Merced™ Performance Suite

System Administration Guide

Software Version 2.7



Copyright © 2006 Merced Systems, Inc. All rights reserved.
Merced is a trademark of Merced Systems, Inc. All other trademarks are the property of their respective owners.
All information in this document is based on the latest product information available at the time of printing. Merced Systems has carefully reviewed the accuracy of this document, but shall not be held responsible for any omissions or errors that may appear.
Merced Systems reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation to notify any person of such revision changes.

Document Revision 27-002-0

April 21, 2006

Contents

About this Document vii

Chapter 1 **Network Architecture** 1

Clusters 2
Environments 4
- Production environment. 4
- Staging environment 4
- Development environment. 5
- Databases for different environments 5

Hardware. 7
Software structure 9
- Constructing an application 9
- Customizing an application 10
- Application directory. 11

Chapter 2 **Installation Set Up** 13

Physical environment. 14
- Location 14
- System user 14
- Network 15

Application server machines 19
- MERCED_HOME. 19
- Time zone 19
- Enabling charts in Unix. 21
- Java 23
- Python 23
- Running engines and proxies 24

Database server machines 30
Oracle 30
DB2 Universal Database 32
SQL Server 35

Chapter 3 System Configuration 37

Cluster configuration 38
Cluster component 38
Node component 39
Host component 40
Proxy component 41
Engine component 45
Setting ports 49
Database configuration 52
Database password encryption 52
Connection component 54
Linking connection to repository 55
Connection pools 56
Testing database configuration 58

Chapter 4 Authentication 59

How authentication works 60
Standard authentication 63
Implementing standard authentication 63
Creating dispatching validation services 64
LDAP authentication 67
Configuring LDAP validation service 67
Using SSL with LDAP validation 72
Implementing LDAP authentication 73
NTLM single sign-on authentication 74
SPNEGO single sign-on authentication 78
Implementing SPNEGO authentication 78
Debugging SPNEGO authentication 84
WebSEAL single sign-on authentication 88

Chapter 5 Monitoring 91

Maintenance 92
Understanding system processes 92
Maintaining application server machines 94
Maintaining database server machines 100
Maintaining connectivity 101

Automated monitoring and maintenance 105
Monitoring jobs 105
Maintenance jobs 107
Debugging 108
Locating log files 108
Reading log files 109
Analyzing log files 110
Troubleshooting 112
Common system issues 112
Troubleshooting tasks 113

Appendix A **Glossary** 117

Appendix B **System Requirements** 125

Application server machine 126
Database server machine 128
Client machine 129

Appendix C **Commands** 131

Using quotes in commands 133
dumpcc 134
jsql 135
m 137
m cubepop 138
m dice 144
m encrypt 147
m export-cbeans 148
m file-splitter 149
m import-cbeans 150
m load-metrics 153
m mserver 155
m optimize-db 157
m optimize-repository 158
m repository-upgrader 160
m schemaload-main 161
m schemaload-reports 162
m semaphore 163

Appendix D **Data Dictionary** 165

Categories 166
Alerts, messages, and tasks 166
Forms 167
Head count job 167
Identities 167
Job executions 167
Login 168
Org chart 168
Scheduled jobs 169
Time grains 170
Transactional reports 170
Other tables 171
Data types 172
List of tables 173

Appendix E **Schema Diagrams** 229

Identities 230
Job executions 231
Org chart 233

Index 235

About this Document

This *System Administration Guide* describes the network configuration and the system maintenance and monitoring that Merced Performance Suite requires. This document will help you understand:

- How the system fits within your network architecture
- What hardware and software is required
- How you configure server machines to support Merced Performance Suite
- How you configure the application to support various authentication methods
- How to monitor the hardware, network, and database
- What tasks you need to complete to maintain the application

The primary audience of this document is system administrators, those responsible for maintaining the hardware and network where the system runs, monitoring Merced Performance Suite at a system level, and troubleshooting issues that might arise.

The secondary audience of this document is the application project team—a separate group of analysts, developers, and engineers—who customizes the application configuration, builds a deployable application, and establishes and monitors data jobs that support the system.

This document assumes that you have read *Introduction to Merced Performance Suite* and are familiar with the software architecture and components of Merced Performance Suite as well as the uses of the application.

This document contains the following chapters:

- Chapter 1, "Network Architecture" — Describes the architecture that supports Merced Performance Suite, the types of environments you might set up, the hardware needed for the architecture in each environment, and the software structure that supports application configuration.
- Chapter 2, "Installation Set Up" — Describes how to set up the physical environment and each machine where you install Merced Performance Suite.
- Chapter 3, "System Configuration" — Describes the properties files that configure the cluster and the database.
- Chapter 4, "Authentication" — Describes the different methods of authentication that Merced Performance Suite supports and how to implement each method.
- Chapter 5, "Monitoring" — Describes the maintenance and monitoring tasks that you must complete to ensure that Merced Performance Suite runs properly.
- Appendix A, "Glossary" — Lists definitions of terms special to or frequently used in Merced Performance Suite documentation.
- Appendix B, "System Requirements" — Lists the supported operating systems and database applications for machines running Merced Performance Suite.
- Appendix C, "Commands" — Lists details about commands you might use when administering Merced Performance Suite.

1 Network Architecture

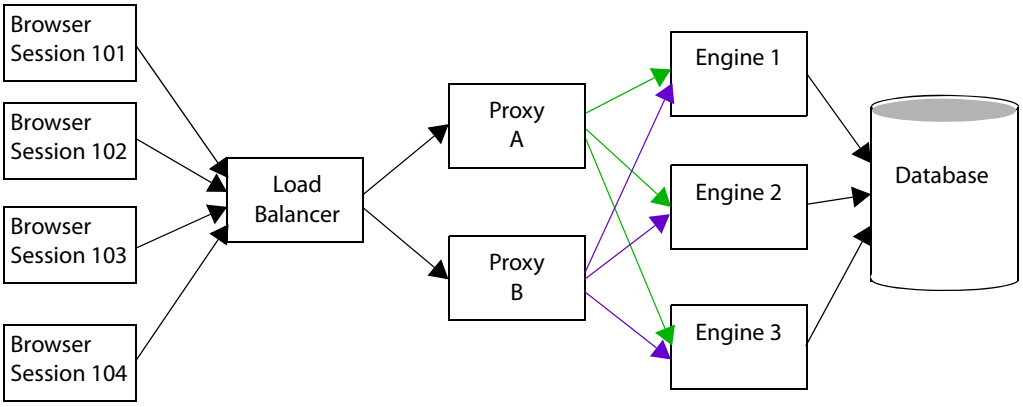
This chapter discusses the network architecture in which you deploy Merced Performance Suite. The system's network architecture is based on clusters, an architecture that supports horizontal scalability. Clusters enable multiple servers to share the computational load.

This chapter discusses:

- Clusters
- Environments
- Hardware
- Software structure

Clusters

A cluster enables you to operate multiple application engines that access the same database. With a cluster, users are unaware that there are multiple application engines; users sense a single application running from a primary URL.



There are four elements in a cluster:

- Load balancer — The load balancer is an independent network device that maps a particular network address (a URL) to an appropriate server, distributing the load among multiple servers. The load balancer sends a browser session to a subordinate server, usually a proxy.
- Proxies — A proxy is an instance of a Web server. Proxies determine what engine to run the browser session on, and redirect the browser to that engine. Proxies delegate work to engines and serve static, non-changing content, like images; proxies do not actually do any computation. A proxy runs Apache Web server software, included with Merced Performance Suite.
- Engines — An engine is an instance of an application server. Engines do the actual work of rendering reports and Web pages. Engines also authenticate users. An engine is the core

application. If an engine fails or is taken out of service, the proxy automatically reroutes traffic to other engines. Thus, no engine is a single point of failure.

- Database — A database stores the data that Merced Performance Suite uses and computes. All engines share this database, though they cache much of the data. The database stores the OLTP data mart and the OLAP data mart.

An organization might use a simplified cluster. For example, a cluster can omit a load balancer and consist of a single proxy with one or more engines and a database. An engine is the basic unit—the core application that end-users view and use. Any cluster must have at least one engine. In fact, you can run Merced Performance Suite with a single engine and no proxies or load balancers. This minimal configuration, however, offers no failover.

Merced Performance Suite is a Java application. Each engine and proxy must run inside its own Java Virtual Machine (JVM), a running Java process.

You install a cluster on a set of server machines. Each machine is called a node. See "Hardware" on page 7 for more information about the types of machines typically used for a cluster and how you might configure elements of the cluster on those machines.

Environments

To accommodate different functions and ensure a fully functioning production application, your organization might set up different environments. Each environment is a self-contained, functioning system containing a database and application server machines. Typical environments include:

- Production environment
- Staging environment
- Development environment

Production environment

A production environment is the cluster that users access when logging in to Merced Performance Suite. The production environment contains the database of record, which stores data extracted from sources and calculated data that measures performance.

Staging environment

A staging environment mimics a production environment, using the same hardware specifications and configuration. However, a staging environment is completely separate from the production environment. With a staging environment, you can validate the network architecture and configuration, including changes introduced by configuration or version changes in the application, without affecting the production environment.

Often, organizations use the staging environment for integration testing, when combining many configuration changes before deploying a new configuration. Other organizations use a separate QA (quality assurance) environment for testing functionality and use the staging environment to test and validate the architecture.

Development environment

A development environment also mimics a production environment, though the hardware specifications might not be an exact match. Multiple development environments can share a single database server machine—simply create a different database user, complete with its own set of tables and indices, for each environment. In fact, if you have limited resources, you might configure development environments to share a database server machine with a staging environment. Production environments, however, always require separate database server machines.

Organizations use the development environment to create and test configuration changes that customize an application. If you have sufficient resources, create a separate development environment for each application developer.

Databases for different environments

You need to create and maintain different databases (ideally, on different machines) for different environments. Typically, you create databases for the staging and development environments after deploying the production database.

➢ To create a staging or development database

1. Create a backup copy of the production database (in other words, export the production database).

   The backup copy must preserve details, such as indexes and sequences (and database statistics for DB2 UDB and Oracle databases).

2. Restore a backup copy of the database—that is, import the backup database—to your staging or development environment.

The restore process must preserve details such as indexes and sequences (and database statistics for DB2 UDB and Oracle databases).

