. The EDMMGPRO method gets the *dbobject* object from the PROFILE database and puts it in storage as *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 object is gotten from the _UNKNOWN domain.

Alias Name ZGETPROF

Parameters `dbobject,inobject,ZUSER,instance`

Parameter Name	Description
dbobject	The PROFILE database 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 (optional)	The name of the instance.

Example

```
/*************** REXX ****************************/
    PARM='ZSTATUS,ZSTATUS,'ZMASTER.ZUSERID;
    ADDRESS EDMLINK ZGETPROF PARM;
/* GET OLD PROFILE.?.ZSTATUS                      */
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMNFYT

Note: For more information, refer to *Chapter 3: Notifying Clients*.

Novadigm does not recommend using EDMMNFYT too often, EDMMPUSH is the preferred Notify method.

Method Name EDMMNFYT

Description Notify enables you to initiate a PUSH (the execution of a program or programs on a client desktop from another location). To execute Notify successfully, EDMEXECD must be running on the Clients on which you are executing a PUSH.

Alias Name ZNOTIFY

Parameters

```
target IP address,port,UserID,
password,"process to run",domain
name,instance name
```

Parameter Name	Description
"process to run"	The application you are forcing the client desktop to execute.
domain name	The name of the domain where the notification results are stored. If this parameter is not specified, the domain name will be automatically generated as a function of date/time.
instance name	The instance name containing the results of the single notification. If this parameter is not specified, the instance name will be automatically generated as an eight-digit number (default is 00000001).
password	The ZNFYPWD for the target terminal's ZMASTER object.
port	For MVS only. The port number. Should have the same value as the ZMASTER port number.
target IP address	The IP address of the client desktop on which you are executing a PUSH.
userID	The Client user ID.

Note: You should specify both the domain name and instance name parameters. If these 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("EDMMNFYT",0)
NHEAPS = EDMMNFYT
DO CURRHEAP = 1 TO NHEAPS BY 1
  RC = EDMGET("EDMMNFYT",CURRHEAP)
  NIPADDR = EDMMNFYT.IPADDR
  NPORT = EDMMNFYT.PORT
  NUSER = EDMMNFYT.USER
  NPASSW = EDMMNFYT.PASSW
  NCMDLINE = strip(EDMMNFYT.CMDLINE)
  NHANDLE = EDMMNFYT.HANDLE
  /*** CALL EDMMNFYT TO ISSUE THE NOTIFY ***/
  ADDRESS EDMLINK "EDMMNFYT" NIPADDR || ',' || NPORT
|| ',' || NUSER || ',' || NPASSW || ',"' || NCMDLINE
|| '"'|| ',' NHANDLE || ',' || CURRHEAP;
END
```

In MVS, you will configure NFYTTST in console.

On the Unix server, to ensure that the Unix process was started, type the following command at the Unix prompt:

```
ps -u [username]
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMOLOG

Method Name EDMMOLOG

Description Writes the contents of an in-storage object to the Manager log. Note that EDMMOLOG will write to the Manager log even if all other TRACE settings are off.

Alias Name ZVARLOG

Parameters `object`

Parameter Name	Description
object	The name of the in-storage object to be displayed.

Example

```
ADDRESS EDMLINK EDMMOLOG 'ZMASTER';
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMPHIS

Method Name EDMMPHIS

Description Puts an in-storage object into the HISTORY file database. The EDMMPHIS method takes the *inobject* 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.

Alias Name ZPUTHIST

Parameters `dbobject,inobject`

Parameter Name	Description
dbobject (optional)	The object name that will be put in the HISTORY file.
inobject	The object name of the in-storage object.