Depending on the code and configuration differences between your production and staging environments, you might need to complete some upgrade steps after each restore.

Hardware

Merced Performance Suite requires hardware that supports scalability and reliability. Machines must support the cluster elements and the system jobs, such as the jobs that load the data marts, including DICE jobs and CubePop jobs. There are three types of machines:

- Application server machine— Runs the engines, proxies, and system jobs
- Database server machine — Runs the database
- Load balancer — A separate network device that routes network addresses

See Appendix B, "System Requirements," for a list of platforms supported on each type of machine.

Configuration of hardware to support Merced Performance Suite varies by the type of environment, by the hardware specification, and by the number of users. You want to configure the hardware to make best uses of resources, subject to memory and disk space constraints, while offering adequate redundancy to ensure the application is continuously available if a machine, engine, or proxy fails.

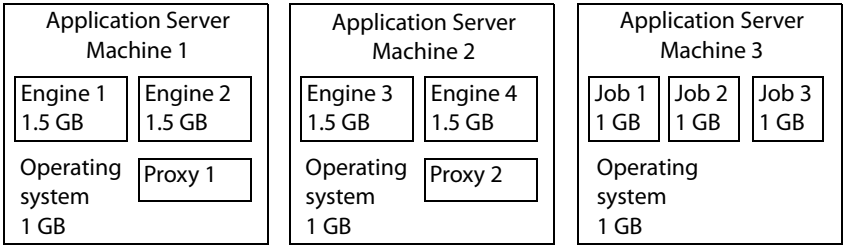
For example, a production environment might include two application server machines to provide optimum performance and might even include two database server machines to provide database failover. On the other hand, a staging or development environment might include a single application server machine and a single database server machine.

The amount of RAM on an application server machine affects how you can configure that machine. An application server machine supports:

- Operating system — Typically, you must retain 1 GB of RAM for operating system tasks.

- Engines — An engine is a Java process that requires a continuous block of memory. Windows systems limit continuous memory blocks to approximately 1.5 GB. If your machine has more memory, you can run more than one engine on the machine. Solaris systems generally do not limit continuous memory, and Linux systems have various limits. Engine configuration can specify the amount of memory that the engine uses.
- Proxies — A proxy uses very little memory and can usually share a machine with an engine. For redundancy purposes, configure two proxies per cluster, on separate machines.
- Jobs — A job is also a Java process that requires a continuous block of memory. Job configuration limits the amount of memory the job uses. By default, neither DICE jobs nor CubePop jobs use more than 1GB of RAM. You might devote a machine entirely to running jobs, or you might run jobs on the same machine as an engine, depending on the number of jobs you run at once.

For example, if you have Windows-based application server machines with 4 GB of RAM, a machine can support two engines. In this case, you might use three machines—two of them might run engines and proxies and the third might run jobs.



NOTE: Adequate memory must be available for an engine or job to start up. When a machine has inadequate memory, engines and jobs will not start and log files will contain no errors.

Software structure

This section discusses the software structure that supports the Merced Performance Suite application. This information will help you understand how an application is constructed, how to configure an application, and the structure of the directory that supports the application. Topics include:

- Constructing an application
- Customizing an application
- Application directory

Constructing an application

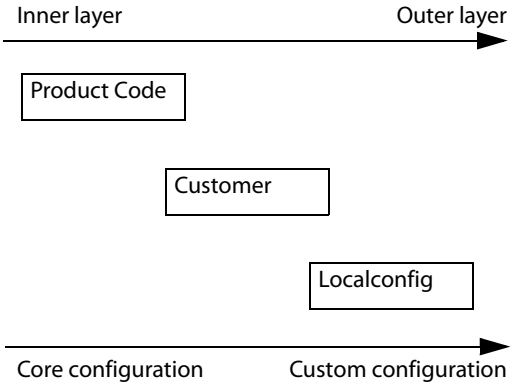
Merced Performance Suite is implemented in Java. The application's compiled code works with components. Properties files define each component. By setting property values in the properties files, you can customize the application to meet your organization's needs without changing and compiling code.

Merced Performance Suite includes a programming interface, called Creator, that interprets properties files and creates components from them. Creator is the mechanism for creating components and registering them within the system, making them available to other components. Each component and process in the system uses Creator to get required components.

You define components based on a core directory structure starting at the root directory. When Creator constructs components, it uses a relative directory path. Thus, when referencing other components in a properties file, you can reference the relative path within the Creator namespace.

Creator constructs Merced Performance Suite components in layers. Inner layers contain properties files that define application modules and default behavior. Outer layers contain properties files that customize the application. Each layer repeats the core directory structure. Creator constructs components by

merging properties files as it traverses the layers. Special files, called PACKAGE.ctl files, in the product code layer and the customer layer specify how Creator traverses layers and merges properties files.



Customizing an application

The Merced Client Services Group (CSG) and your organization's application project team configure Merced Performance Suite for your organization's needs, creating custom properties files and building a deployable application. CSG and the application project team do their customization work in the customer layer. You install the final built files that compose the deployable application—the engine—in a directory known as MERCED_HOME.

You can also customize a deployed application by placing properties files in the localconfig directory of MERCED_HOME. The localconfig directory makes up the localconfig layer, the outermost layer that Creator uses when constructing components.

For example, Chapter 3, "System Configuration," describes properties files that you use to configure the cluster. If you make a change to the network, like changing a machine's host name, you must update the appropriate properties files. The preferred way to update the application is to inform CSG or the

application project team, have them update the customer layer and build a new application, and then deploy that new build. However, for quick changes, you can create the properties file, insert it in the localconfig directory, and restart the application engine.

For more detailed information about Creator and application customization, see Merced Support at https://support.mercedsystems.com.

Application directory

Each application server machine must have a single directory, known as MERCED_HOME, that contains the files that support the Merced Performance Suite application, including proxies, engines, and jobs.

The MERCED_HOME directory has four key sub-directories:

- bin — Contains all executables and commands that you run, such as commands to run DICE, CubePop, or mserver.
- lib — Contains Java class files and properties files in compressed .jar files, such as devel.dice.jar, devel.cube.jar, and devel.reports.jar.
- var — Contains files and resources that the system generates as it runs. There is a sub-directory for each executable in the bin directory and for each server (that is, each engine or proxy).
- localconfig — Contains properties files that you insert to customize system temporarily. This is always the last directory that Creator uses when merging properties files. This directory starts empty in newly installed systems.

The MERCED_HOME directory also contains other directories that contain information needed to run the engine.

2 Installation Set Up

This chapter discusses how to set up your physical environment and your machines—the installation environment—to support Merced Performance Suite. Sections include:

- Physical environment
- Application server machines
- Database server machines

Physical environment

The physical environment where you install Merced Performance Suite involves the server machines, database, and network hardware. Properly establishing the physical environment can improve performance and ensure the security of the data that the application maintains.

This section discusses:

- Location
- System user
- Network

Location

Merced recommends storing sensitive hardware in locked facilities with limited access. “Sensitive hardware” includes hardware that stores data and that data passes through, such as server machines, databases, and network hardware.

System user

Merced recommends installing and running Merced Performance Suite as a non-system user—a user other than root on Unix-based machines and Administrator on Windows machines. Installing the application as a non-system user ensures that non-system users can run the application. Running the application as a non-system user prevents anyone from using the application to access and alter or eliminate key system files.

Network

Merced makes several recommendations regarding network configuration:

- Subnet configuration — Install database server and application server machines within the same subnet. The packet size from database queries can become large; traversing large distances across the network can significantly hinder performance.
- Data encryption — Install Merced Performance Suite with SSL (secure sockets layer). SSL uses a private key to encrypt data as it travels over the network. SSL requires a valid certificate from an appropriate authority.
- Firewall — Configure a firewall in front of the Merced Performance Suite cluster and load balancer, if your site uses one.

This section provides further details about:

- Secure sockets layer (SSL)
- Firewall

Secure sockets layer (SSL)

SSL (Secure Sockets Layer) is a protocol that ensures secure communications. SSL encrypts data as it travels over the network.

Why use SSL

Browsers communicate with Web servers using a protocol called HTTP. Regular HTTP does not encrypt data; third parties can "sniff" or steal passwords sent over HTTP while data is en route. Because SSL encrypts all HTTP traffic, including passwords, SSL can help prevent the theft of a user's password. When the network encrypts HTTP using SSL, the resulting protocol is known as HTTPS.

For more details on SSL, see http://en.wikipedia.org/wiki/Secure_Sockets_Layer.

Certificate Authorities

A server that uses SSL must have an SSL certificate. For a fee, you can obtain an SSL certificate from a business called a Certificate Authority (CA). Merced uses GeoTrust, but you can use any trustworthy CA as long as its certificate is distributed with your organization's supported browser.

What is a certificate

A certificate is basically a CA's digital signature, certifying that a public key belongs to your DNS domain name. A file containing a certificate frequently also contains the public and private keys. If the file contains a private key, you must keep the file secure; if someone steals your server's private key, that person has stolen your server's identity.

Certificates commonly expire after one to three years.

Since converting formats can be difficult, Merced recommends that you obtain a certificate in PEM-encoded X.509 format. See http://en.wikipedia.org/wiki/X.509 for more information about this format.

Obtaining a production certificate

If you already have a certificate for your domain name, you can use that certificate. Inquire within your organization. Otherwise, you must generate a new certificate.

➢ To generate a new certificate

1. Create a public/private key pair.

   You can use the tool kit available from OpenSSL (http://www.openssl.org).

2. Create a Certificate Signing Request (CSR) from your key pair.

3. Submit the key pair and CSR to the CA of your choice.

4. Wait for your CA to return your certificate.

5. Install your certificate in Merced Performance Suite.

   Merced Performance Suite supports SSL to proxies. You link the proxy to the SSL certificate when configuring the proxy component. See "Proxy component" on page 41.

For details about using the OpenSSL toolkit to complete this procedure, see the OpenSSL manual (http://www.openssl.org/docs/apps/openssl.html).

Educating users

SSL relies on users to check the authenticity of server certificates. Train your users not to accept SSL sessions with your server if the Web browser cannot verify the certificate.