Example

```
ADDRESS EDMLINK EDMMPHIS 'ZCOMPARE,ZCOMPARE';
ADDRESS EDMLINK EDMMPHIS 'ZSTATUS';
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMPPRO

Method Name EDMMPPRO

Description Puts an in-storage object into the PROFILE database. The EDMMPPRO method takes the in storage *inobject* and puts it in the PROFILE file as *dbobject.* The domain in the PROFILE file is ZUSERID, found in the *inobject* or in the ZMASTER object. If ZUSERID is not found, the *dbobject* is put in the _UNKNOWN domain.

Alias Name ZPUTPROF

Parameters `dbobject,domainid,inobject,`

Parameter Name	Description
dbobject	The object name that will be put in PROFILE database. If dbobject is not specified, the default is the object name.
domainid	The value of an optional third operand may 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 'ZCOMPARE,ZCOMPARE ';
    ADDRESS EDMLINK ZPUTPROF 'ZSTATUS ';
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMPROM

Method Name EDMMPROM

Description This method allows the addition and updating of heaps in the PRIMARY database based on the contents of the passed parameter object. Each heap in the object specifies an instance to be added, updated or deleted.

Alias Name ZPROMANY

Parameters `object`

Parameter Name	Description
Object	The name of the in-storage object.

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMPUSH

Method Name EDMMPUSH

Description Receives input requests, gets required parameters and then puts requests to the right queues that later will be processed by a specific Notify Manager. An in-bound object, or even a dynamic object, created as a result of the object resolution may be used to deliver requests to the EDMMPUSH. The return code associated with the in-bound object may initiate further action. (See *Chapter 3: Notifying Clients* for more information.)

Alias Name New

Parameters `object name,default object = ZNOTIFY`

Variable	Description
NFYDELAY	Specifies the interval for delay before trying to re-notify a Client. If no value is entered, the default value is the value specified in the NFYT_TIMEOUT or NFY6_TIMEOUT setting of the MGR_NOTIFY section of the Manager Settings file.
NFYHNDL	Specifies 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	Specifies the maximum number of retries. If no value is entered, the default value is the value specified in the NFY_RETRY setting of the MGR_NOTIFY section of Manager Settings.
NTFYRTIM	Novadigm timestamp defining the time after which the notification should occur.

Variable	Description
NFYPROC	Controls processing of the current heap request. If the value is **Y** (yes), the heap will be processed. If the value is **N** (no), the request for the current heap will be ignored. The default value for this variable is **Y**.
NFYTYPE	Defines the type of the Notify requested. The following values are allowed: • TCP • SIPX • NETB • LU62 • EMAIL Just the first three bytes of the type are used for the identification, so SIPX and SIP will be treated as the same. There is no default value for this variable. If this variable is not defined, the current heap of the object will be ignored.
NFYUINFO	Allows you to enter user information.

Example

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMRESO

Method Name EDMMRESO

Description Resolves the object specified by the parameter string, and the resulting objects are left in storage. Any prerequisite objects needed by that resolution must already have been built in storage. For example, for USER.&ZUSERID resolution, a ZMASTER object might have to be constructed containing ZUSERID, ZOS variables, otherwise, the resolution may not be entirely successful.

Alias Name ZSIMRESO

Parameters `file.domain.class.instance(message)`

Parameter Name	Description
file (optional)	The file that contains the instance to resolve.
domain (optional)	The domain that contains the instance to resolve.
class	The class that contains the instance to resolve.
instance	The instance to resolve.
message (optional)	This specifies the message for conditional resolution paths.

Example

```
RESOLVE SYSTEMX.USER.  ZMASTER.ZUSERID

DOMAIN='SYSTEMX ';
CLASS ='USER';
INST  = 'USER1';
PATH  ='EDMSETUP';
PARM  = DOMAIN || ' ' || CLASS || ' ' || INST || ' '
     || PATH;
ADDRESS EDMLINK ZSIMRESO DOMAIN CLASS INST PATH
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMRPRO

Method Name EDMMRPRO

Description This method allows the addition, updating and deletion of heaps in the PRIMARY database based on the contents of the passed parameter object. Each heap in the object specifies an instance to be added, updated or deleted.

Alias Name New

Parameters `object`

Parameter Name	Description
object	The name of the in-storage object. The object can have multiple heaps where each heap in the object represents an instance in the database to be altered.

Possible Return Codes

Return Code	Description
0	The method was successful.
>0	The method was not successful.

The instance indicates which database instance will be altered by specifying five control variables:

Control Variable	Description
ZOBJDOMN	Target domain - e.g., "SYSTEMX"

Control Variable	Description
ZOBJFILE	Target file – e.g., "Primary"
ZOBJCLAS	Target class - e.g., "ZRSOURCE" or "ZSERVICE"
ZOBJNAME	Target instance - any valid instance name
ZREASON	Action requested - "ADD", "UPDATE" or "DELETE"

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 database. The fields that can be updated are variables, class connections, expressions, and methods. There are differences in specification for the three field types. Regardless of field type, the target instances' field length determines 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. The methods are indicated by variables in the parameter object that are named MTHD*nnnn*, where *nnnn* is 0001-9999. The value of the variable in the object should contain

"*methodfieldname*=xxxxxxxxxx"

Note: The string before the equals (=) sign must be eight characters in length.

where *methodfieldname* is used to identify the target method field . For example:

"_ALWAYS_=ZSYSTEM.ZMETHOD.SIGNON_METHOD"

or

"EDMSETUP=ZSYSTEM.ZMETHOD.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 0001-9999. Types of connection fields include CONN*nnnn* (connection), INCL*nnnn* (Includes), _ALW*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_=ZSYSTEM.ZSERVICE.MY_SERVICE").***

You can update specific target instances while not overwriting some existing values (e.g., 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. An empty value specified before a comma indicates that the connection should skip over the existing value and not update that value. For example, the format for skipping over the first two variables and updating the third would appear as:

  CONN0001 "EDMSETUP=, ,ZSERVICE.ABC"

***These variable names are the same format as object resolution with a message type = "_NONE_". This is designed to allow for the output of these resolutions (sometimes referred to as reporting resolutions) to be used unchanged as input to EDMMRPRO.

Notes on EDMMRPRO Usage

- The file, domain and class must already exist, EDMMRPRO will not add any of these levels dynamically. Also any fields being processed must already be defined in the target class, EDMMPROM will not modify classes.
- Different classes instances can be altered during one execution of EDMMRPRO, it is not advisable to do so as certain field names may overlap (particularly methods and connections).
- Different databases can*not* be altered in one execution of EDMMRPRO.

EDMMSIGN

Method Name EDMMSIGN

Description This method enables you to authenticate a client session against the database. The password stored in the ZPWD variable in the specified object is compared to the password stored in the user's profile. If the passwords match, the session continues unless the message "PASSWORD INVALID" is sent back to the client. Passwords can be changed by specifying the new password in "ZNEWPWD" and the old password in "ZPWD."

Alias Name EDMSIGN

Parameters `object name`

Parameter Name	Description
object name	Specifies the name of the object from which the ZPWD variable is extracted. If no object name is specified, the ZMASTER object is used as the default.

Example

REXX

```
EDMMSIGN
     (Uses the ZMASTER Object)
EDMMSIGN & (ZCURRENT>ZCUROBJ)
```

Possible Return Codes

Return Code	Description
0	The method was successful.
4	New user.
8	An error was detected, the method failed.

EDMMSGNR

Method Name EDMMSGNR

Description This method enables you to authenticate a client session against the password for the user found in an external security system such as RACF, Top Secret, or in Windows NT security. The password stored in the ZPWD variable in the specified object is compared to the one stored in the native security system for the user ID specified in ZMASTER.ZUSERID. If the passwords match, the session continues unless the message "PASSWORD INVALID" is sent back to the Client.