If users do not demand a certificate that can be verified by their browser, they will be vulnerable to man-in-the-middle attacks.

Firewall

SSL protects data sent over the network. In contrast, a firewall protects your servers from unexpected access via the network.

Merced recommends configuring a firewall in front of your Merced Performance Suite cluster and load balancer, if your site uses one.

The firewall should only allow incoming connections to:

- The "front door" HTTPS port
- Any secure administration ports that are required

The firewall should disallow incoming connections to:

- Engines
- Non-essential administration ports

The firewall must allow outgoing connections from DICE jobs. DICE jobs use a variety of protocols to connect with source systems.

Application server machines

This section discusses how to set up the application server machine before installing and running Merced Performance Suite and how you can run engines and proxies. Specific topics include:

- MERCED_HOME
- Time zone
- Enabling charts in Unix
- Java
- Python
- Running engines and proxies

MERCED_HOME

The MERCED_HOME directory is the directory where you install all files supporting Merced Performance Suite and where the system writes all log files by default. See "Application directory" on page 11 for a description of this directory.

Create a directory, such as D:/Merced, to serve as the MERCED_HOME directory. Create this directory in the larger partition or disk on your application server machine (the drive should have at least 200 MB of free space).

NOTE: The name of the MERCED_HOME directory cannot contain spaces.

Time zone

Merced Performance Suite components reference two time zones:

- Internal time zone
- Standard time zone

Internal time zone

Merced Performance Suite components use their internal time zone when writing log messages. By default a component's internal time zone is set to the time zone of the operating system where the component starts. You can find the operating system time zone by checking the system clock in Windows, or by checking the TZ environment variable in Unix.

To render reports properly, the application engine's internal time zone must match the cluster's standard time zone. You can configure an engine to use an internal time zone other than the default on the application server machine. To configure the internal time zone for all engines in your cluster, set the value of the **defaultTimeZone** property in the /Configuration component in the customer layer. Setting this property also sets the internal time zone for select main service components, such as DICE and CubePop, which use the internal time zone to stamp log messages. Alternatively, you can configure the internal time zone separately for each engine, using its **timeZone** property (see "Engine component" on page 45).

Standard time zone

Merced Performance Suite maintains another time zone, called the standard time zone, in the /com/merced/timezone/TimeZoneManager component. Other components reference the standard time zone. When configuring DICE conversions, you often convert source system data to this standard time zone. Recording data in a standard time zone allows you to track events uniformly throughout an organization. Similarly, CubePop references a standard time zone when creating the time dimension. Application engines must reference the same standard time zone as CubePop to avoid reporting errors.

By default, the standard time zone is set to Pacific Time. You can override this value with the **standardTimeZone** property in the /com/merced/timezone/TimeZoneManager component. To avoid data discrepancies and reporting errors, set the standard time zone globally for the entire cluster.

Contact CSG or your organization's application project team for assistance.

Enabling charts in Unix

Merced Performance Suite charts use the Java graphics library, which relies on native graphics libraries. In many Unix platforms, native graphics libraries rely on the X Window System. When rendering charts, those libraries use the DISPLAY environment variable to communicate with the X server.

When running a Unix-based application server machine, you must ensure that your system is set up to allow charts to display correctly.

➢ To enable charts in Unix systems

1. Install X server.

   Only a minimal configuration is necessary. You do not need to install a display manager or GUI environment. You might need the following packages:

   - XFree86-libs-data-4.3.0-78.EL
   - XFree86-Mesa-libGL-4.3.0-78.EL
   - XFree86-xfs-4.3.0-78.EL
   - XFree86-xauth-4.3.0-78.EL
   - XFree86-libs-4.3.0-78.EL
   - XFree86-font-utils-4.3.0-78.EL
   - XFree86-base-fonts-4.3.0-78.EL
   - XFree86-4.3.0-78.EL

2. Disable external network connectivity for X server, enabling X Windows to listen only for local requests.

   a. Create a file named /etc/X0.host.

b. Insert a single line in the file:

```
localhost
```

3. Start X server.

Insert the command `X :0` in the server machine's start-up script. This command starts X server but not the display manager.

For example, you might create a script named /etc/init.d/xstart.

/etc/init.d/xstart

```
# cat /etc/init.d/xstart
#!/bin/sh
#
# chkconfig: 3 99 10
# description: Start an empty X server for NetCharts to connect to.
# processname: X
# config: /etc/X11/XF86Config
# pidfile: /var/run/X.pid

/usr/X11R6/bin/X :0 &

# Uncomment the following line if you are using an external method to
# control access to the X server, i.e., a firewall not X0.host
# /usr/X11R6/bin/X -ac :0
```

Enable this script during machine start-up by running the command:

```
chkconfig on xstart
```

4. Set the DISPLAY environment variable to ":0.0"; type:

```
export DISPLAY=:0.0
```

After the Merced Performance Suite application is installed and running, verify that charts actually appear correctly by visiting a report that displays a chart.

Java

Install Java 2 Platform, Standard Edition, v 1.4.2_05 SDK in the default directory on all application server machines (machines running engines, proxies, or jobs).

Set the JAVA_HOME environment variable to point to the directory where Java is installed on your machine. Typically, this is C:\j2sdk1.4.2_05.

To set the variable in a Unix environment, you might type:

```
export JAVA_HOME=/usr/local/j2sdk1.4.2_05
```

NOTE: The name of the JAVA_HOME directory cannot contain spaces.

Python

Install Python version 2.2.1 in the default directory on any application server machine (a machine running engines, proxies, or jobs). Merced Performance Suite uses Python to link system variables to directories on your machine when starting an application.

Set the PYTHONHOME environment variable to point to the directory where Python is installed on your machine.

Running engines and proxies

After installing Merced Performance Suite you need to set up the application server machine to run engines and proxies. The machine's operating system affects the options for running engines and proxies.

- Windows systems — Run engines and proxies as system services. Alternatively, you can run engines using the m mserver command. Use scheduled tasks or a similar application to schedule restarts of engines and proxies.
- Unix systems — Use the m mserver command to start engines and proxies. Use the cron job to schedule restarts of engines and proxies.

This section contains more information about:

- Selecting user to run engine or proxy
- Using system services
- Using m mserver command
- Scheduled tasks
- Log files

Selecting user to run engine or proxy

You must run an engine or proxy as a particular user. Merced recommends you select:

- A non-system user (a user other than root on Unix machines and Administrator on Windows machines). Running engines and proxies as non-system users prevents anyone from using the engines or proxies to access and alter or eliminate key system files.
- Different users for each engine and proxy on a particular machine. Engines, proxies, and DICE conversion jobs are Java processes. In your system process monitor, such as Task Manager for Windows systems, you can use the user name to

distinguish between different processes. Distinguishing between different engines and proxies is helpful when troubleshooting.

When you run a process as a non-system user, the process cannot use the range of port numbers that IANA (Internet Assigned Numbers Authority) designates for system users (ports below 1024). However, the IANA standard port for HTTP access is 80 and the IANA standard port for HTTPS access is 443. Browsers use these standard ports to contact a domain if the URL does not specify another port. Therefore, you must use the standard port if you do not want to include a port number in the URL to access the Merced Performance Suite application.

To use these standard ports when using a load balancer, configure the load balancer to map your domain and the IANA standard port to the port that the proxy listens to. If you are not using a load balancer, run the proxy as the system user and configure the proxy to listen to the IANA standard port; run engines as non-system users.

Using system services

System services integrate engines and proxies with the Windows operating system. System services allow your proxies and engines to run independently; a user need not be logged in for engines or proxies to run. With system services, engines and proxies can continue running when you log out and can start automatically after the machine boots up. System services cannot always access shared network drives. Therefore, multiple application server machines cannot share a directory on a single machine, such as a single MERCED_HOME directory.

You can see the system services registered on your machine from **Services**, accessible from **Administrative Tools** on **Control Panel**.

Registering

For each node that has an engine, register one system service for all engines on that node. In a command line for the node, type:

```
m mserver register-system-service
```

The /com/merced/server/MServerMainService component determines the service name that the command registers in Windows. The default service name is com_merced_server_MServerMainService, created by replacing the directory slashes in the component name with underscores. If desired, you can change the service name by setting the **systemServiceName** property of the MServerMainService component. Remember that no two services can have the same name.

For each proxy, register one system service on its node. In a command line for the node, type:

```
m "(creator:/system/MyProxy)" register-system-service
```

where */system/MyProxy* is the proxy component.

The proxy component determines the service name that the command registers in Windows. In this case, the default service name is system_MyProxy, created by replacing the directory slashes with underscores. You can change the service name by setting the **systemServiceName** property of the proxy component. Remember that no two services can have the same name.

NOTE: Proxies create a second "helper" service that you can ignore. For example, if the service name is system_MyProxy, then the helper service name is system_MyProxy_helper.

See Appendix C, "Commands," for more information about these commands.

After registering a service, start the service to run the corresponding engines or proxy. You can use **Services** to start the service. To run a system service as a particular user, use **Services** to change the logon information after registering the service.

Unregistering

Unregistering a system service removes the service from the Windows registry. Before unregistering a system service, stop that service, if it is running.

To unregister the system service for engines on a node, open a command line on the node and type:

```
m mserver unregister-system-service
```

To unregister the system service for a proxy, open a command line on the node and type:

```
m proxy component unregister-system-service
```

where *proxy component* is the proxy component specified as a BEL (bean expression language) expression.

See Appendix C, "Commands," for more information about these commands.

Before re-registering a service, you might need to reboot the machine.

Using m mserver command

On Unix machines, the m mserver command can independently control each engine and proxy; you use the m mserver command to start and stop engines and proxies. On Windows machines, you can use the m mserver command to start and stop engines, but not proxies. Use the appropriate command syntax:

- To start an engine or proxy, open a command window on a node where the engine or proxy is configured, and type:

  ```
  $MERCED_HOME/bin/m mserver start server_name
  ```

where *server_name* is the name of the engine or proxy, specified in its configuration file.

- To stop an engine or proxy, open a command window on a node where the engine or proxy is configured, and type:

  `$MERCED_HOME/bin/m mserver stop` *server_name*

  where *server_name* is the name of the engine or proxy, specified in its configuration file.

For more information about the m mserver command, see Appendix C, "Commands."

In Unix systems, you can use cron to manage engines and proxies with the m mserver command.

In Windows systems, you can use scheduled tasks to manage engines that you run with the m mserver command. For example, you can create scheduled tasks that start automatically at system start up.

Scheduled tasks

Scheduled tasks run executable files or commands at selected times or at your prompt. Access scheduled tasks by selecting **Scheduled Tasks** in **Control Panel**.

You can use scheduled tasks to schedule restarts of engines and proxies. Typically, two tasks are involved: a stop task and a start task. After stopping an engine, wait ten minutes before starting it again. Use a command consistent with your method for running engines and proxies:

- With m mserver command — Use the m mserver command to stop the engine and to start it again. See Appendix C, "Commands," for details about this command.
- With Windows system services — Use the net stop command to stop the service and use the net start command to start the service. Run these commands by typing `net stop` *service_name* or `net start` *service_name* where *service_name* is the name of the registered service.

You can also use scheduled tasks to manage engines that you run using the m mserver command. You can create a scheduled task that starts automatically at system start up, or you can create a disabled scheduled task that you start manually, avoiding command lines or memorization of precise commands. In the task's properties window, you can specify the user that runs the task (the engine).

NOTE: Do not schedule two scheduled tasks to run at exactly the same time. Due to operating system limitations, tasks scheduled at the same time might not run.

Log files

When an engine or proxy starts, it creates a new log file. By default, engines write logs to:

`MERCED_HOME/var/`*`engine_name`*`/logs/`*`date`*`.log`

where *engine_name* is the name of the engine (defined by the component's **name** property) and *date* is the date the engine started.

By default, proxies write logs to:

`MERCED_HOME/var/`*`proxy_name`*`/logs/`*`date`*`.log`

where *proxy_name* is the name of the engine (defined by the component's **name** property) and *date* is the date the proxy started.

These log files contain information to help you troubleshoot the system. See "Debugging" on page 108 for information about reading and analyzing log files.

As part of maintaining the application server machine, you must ensure that log files remain a reasonable size and do not take up too much disk space. See "Restarting engines and proxies" on page 97 and "Managing log files" on page 98 for more information.

Database server machines

This section discusses how to set up the database server machine for Merced Performance Suite. Suggested configuration settings vary by DBMS (database management system):

- Oracle
- DB2 Universal Database
- SQL Server

For suggestions about maintaining the database server machine, such as recommendations for backups, see "Maintaining database server machines" on page 100.

Oracle

For creating a target Oracle database, Merced offers recommendations regarding:

- Physical structure
- Initialization file
- User tablespace

Physical structure

To improve performance, Merced suggests that you configure separate, sequestered physical partitions for:

- Redo logs
- Archive logs
- Rollback or UNDO tablespace
- Temp tablespace
- Default user tablespace
- System tablespace

Initialization file

Update the default init.ora file based on your hardware specifications and system usage. Merced offers several suggested settings.

Suggested settings for init.ora

Parameter	Value
db_files	200
db_file_multiblock_read_count	16
shared_pool_size	800M
log_checkpoint_interval	10000
processes	200
db_writer_processes	3
parallel_automatic_tuning	true
parallel_threads_per_cpu	*number of CPUs*
parallel_adaptive_multi_user	true
parallel_max_servers	*two times number of CPUs*
timed_statistics	true
db_block_size	8192
db_file_multiblock_read_count	16
hash_join_enabled	TRUE
java_pool_size	0
job_queue_processes	10
open_cursors	300
pga_aggregate_target	256M
query_rewrite_enabled	'FALSE'
star_transformation_enabled	'FALSE'
undo_management	'AUTO'
workarea_size_policy	'AUTO'
aq_tm_processes	0
db_cache_size	512M
shared_pool_reserved_size	80M

User tablespace

Create exactly one user and one user tablespace for Merced Performance Suite. Assign the created tablespace as the user's default tablespace. Merced Performance Suite does not use separate tablespaces for indexes, large tables, or any other subcategory of database objects. Merced suggests several settings for configuring the user's default tablespace.

Recommended settings for user's default tablespace

Setting	Value
Tablespace attributes	Locally managed, Uniform extents of 256k
Segment Space Management	Automatic
Enable Logging	Yes
Block Size	8192

Give the Merced Performance Suite user permission to create and drop schema objects, and to read and write those objects. This usually equates to granting CONNECT and RESOURCE roles. If granting the RESOURCE role creates problems during imports, revoke the UNLIMITED TABLESPACE system privilege.

DB2 Universal Database

For creating a target IBM DB2 Universal Database (DB2 UDB), Merced offers recommendations regarding:

- Registry variables
- Configuration variables
- Tablespace configuration
- User privileges

Remember that a new database instance requires a comprehensive setup and testing process.

Registry variables

Suggested registry settings for DB2 UDB

Registry setting	Value
DB2_SKIPINSERTED	ON
DB2_SKIPDELETED	ON
DB2_STRIPED_CONTAINERS	ON
DB2_HASH_JOIN	YES
DB2MEMDISCLAIM	YES
DB2COMM	TCPIP
DB2_PARALLEL_IO	*
DB2AUTOSTART	YES

Configuration variables

Suggested configuration variables for DB2 UDB

Variable	Value
LOGFILSIZ	15000
LOGPRIMARY	20
LOGBUFSZ	256
LOCKLIST	50000
MAXLOCKS	80
SORTHEAP	4096
STMTHEAP	8192
UTIL_HEAP_SZ	65424
STAT_HEAP_SZ	5132
MAXFILOP	1000
APPLHEAPSZ	4096
DBHEAP	6000
APP_CTL_HEAP_SZ	256
NUM_IOCLEANERS	7
NUM_IOSERVERS	16

Tablespace configuration

Create four buffer pools to support the tablespaces that Merced Performance Suite uses. These buffer pools should not use extended storage.

Buffer pool	Size	Page size
4K	98304	4096
8K	32768	8192
16K	12288	16384
32K	4096	32768

Merced Performance Suite uses three types of tablespaces—regular, user temporary, and system temporary. For each type of tablespace, create a tablespace for each buffer pool—4K, 8K, 16K and 32K page size. In other words, create:

- Four regular tablespaces managed by database (DMS)
- Four user temporary tablespaces managed by system (SMS) with prefetch size automatic
- Four system temporary tablespaces managed by system (SMS)

User privileges

Database users created for Merced Performance Suite must have:

- CREATEIN, DROPIN, and ALTERIN privileges on the database schema

  These privileges allow Merced Performance Suite to create, drop, and alter tables, indexes and views. If your database schema and database user are both named "MERCED," you can grant these privileges with the following command:

  ```
  GRANT CREATEIN, DROPIN, ALTERIN ON SCHEMA MERCED TO
  USER MERCED
  ```

- CONTROL privilege on tables and indexes

  CONTROL privilege allows Merced Performance Suite to initiate computation of database statistics. The user that creates a table or index automatically has CONTROL privilege for that table or index.

SQL Server

For creating a target Microsoft SQL Server database, Merced offers recommendations regarding:

- Collation
- Null values

Collation

Merced Performance Suite is case-sensitive. In SQL Server, case sensitivity is determined by the collation type. The SQL Server default collation type is case-insensitive. Therefore, when installing SQL Server for use with Merced Performance Suite, you must select an alternate, case-sensitive collation type, such as SQL_Latin1_General_CP1_CS_AS.

After you have installed SQL Server and created a database, you can check the database's collation by examining the database properties with Enterprise Manager. If necessary, you can change the collation type for an individual database with the **alter database** command. For example, to change the collation of the database named MY_DB to SQL_Latin1_General_CP1_CS_AS, you would type:

```
alter database MY_DB collate SQL_Latin1_General_CP1_CS_AS
```

Null values

When configuring a SQL Server database, you must set one database property, ANSI_NULL_DEFAULT, to ON. You can use default settings for other properties.

Required SQL Server settings

Parameter	Value
ANSI_NULL_DEFAULT	ON

3 System Configuration

This chapter discusses properties files defined when configuring the Merced Performance Suite cluster. The Merced Client Services Group and the application project team create these properties files and include them in the application build. This is the preferred way to configure a cluster.

However, when necessary, you can make changes without changing the application build by inserting changed properties files in the MERCED_HOME/localconfig directory. Changed properties files must have the same name, including directories, as the file they replace or add to.

This chapter discusses:

- Cluster configuration
- Database configuration

Cluster configuration

To configure a Merced Performance Suite cluster, you must create properties files that define five types of components:

- Cluster component
- Node component
- Host component
- Proxy component
- Engine component

For information about setting ports when configuring these components, see "Setting ports" on page 49.

Cluster component

A cluster component defines a set of nodes (physical machines) that run together. These nodes run the engines and proxies.

Customer component: /system/Cluster

Property	Description
$class	Component's Java class. For cluster components, set to: `com.merced.server.Cluster`
$scope	Set to: `global`
nodes	List of node components that represent each physical machine that belongs in this cluster.
enforceLocalNodeOperationsOnly	Boolean flag that forces engines and proxies to run on nodes designated by their configuration. Set to: `true`

For example, suppose your cluster has three machines. The properties file that configures your cluster component might look like:

/system/Cluster.properties

```
$class=com.merced.server.Cluster
$scope=global

nodes=\
 (creator:Node1),(creator:Node2),(creator:Node3)

enforceLocalNodeOperationsOnly=true
```

Node component

A node component defines a physical machine that is part of a cluster. The node definition specifies the host information needed to contact the machine and a list of servers that run on the machine. These servers can be engines or proxies.

Customer component: /system/Node

Property	Description
$class	Component's Java class. For node components, set to: `com.merced.server.Node`
$scope	Set to: `global`
name	Name that identifies this node.
host	Host component that specifies how to find this particular machine.
servers	Engine and proxy components that run on this machine.

For example, suppose the machine designated as Node1 is described as the host Machine1 and runs two engines and one proxy. The properties file that configures this node component might look like:

/system/Node1.properties

```
$class=com.merced.server.Node
$scope=global

name=Node1

host=(creator:hosts/Machine1)

servers=\
  (creator:Proxy1),\
  (creator:Engine1),\
  (creator:Engine2),\
```

Host component

A host component defines how to find a particular machine (or node).