Passwords can be changed for Windows NT and a new value specified in duplicate in ZMASTER.ZNEWPWD and ZMASTER.ZVERPWD. After the password has been successfully changed, the new value is migrated to ZMASTER.ZPWD and ZMASTER.ZNEWPWD and ZMASTER.ZVERPWD are reset. There are also new variables that define the password length. These are ZPWDL, ZNEWPWDL, and ZVERPWDL. A value of 0 in the password length means that the length is not specified. A length of non-0 defines how many bytes of the password are significant.

Note that passwords can be changed by specifying the new password in "ZNEWPWD" and the old password in "ZPWD" for MVS only.

Alias Name EDMSIGNR

Parameters `object name,default object=ZMASTER`

Parameter Name	Description
object name	Specifies the name of the object from which the ZPWD variable is extracted.

Example

```
/*******************REXX**************************/
EDMMSGNR
     (Uses the ZMASTER Object)
EDMMSGNR & (ZCURRENT>ZCUROBJ)
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMSORT

Method Name EDMMSORT

Description Sorts the heaps of an in-storage object by the values of specified variables and according to the desired collating sequence.

Alias Name ZOBJSORT

Parameters `SORT,object,variable1,variable2, variable3`

Parameter Name	Description
sort sequence	Ascending (SORT) or descending (SORTD).
object	The name of the object to be sorted.
variable name(s)	Up to three variable names can be specified.

Example

```
PARM1 = 'SORT,'
PARM2 = 'ZSERVICE'
PARM3 = ',ZOBJNAME,ZOBJDATE,ZOBJTIME ';
PARM  = PARM1 || PARM2 || PARM3;
ADDRESS EDMLINK EDMSORT PARM;
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMTUCH

Method Name EDMMTUCH

Description Updates the date/time stamp of an instance in the database.

Alias Name ZTOUCH

Parameters `class,domain,instance`

Parameter Name	Description
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 = "SYSTEMX";
CLASS  = "ZRSOURCE";
INST   = "TEST_OBJECT";
PARM= SUBSTR(DOMAIN,1,8)||SUBSTR(CLASS,1,8)||SUBSTR
      (INST,1,32);
ADDRESS EDMLINK EDMMTUCH PARM ;
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMULOG

Method Name
EDMMULOG

Description
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 Manager Settings file. In addition the ACTIVATE= setting must be specified as YES.

Alias Name

Parameters
MSG

Parameter Name	Description
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
```

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

Manager Settings Section

```
[MGR_USERLOG]
ACTIVATE   = YES
DIRECTORY  = .
THRESHHOLD = 500000
FLUSH_SIZE = 128
COLUMN_WIDTH = 128
PIPE_SIZE  = 100000
```

EDMMVDEL

Method Name EDMMVDEL

Description Deletes all in-storage objects.

Alias Name ZVARDEL

Parameters None

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMVGBL

Method Name EDMMVGBL

Description Migrates values from one in-storage object to another. This method then deletes the source object.

Note: Only variables with appropriate flag settings will be migrated.

Alias Name ZVARGBL

Parameters `destination,source`

Parameter Name	Description
destination	Object with values being migrated to.
source	Object with values being migrated from.

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

EDMMXREF

Method Name EDMMXREF

Description EDMMXREF cross-references class and instance usage during object resolution and collects information on the cross reference objects.

The EDMMXREF method will generate objects that enable administrators to cross-reference users with any, and all, Departments, Workgroups, and Services that they are affiliated (connected) with.

To implement EDMMXREF method:

- Add new method instances to ZMETHOD. Some are methods to create the objects from the PRIMARY database and others to write these objects to the Profile database.
- Update the Manager process (ZMASTER) used during the client connect process by adding new methods to ZSYSTEM.ZPROCESS.ZMASTER.
- Update your class template(s), if necessary, to add new method attributes.
- Update the _BASE_INSTANCE_ to specify the method instances to be executed.

Alias Name ZXREF

Parameters `object`

Parameter Name	Description
object	

Possible Return Codes

Return Code	Description
0	The method was successful.
8	An error was detected, the method failed.

Index

F

G

H

I

J

L

M

N

O

P

R

S

T