Customer component: /system/hosts/Host

Property	Description
$class	Component's Java class. For host components, set to: `com.merced.server.Host`
$scope	Set to: `global`
inetAddress	String that identifies how to find the physical server using a TCP protocol.

For example, suppose that on your network the string "machine1.company.com" finds the machine called Machine1 in your cluster. The properties file that configures this host component might look like:

/system/hosts/Machine1.properties

```
$class=com.merced.server.Host
$scope=global

inetAddress=machine1.company.com
```

Proxy component

A proxy component defines a server process that points browser sessions to engines defined in the cluster's nodes. After receiving a browser session request, the proxy queries the nodes in the cluster to find available engines. If the browser session has already used an engine, then the proxy points the browser session to the same engine. Performance is improved by using the same engine throughout a session. If a session is new, the proxy points the browser session to the next available engine.

Defining multiple proxies in a cluster allows a proxy to act as a failover layer in case a sibling proxy server fails. If you define multiple proxies, your cluster requires a load balancer.

Customer component: /system/Proxy

Property	Description
$class	Component's Java class. For proxy components, set to: `com.merced.apache.ApacheMainService`
$scope	Set to: `global`
name	Name that identifies this proxy. The m mserver command uses this name to identify the proxy. Name must be unique among all engines and proxies in the cluster.
webHttpPort	Port that the proxy listens to for messages from Web browsers. Depending on your configuration and needs, you might want to configure this port to an IANA standard port. See "Selecting user to run engine or proxy" on page 24 for more information. See "Setting ports" on page 49 for additional information about configuring ports.
proxyProtocol	Communication protocol the proxy uses. To enable the proxy to talk to each engine, set to: `ajp13`
envMap	*Optional*. Map of environment variables that the proxy uses. By default, proxy uses map set in the **defaultEnvMap** property of the /Configuration component. If not set there, proxy uses operating system values.
timeZone	*Optional*. Time zone that the proxy uses. By default, proxy uses value set in the **defaultTimeZone** property of the /Configuration component. If no value is set there, proxy uses operating system value by default.
systemServiceName	*Optional*. The system service name created when registering the proxy component as a service. See "Using system services" on page 25. If this property is not specified, the system service name is the same as the full component name, replacing directory slashes with underscores.

Customer component: /system/Proxy *(Continued)*

Property	Description
shouldEnforceLogFileMaxSize	*Optional*. Boolean that indicates whether proxy automatically creates new log file when size of log file exceeds **logFileMaxSize**. Default is false.
logFileMaxSize	*Optional*. Maximum log file size, in bytes. The proxy automatically creates a new timestamp-named log file when the log file exceeds this size and **shouldEnforceLogFileMaxSize** is set to true. Default is 400000000 (400 MB).

configurationFileArgs	*Optional*. Configuration file arguments.
	SSL encryption
	If you are using SSL to encrypt data as it travels over your network, add the properties that specify where you store the certificate files:
	`sslCertificateFile=`*path-to-cert-and-key-file*
	Specify the path relative to the MERCED_HOME directory.
	If you store the SSL certificate and private key in separate files, specify both locations:
	`sslCertificateFile=`*path-to-cert-file*`, \`
	`sslCertificateKeyFile=`*path-to-key-file*
	Adding these properties turns on SSL and turns off non-SSL access to the proxy, forcing users to use SSL.
	Test the configuration by visiting the proxy in your browser using the https: protocol scheme.
	See "Secure sockets layer (SSL)" on page 15 for more information about SSL.
	Custom Apache log files
	You can customize the format of Apache log files by specifying the desired structure:
	`accessLogFormat="`*apache format*`"`
	where *apache format* is a string that Apache interprets. See Apache documentation for information about specifying this string:
	http://httpd.apache.org/docs-2.0/mod/mod_log_config.html#formats

For example, suppose the proxy component that runs on Node1 listens to port 9786 and uses SSL encryption. The properties file that configures this proxy component might look like:

/system/Proxy1.properties

```
$class=com.merced.apache.ApacheMainService
$scope=global

name = Proxy1

webHttpPort = 9786

proxyProtocol = ajp13

configurationFileArgs+=\
  sslCertificateFile=apache/merced-wildcard-sslcert.pem
```

NOTE: You can create a custom error page, custom500.html, that the proxy displays when all engines in a cluster are down. To have the proxy display that page, configure the /com/merced/apache/ApacheMainService component: use the **configurationFileArgs** property to map `error500page` to the location of the custom500.html file.

Engine component

The engine component defines the core server process that runs the application and renders reports and Web pages. You must designate exactly one engine in each cluster as Admin Server.

Customer component: /system/Engine

Property	Description
$class	Component's Java class. For engine components, set to: `com.merced.server.MercedApplicationServer`
$scope	Set to: `global`
name	Name that identifies this engine. The m mserver command uses this name to identify the engine. The name must be unique among all engines and proxies in the cluster.
ajp13Port	Port that the engine listens to for messages from proxies. See "Setting ports" on page 49 for more information about setting ports.
httpPort	Port that the engine listens to for messages from Web browsers. When using a proxy, direct access to an engine through this port is unnecessary. However, administrators might find direct access to the engine that serves as Admin Server helpful. See "Setting ports" on page 49 for more information about setting ports.
shutdownPort	Port that the engine listens to for m mserver commands that stop the component. See "Setting ports" on page 49 for more information about setting ports. See Appendix C, "Commands," for more information about the m mserver command.

Customer component: /system/Engine *(Continued)*

Property	Description

javaVMOptions	Literal command line arguments for JVM startup. Several arguments are required:
	• Virtual machine mode
	Set the mode in which the engine operates. Merced recommends server mode, which optimizes overall performance. Enter:
	`"-server"`
	The default is client mode, which optimizes footprint size and startup time.
	NOTE: The first argument must indicate the mode.
	• Minimum heap size
	Set the minimum amount of memory the engine uses (the start-up memory requirement), with the Xms argument. For example, to start the engine with 1.5 GB of RAM, enter:
	`"-Xms1536m"`
	• Maximum heap size
	Set the maximum amount of memory the engine can use with the Xmx argument. Allocate as much memory as possible, limited by available machine memory and continuous memory block requirements. See "Hardware" on page 7.
	For example, to ensure the engine never attempts to consume more than 1.5 GB of RAM (a Windows limit), enter:
	`"-Xmx1536m"`
	The default value is 64 MB. This is inadequate for a Merced Performance Suite engine.
	NOTE: Merced recommends that you set the minimum and maximum heap size to the same value, allowing the engine to preallocate all necessary memory. Setting these to the same value increases predictability and can decrease startup time. The engine will fail to start if that memory is not available.
	For information about other arguments, see http://java.sun.com/docs/hotspot/VMOptions.html and http://java.sun.com/docs/hotspot/gc1.4.2/index.html.

Customer component: /system/Engine *(Continued)*

Property	Description
localconfig	*Optional*. File that supports Admin Server. For the engine in the cluster that serves as Admin Server, set to the Admin Server module: `file:lib/product.adminserver.jar` For other engines in the cluster, omit this property.
shortDescription	*Optional*. Short description of the engine. The user interface displays this description on some pages accessible from the Admin page.
shouldEnforceLogFileMaxSize	*Optional*. Boolean that indicates whether engine automatically creates new log file when size of log file exceeds **logFileMaxSize**. Default is false.
logFileMaxSize	*Optional*. Maximum log file size, in bytes. Engine automatically creates a new timestamp-named log file when the log file exceeds this size and **shouldEnforceLogFileMaxSize** is set to true. Default is 400000000 (400 MB).
envMap	*Optional*. Map of environment variables that the engine uses. By default, proxy uses map set by the **defaultEnvMap** property of the /Configuration component. If not set there, proxy uses operating system values.
timeZone	*Optional*. Time zone that the engine uses. The engine's time zone must match the database's time zone for reports to render properly. By default, engine uses value set in the **defaultTimeZone** property of the /Configuration component. If no value is set there, engine uses operating system value by default.

For example, suppose that Engine1, which runs on Node1, communicates with proxy components on port 9761, with Web browsers on port 9771, and with the m mserver command on port 9711; this engine is not the cluster's Admin Server. Because it

runs on a Windows machine, the engine can acquire no more than 1.5 GB of RAM. The properties file that configures this engine might look like:

/system/Engine1.properties

```
$class=com.merced.server.MercedApplicationServer
$scope=global

name = Engine1

javaVMOptions= "-server", "-Xms1536m", "-Xmx1536m"

ajp13Port=9761

httpPort=9771

shutdownPort=9711
```

Admin Server

Admin Server is a specially designated engine in a cluster. To avoid conflicts, some key tasks must be initiated exactly once per cluster. Admin Server initiates these tasks. These tasks include:

- File Sniffer — A process that searches for files and initiates DICE jobs that extract data from the file and load the data into the database.
- Temporal Property Cleaner — A process that updates cached "right now" values for temporal properties. This process is especially important if your data model contains temporal properties that change in the future.
- Login Ticket Sweeper — A process that deletes expired login tickets. To facilitate login failover, engines leave a login ticket in the database when someone logs in. Tickets expire after a period of inactivity.
- Monitors — Processes that watch for changes in the database and generate alerts.

When selecting an engine to be Admin Server, be aware of these constraints and requirements:

- Admin Server must be able to access directories where you place files for File Sniffer. On Windows systems, shared network drives are not always accessible.
- To upload files manually, you must access the File Sniffer administration pages through Admin Server. In this case, therefore, you need to access the Admin Server engine directly, rather than through a proxy.
- Tasks only run when Admin Server is running. You must ensure that these tasks do not conflict with Admin Server maintenance, such as with regularly scheduled engine restarts. Conflicts are important to avoid for tasks scheduled infrequently:
  - Temporal Property Cleaner — Configured by the /com/merced/metabeans/TemporalPropertyCleaner component
  - Login Ticket Sweeper — Configured by the /com/merced/user/failover/SweepExpiredTicketsTask component

➢ To configure Admin Server

1. Select a single engine in the cluster that will serve as Admin Server.
2. When configuring that engine component, set the **localconfig** property to `file:lib/adminserver.jar`.

Setting ports

When defining engine and proxy components, you specify the ports that the component listens to. The most familiar port is the proxy's Web HTTP port, which is the port a user's browser sends application requests to. (In clusters without a proxy, this is the engine's HTTP port.) See the section about each component for a

list of the ports that you must configure for that component. You can set the required ports explicitly when you configure the component.

Alternatively, you can define a base port for the component. Assigning a base port triggers a default port assignment strategy: Merced Performance Suite automatically assigns the component a set of ports relative to the base port. The default port assignment strategy sets port values for ports not set explicitly in any configuration layer.

You can also define a server offset port for each component. The system adds the server offset port to the base port to derive port assignments. You can use the base port to define a unique base for the cluster and use the server offset port to define a unique port for each component in that cluster. In this case, each component would have a unique set of ports under the umbrella port associated with the cluster.

Property	Description
basePort	Port from which other ports are defined. Define this port for a proxy or engine component to make the component use the default port assignment strategy. You might select a unique number for each cluster.
serverOffsetPort	Integer that the system adds to the **basePort** when determining other port values for a component. You might select a unique number for each engine and proxy in the cluster. Defaults to zero.
shutdownPort	Port that the proxy or engine component listens to for m mserver commands to stop the component. Defaults to 10 + basePort + serverOffsetPort.
telnetPort	Port that the proxy or engine listens to for advanced debugging commands. Also called the Admin port. Defaults to 50 + basePort + serverOffsetPort.

Property	Description
ajp13Port	Port that the engine listens to for messages from a proxy. Defaults to 60 + basePort + serverOffsetPort.
httpPort	Port that the engine listens to for messages from Web browsers. Defaults to 70 + basePort + serverOffsetPort.
webHttpPort[a]	Port that the proxy listens to for messages from Web browsers. For SSL-enabled proxy servers, this is the SSL port. Defaults to 80 + basePort + serverOffsetPort.

a. Port that an end-user's browser sends application requests to.

For example, if you configure a component by setting the **basePort** property to 8000 and do not set the **serverOffsetPort** property, leaving it to default to 0, then Merced Performance Suite assigns the component the following ports:

- shutdownPort — 8010
- telnetPort — 8050
- ajp13Port — 8060
- httpPort — 8070
- webHttpPort — 8080

Turn a port off completely by setting its value to -1. For example, in a cluster configured with a proxy, you might set an engine's **httpPort** property to -1. In a cluster, engines need not listen for messages direct from browsers; instead, engines receive messages from a proxy, on the ajp13Port. Remember, however, that administrators might desire direct access to engines designated as Admin Server.

Setting the **basePort** property to -1 turns off the default port assignment strategy. Any ports set through the default port assignment strategy—ports not defined explicitly—are turned off.

Database configuration

To configure the database element of Merced Performance Suite, you must define a component representing the database connection and link that component to the repositories that the application uses. This section describes this configuration process. Topics include:

- Database password encryption
- Connection component
- Linking connection to repository
- Connection pools
- Testing database configuration

Database password encryption

You can configure Merced Performance Suite to require encrypted database passwords in properties files. Using encrypted database passwords offers an additional level of data security, preventing unauthorized persons from reading properties files to learn passwords, which can be used to access the database.

Merced Performance Suite supports encrypted database passwords in properties files that configure JDBC configuration components, the components Merced Performance Suite use to connect with databases. When you configure the application to require encrypted passwords, all passwords specified in JDBC configuration components—including passwords for the Merced Performance Suite database and passwords for source system databases—must be encrypted; any unencrypted passwords produce errors.

➢ To use database password encryption

1. Configure your application to require database password encryption.

   Set the **databasePasswordsAreEncrypted** property to true in the /Configuration component (a product component).

   /Configuration.properties

   ```
   databasePasswordsAreEncrypted=true
   ```

2. Specify only encrypted passwords in the properties files for JDBC configuration components.

   Use the m encrypt command to determine the encrypted text. In a command line on your application server machine, type:

   `$MERCED_HOME/bin/m encrypt` *mypassword*

   where *mypassword* is the clear text (decrypted) version of the password. If the password contains special characters, you might need to surround *mypassword* with quotes. See "Using quotes in commands" on page 133.

   NOTE: You cannot encrypt passwords exceeding 512 characters.

Connection component

You must configure a JDBC configuration component to represent the connection to the database that supports Merced Performance Suite.

Customer component: /com/merced/service/jdbc/config/JdbcConfiguration

Property	Description
$class	Component's Java class. For JDBC configuration components, set to: `com.merced.service.jdbc.config.JdbcConfiguration`
$scope	Set to: `global`
user	Database account username.
password	Database account password.
url	The JDBC URL for the database connection.

databaseFlavor	Component that specifies database type.
	For IBM DB2 UDB databases, enter:
	`/com/merced/service/jdbc/config/UDB8`
	For Microsoft SQL Server databases, enter:
	`/com/merced/service/jdbc/config/MSSql`
	For Oracle 9i databases, enter:
	`/com/merced/service/jdbc/config/Oracle`

databaseName	The name of the database.
defaultSchema	*Optional.* Name of the schema where Merced Performance Suite stores its database objects. Only DB2 UDB databases use this property. If omitted, DB2 UDB sets the schema to the username, by default. This property is useful in development environments sharing a single database server machine. Rather than creating separate DB2 UDB users, you can create a single user and separate schemas.

Typically, you place all JDBC configuration components in the /com/merced/service/jdbc/config part of the Creator namespace. However, this is not required; you can place JDBC configuration components in any namespace.

For example, suppose you are configuring an Oracle database. The URL to reach this database is "jdbc:oracle:thin:@sampledb.mercedsystems.com:1521:MYDB," the user name is "foo," and the password is "bar." The properties file for its JDBC configuration component might look like:

/com/merced/service/jdbc/config/SampleJdbcConfiguration.properties

```
#
# SampleJdbcConfiguration
#
$class=com.merced.service.jdbc.config.JdbcConfiguration
$scope=global

# databaseFlavor
databaseFlavor=(creator:Oracle)

# user
user=foo

# password
password=8C0F8620415BAA31F1ACB18479401E5C

# url
url=jdbc:oracle:thin:@sampledb.mercedsystems.com:1521:MYDB
```

Linking connection to repository

The application defines several metabean repositories. The application uses these repositories to store data in—and retrieve data from—database tables. Each repository needs a connection to link to the database. After creating a JDBC configuration component, register it with the system, creating a named connection that repositories can use to link with the database.

Register your JDBC configuration component using the /Configuration component. Edit the **jdbcConfigurations** property to add the desired JDBC configuration component for each named connection. Typically, repositories use one of two named

connections: **datamartRepository**, which most repositories use to access the database for standard uses, or **testRepository**, which some repositories use for regression tests.

For example, to register the SampleJdbcConfiguration component as the database connection for both data mart repositories and test repositories, modify the properties file for the /Configuration component:

/Configuration.properties

```
#
# Configuration
#
# jdbcConfigurations
jdbcConfigurations+=\
 datamartRepository=\
  (creator:/com/merced/service/jdbc/config/SampleJdbcConfiguration),\
 testRepository=\
  (creator:/com/merced/service/jdbc/config/SampleJdbcConfiguration)
```

Connection pools

Opening connections to a database can increase transaction time and hinder performance. A connection pool manages a set of database connections for an engine or a job, allowing that engine or job to reuse connections without opening new ones.

Merced Performance Suite automatically associates the JDBC configuration component with the ConnectionPool component (/com/merced/datamart/repository/ConnectionPool) for each engine. The ConnectionPool component has four key properties.

Product component: /com/merced/datamart/repository/ConnectionPool

Property	Description
minCount	The minimum number of connections that must be opened. Default value is 0.
maxCount	The maximum number of connections that an engine will open. Default value is 50.
maxIdle	The maximum number of idle connections before the connection pool starts closing connections, one per minute. Must be less than or equal to **maxCount**. A negative value turns off idle connection checking. Default value is 10.
evictionPeriod	The number of milliseconds between checks for idle connections. If the number of idle connections exceeds **maxIdle** for two checks in a row, one connection is closed per check. Connection pool does not check for idle connections if **maxIdle** is less than 0. Default value is 60000 (which is one minute).

You might consider changing the **maxcount** property. Databases often limit the number of connections. You must ensure that your database supports the maximum number of connections that might be opened simultaneously by all connection pools. A worst-case estimate is that all engines, DICE, and CubePop open the maximum connections simultaneously. In this case, you can estimate the maximum connections that must be supported by your database as:

(number of engines + 2) * **maxcount**

Testing database configuration

You can use the jsql command to test JDBC configuration components. See Appendix C, "Commands," for a description of this command.

4 Authentication

Authentication is the process that verifies a user's identity. Authorization is the process that verifies a user's permission to access the application. Merced Performance Suite implements authorization with authentication. The system supports traditional authentication, presenting users a Login page where they enter a user name and password, and single sign-on authentication, identifying users based on their machine login and signing them in without requiring them to visit the Login page.

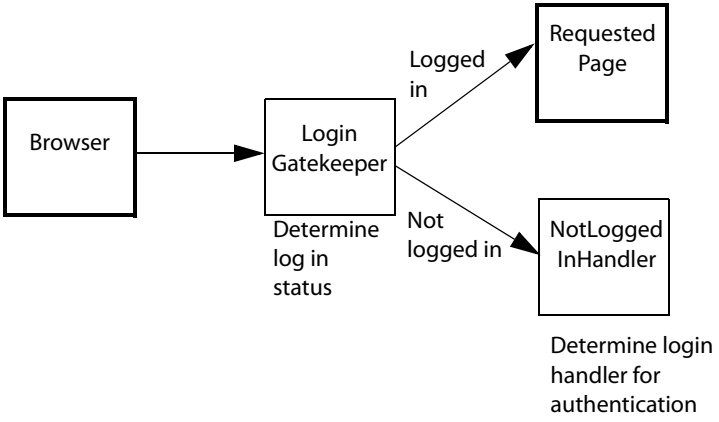
This chapter describes the types of authentication that Merced Performance Suite supports. Topics include:

- How authentication works
- Standard authentication
- LDAP authentication
- NTLM single sign-on authentication
- SPNEGO single sign-on authentication
- WebSEAL single sign-on authentication

How authentication works

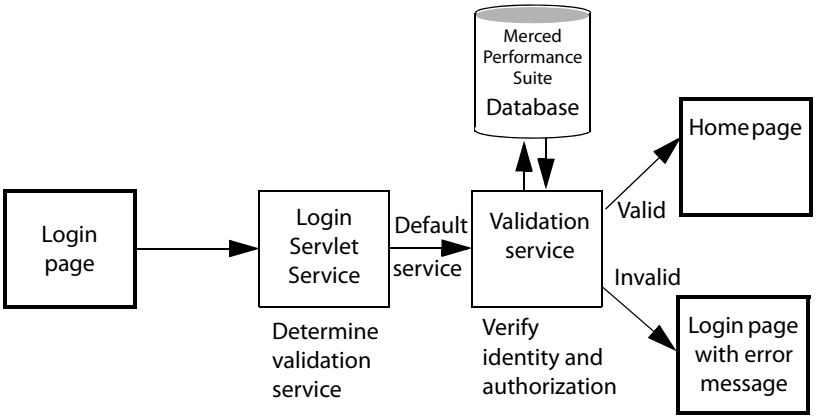
Authentication verifies a user's identity. Authorization verifies a user's permission to use Merced Performance Suite. Before logging a user in, Merced Performance Suite authenticates and authorizes the user. Based on application configuration, Merced Performance Suite uses different methods.

All requests from a browser session to an engine pass through the LoginGatekeeper component (/com/merced/user/servlet/LoginGatekeeper). If the user is logged in, LoginGatekeeper directs the session to the requested page; if not, LoginGatekeeper uses the **notLoggedInHandler** property to determine the appropriate login handler. The login handler specifies the method for authenticating users who are not logged in.



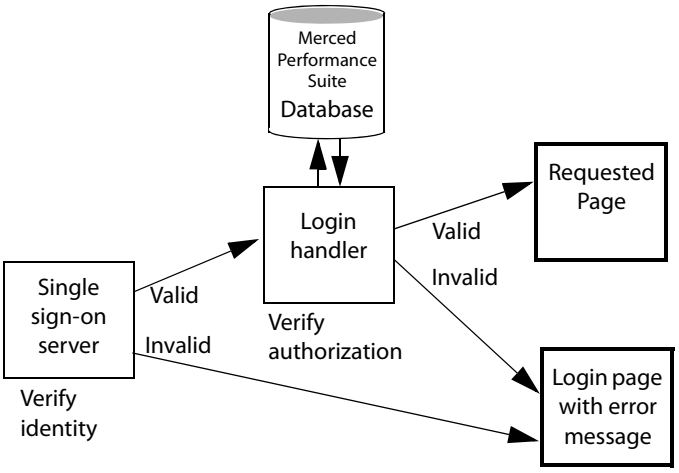
The default login handler uses traditional authentication and directs users who are not logged in to the Login page. Once users enter their user name and password, the engine checks the LoginServletService component (/com/merced/user/servlet/LoginServletService) to find a validation service. The validation service specifies how to verify a user's identity and authorization. The default validation service verifies identity and authorization by comparing the entered user name and password to those in the Merced Performance Suite database. Other validation

services might only verify user name or might use an external server, such as an LDAP server, to verify particular user names and passwords.



Other login handlers might make use of single sign-on authentication. In most single sign-on methods, the engine passes information from the browser session to an external single sign-on server, which verifies the user's identity. The single sign-on server passes the user name back to the engine, which verifies that the user is authorized to use Merced Performance Suite by comparing the user name to those in the Merced Performance Suite database. If the single sign-on server fails to

verify the user's identity or the engine fails to verify the user's authorization, the engine returns the user to the Login page with an appropriate error message.



Environments using WebSEAL authentication verify user identity before browser requests reach the Merced Performance Suite cluster. For details, see "WebSEAL single sign-on authentication" on page 88.

Standard authentication

Standard authentication is a traditional authentication method. Users who are not logged in are directed to the Login page, where they enter their user name and password. The system compares the entered information to the information stored internally—to the encrypted password stored in the Merced Performance Suite database. (System administrators maintain passwords through DICE or Employee Manager.) A successful match indicates both authentication and authorization, and the system logs in the user and displays the Home page. After a period of inactivity, a user's session expires, requiring the user to log in again.

This section discusses:

- Implementing standard authentication
- Creating dispatching validation services

Implementing standard authentication

Merced Performance Suite uses the LoginServletService component (/com/merced/user/servlet/LoginServletService) to determine how to validate users. LoginServletService specifies a validation service component, which describes the method engines use to validate users from the Login page. Specify the validation service in the **validationService** property.

Product component: /com/merced/user/servlet/LoginServletService

Property	Description
validationService	The name of the validation service the system uses to authenticate a user's log in.

Validation services

Merced Performance Suite is preconfigured with several validation services in /com/merced/user/validation:

- MercedValidationService — Uses the encrypted password stored in the database for authentication.
- AlwaysTrueValidationService — Allows users to log in as any valid user without entering a correct password.

  NOTE: Use this service with caution. It is designed for use in a development environment.

- AdminDispatchingValidationService — Uses a special validation service for the admin user name and a default validation service for other users. See "Creating dispatching validation services" for more information about this type of validation service.

  Initially, this service is set to use the MercedValidationService for both cases.

The default system configuration uses MercedValidationService as the validation service. You can create other validation services, such as dispatching validation services.

Creating dispatching validation services

Dispatching validation services specify a default validation service for most users and an override validation service for particular users. For example, an organization might want most users authenticated using LDAP (lightweight directory access protocol) but use the standard database-backed MercedValidationService to authenticate the person logging in as admin. You can define a dispatching validation service that meets your organization's needs.

Define a dispatching validation service by creating a corresponding component.

Customer component: Dispatching validation service

Property	Description
$class	The component's Java class. For dispatching validation service components, set to: `com.merced.user.validation.DispatchingValidationService`
$scope	Set to: `global`
defaultValidator	The validation service an engine uses when an override is not specified for a particular user name.
usernameProperty	Set to: `username`
dispatchMap	The override validation service for particular users. Enter as: *username* `=` *validation service*

For example, suppose you want a dispatching validation component that logs in three developers (Adam, Bob, and Carl) with the "MercedLdapValidationService" and logs in other users

with the default MercedValidationService. The properties file that configures this dispatching validation service component might look like:

/com/merced/user/validation/MercedLdapDispatchValidationService.properties

```
# Dispatching Validation Service
#
# Specific “developer” usernames are dispatched to the LDAP validator
# All others go to the Merced default (MUser) validator
#
#$Id: MercedLdapDispatchValidationService.properties,v 1.1 2003/03/06 $
#
 $class=com.merced.user.validation.DispatchingValidationService
 $scope=global

 defaultValidator=(creator:MercedValidationService)

 usernameProperty=username

 dispatchMap=\
   adam=(creator:ldap/MercedLdapValidationService),\
   bob=(creator:ldap/MercedLdapValidationService),\
   carl=(creator:ldap/MercedLdapValidationService),\
```

LDAP authentication

LDAP (lightweight directory access protocol) authentication is a traditional authentication method. With LDAP, administrators maintain passwords on a separate LDAP server. Like standard authentication, users who are not logged in are directed to the Login page, where they enter their user name and password. In this case, however, the system compares the entered information to the information stored on the LDAP server to authenticate the user.

After authentication, the system compares the user name to the Merced Performance Suite database to authorize the user. After authorization, the system logs in the user, displaying the Home page. After a period of inactivity, a user's session expires, requiring the user to log in again.

This section discusses:

- Configuring LDAP validation service
- Using SSL with LDAP validation
- Implementing LDAP authentication

Configuring LDAP validation service

To use LDAP authentication, Merced Performance Suite requires a component called an LDAP validation service. This component defines the properties required for LDAP authentication. An LDAP validation service verifies a user's identity by checking the user name and password that a user enters with the information maintained in the separate LDAP server. The LDAP validation service checks by attempting to log in to the LDAP server.

There are two types of LDAP validation service components:

- Simple LDAP validation service
- Lookup LDAP validation service

Simple LDAP validation service

Use the simple LDAP validation service component when the distinguished name that the service uses to log in to the LDAP server follows an identical pattern for all users, differing only by the user ID. This service logs in to the LDAP server directly as the user attempting log in, using the specified distinguished name pattern and user-entered password.

The simple LDAP validation service component specifies properties required for LDAP authentication.

Customer component: Simple LDAP validation service

Property	Description
$class	The component's Java class. For LDAP validation service components, set to: `com.merced.user.validation.ldap.SimpleLdapValidationService`
$scope	Set to: `global`
initialContexts	List of host names of potential LDAP servers. Merced Performance Suite connects with first available server. If you want the validation service to use the SSL protocol to communicate with the LDAP server, the host name must specify the ldaps protocol and SSL port. For example: `ldaps://server.mercedsystems.com:636/`
rootDN	Starting point for the LDAP directory structure.

Customer component: Simple LDAP validation service *(Continued)*

Property	Description
userDNPattern	Distinguished name that the LDAP server uses to identify a resource. Specify the full distinguished name, not just the part relative to **initialContext**. Use $1 as the place holder for the Merced Performance Suite user name when specifying the entire distinguished name. **NOTE:** The Merced Performance Suite user name must match the user name component of the distinguished name.
authenticationMethod	LDAP authentication method. Merced Performance Suite supports three methods: • simple • DIGEST-MD5 • CRAM-MD5
usernameProperty	Always enter: `username`

For example, a properties file that configures a simple LDAP validation service component might look like:

/com/merced/user/validation/ldap/MercedLdapValidationService.properties

```
$class=com.merced.user.validation.ldap.SimpleLdapValidationService
$scope=global

initialContexts=\
    "ldap://server1.mercedsystems.com:389/",\
    "ldap://another.server.com:389/"

rootDN="dc=mercedsystems,dc=com"

userDNPattern="uid=$1,ou=Person,dc=mercedsystems,dc=com"

authenticationMethod=simple

usernameProperty=username
```

Lookup LDAP validation service

Use the lookup LDAP validation service component when a single distinguished name pattern does not apply to all users. For example, if the LDAP distinguished name includes the user's group name and there is more than one group, then a single pattern cannot apply. This service connects to the LDAP directory using a Merced Performance Suite-specific user name and password, searches the LDAP directory for the desired user, retrieves the user's distinguished name, and then logs in to the LDAP server using that retrieved distinguished name and the user-entered password.

The lookup LDAP validation service component specifies properties required for LDAP authentication.

Customer component: Lookup LDAP validation service

Property	Description
$class	The component's Java class. For LDAP validation service components, set to: `com.merced.user.validation.ldap.LookupLdapValidationService`
$scope	Set to: `global`
initialContexts	List of host names of potential LDAP servers. Merced Performance Suite connects with first available server. If you want the validation service to use the SSL protocol to communicate with the LDAP server, the host name must specify the ldaps protocol and SSL port. For example: `ldaps://server.mercedsystems.com:636/`
rootDN	Starting point for the LDAP directory structure.
filterPattern	Filter pattern that the LDAP server uses to lookup a user. Use $1 as the place holder for the Merced Performance Suite user name. The typical pattern is: `"uid=$1"`

Customer component: Lookup LDAP validation service *(Continued)*

Property	Description
adminUserDN	Distinguished name of the LDAP user that Merced Performance Suite uses to connect to the LDAP directory. The lookup LDAP validation service connects to the LDAP directory as this user and searches for the user logging in to Merced Performance Suite. You can specify a global admin user or, preferably, an application-specific user. The specified user must have permission in the LDAP tree to perform user lookups for all user accounts that might log in to Merced Performance Suite. Consult the documentation for your LDAP implementation for information on how to grant the appropriate permission to the specified user.
adminPassword	Password for the admin or application-specific user specified in **adminUserDN**.
authenticationMethod	LDAP authentication method. Merced Performance Suite supports three methods: • simple • DIGEST-MD5 • CRAM-MD5
usernameProperty	Always enter: `username`

For example, a properties file that configures a lookup LDAP validation service component might look like:

/com/merced/user/validation/ldap/MercedLdapValidationService.properties

```
$class=com.merced.user.validation.ldap.LookupLdapValidationService
$scope=global


initialContexts=\
    "ldap://server1.mercedsystems.com:389/",\
    "ldap://another.server.com:389/"

rootDN="dc=mercedsystems,dc=com"

filterPattern="uid=$1"

adminUserDN="uid=admin,ou=Person,dc=mercedsystems,dc=com"

adminPassword="secret"

authenticationMethod=simple

usernameProperty=username
```

Using SSL with LDAP validation

LDAP validation services can use the SSL protocol to communicate with the LDAP server. For an LDAP validation service to use the SSL protocol successfully, you must:

- Configure the validation service to use SSL — When defining the LDAP server hosts with the **initialContexts** property, specify the "ldaps" protocol and the SSL port on the LDAP server.
- Set each Java Virtual Machine (JVM) on an application server machine to trust the SSL certificate from the LDAP server — Follow your organization's standard procedure for adding trusted certificates.

Implementing LDAP authentication

When using LDAP validation, Merced recommends that you configure your system so that the admin user name continues to authenticate using the MercedValidationService. Doing so ensures that admin can log in if the LDAP server is not running. You can do this by using a dispatching validation service, such as the preconfigured AdminDispatchingValidationService.

➢ To use LDAP validation

1. Configure an LDAP validation service component.

   In other words, create an LDAP validation service properties file that describes your LDAP system. See "Configuring LDAP validation service" on page 67.

2. Set the validation service to AdminDispatchingValidationService.

   In other words, in the LoginServletService properties file, set the **validationService** property to `AdminDispatchingValidationService`. See "Implementing standard authentication" on page 63.

3. Set the default validator for AdminDispatchingValidationService to the LDAP validation service component that you created for the customer.

   In other words, in the AdminDispatchingValidationService properties file (/com/merced/user/validation/AdminDispatchingValidationService.properties), set the **defaultValidator** property to the new component that you created in step 1. See "Validation services" on page 64.

NOTE: CSG or the application project team usually make these configuration changes in the customer layer, incorporating the configuration in the deployed application.

NTLM single sign-on authentication

NTLM (NT LanMan) is a single sign-on authentication method. Merced Performance Suite currently supports NTLM implemented in Microsoft Active Directory.

Using Active Directory and NTLM, Merced Performance Suite can validate a user's identity by setting up a secure connection between the user's browser and the Active Directory servers. After Active Directory authenticates a user's identity, the Merced Performance Suite engine verifies the authorization of users with this identity and logs the user in automatically, without requiring a password.

Before implementing NTLM authentication, you must know:

- The domain name where authentication will occur
- The hostname of the domain controller

➢ To implement NTLM authentication

1. Customize the NTLM authenticator component for your environment.

   Set the values for the **domainName** and **domainControllerHostname** properties.

Product component: /com/merced/user/validation/ntlm/NTLMAuthenticator

Property	Description
domainName	The name of the domain where authentication occurs.
domainControllerHostname	The name of the domain controller machine that performs NTLM authentication in your environment.

For example, the properties file that customizes the NTLMAuthenticator component might look like:

/com/merced/user/validation/ntlm/NTLMAuthenticator.properties

```
domainName=MERCED
domainControllerHostname=ad.mercedsystems.com
```

2. Set the login handler to NTLMNotLoggedInHandler.

   Override the **notLoggedInHandler** property of the LoginGatekeeper component to point to the NTLMNotLoggedInHandler component.

   For example, the properties file that customizes the LoginGateKeeper component (/com/merced/user/servlet/LoginGatekeeper) might look like:

/com/merced/user/servlet/LoginGatekeeper.properties

```
notLoggedInHandler=\
  (creator:/com/merced/user/validation/ntlm/NTLMNotLoggedInHandler)
```

3. If running Active Directory on a Windows 2003 server, configure that server machine and Active Directory to support Merced Performance Suite.

   See "To set required Active Directory settings for Windows 2003 servers" on page 76.

4. Configure client browsers by making the Merced Performance Suite URL a trusted site:

   a. From the **Tools** menu, select **Internet Options**.

   b. Go to the **Security** tab.

   c. Select the Trusted sites icon, then click the **Sites** button.

d. In the **Add this Web site to the zone** box, enter the domain for your application:

- If using SSL, keep the **Require server verification (https:) for all sites in this zone** check box selected, and type:

  `https://*.`*DOMAIN*

  where *DOMAIN* is the domain name for your installation.

- If not using SSL, clear the **Require server verification (https:) for all sites in this zone** check box, and type:

  `*.`*DOMAIN*

  where *DOMAIN* is the domain name for your installation.

e. Click **Add** to insert the entered site in the trusted list.

f. Click **OK** to accept the changes and close the Trusted sites window.

g. Click **OK** to close the Internet Options window.

➢ To set required Active Directory settings for Windows 2003 servers

1. Set the **requiresecuritysignature** registry value to zero.

   a. On the Windows 2003 server machine, click **Start > Run**.

   b. In the **Open** box, type `regedt32` and click **OK**.

   c. Navigate to the folder My Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters.

   d. Right-click **requiresecuritysignature** and select **Modify**.

   e. Set **Value data** to `0`.

   f. Click **OK**.

2. For each user account, select the required account options.
   a. In **Control Panel**, select **Administrative Tools > Active Directory Users and Computers**.
   b. In the domain that you are managing, navigate to **Users**.
   c. Double click the user name.
   d. On the **Account** tab, select the following check boxes under **Account options**:
      - **Store password using reversible encryption**
      - **Use DES encryption types for this account**
      - **Do not require Kerberos preauthentication**
   e. Click **OK**.

SPNEGO single sign-on authentication

SPNEGO (Simple and Protected GSS-API Negotiation Mechanism) is a protocol that supports single sign-on authentication. SPNEGO allows Internet Explorer to authenticate and authorize itself.

SPNEGO performs a Kerberos authentication via HTTP, and Internet Explorer passes the delegated credential to an engine. The engine verifies the authorization of users with this credential and logs the user in to Merced Performance Suite for this browser session. Merced Performance Suite currently supports SPNEGO implemented in Microsoft Active Directory.

This section discusses:

- Implementing SPNEGO authentication
- Debugging SPNEGO authentication

Implementing SPNEGO authentication

Implementing SPNEGO (Kerberos) authentication involves four key processes:

- Collecting data
- Registering with the domain controller
- Configuring the application
- Configuring the client

For information about debugging SPNEGO authentication, see "Debugging SPNEGO authentication" on page 84.

Collecting data

Prepare for installing Merced Performance Suite in an environment that uses SPNEGO authentication by collecting this data:

- HOST — The host portion of the URL that employees will use to access Merced Performance Suite. Obtain the fully qualified domain name, such as appserver.yourcompany.com.

  NOTE: If you are using a load balancer, the host will most likely be different than the machines that run the engines. The host name should be all lower case and not include any port information.

- DOMAIN — The fully qualified domain name of the Active Directory domain. The domain name should be all upper case.
- CONTROLLER — The fully qualified machine name of the Active Directory domain controller machine.
- USER — Select a user name and password that the Active Directory domain controller can use to issue Kerberos tickets for Merced Performance Suite. Pick a simple username, preferably in all upper case.

Registering with the domain controller

You must register Merced Performance Suite with the domain controller. To do so, you must complete several steps on your environment's Active Directory domain controller machine. Be sure to follow these instructions, as troubleshooting errors might be difficult.

➢ To register Merced Performance Suite with the domain controller

1. Check whether the domain controller has pre-registered the host machine URL for other applications.

   NOTE: IIS is a commonly registered application that can cause problems.

   a. Run this command on the domain controller:

      `setspn -L` *HOST*

      where *HOST* is the host name from the "Collecting data" process.

   b. Examine the results.

      Entries beginning with "http" are applications that have registered the host machine URL. You must remove these applications' registrations.

2. Remove the host machine URL registration for other applications.

   To remove an application's registration, type:

   `setspn -D` *http-entry HOST*

   where *http-entry* is a copy of a result from the `setspn -L` command above—a result starting with "http. "

   Run the `setspn -D` command for each "http" entry that the `setspn -L` command returned.

   Do not run the `setspn -D` command for any results that the `setspn -L` returned that do not start with "http."

3. Create and configure a user for SPNEGO authentication on the domain controller.

   Pick a simple username, in all caps, such as `MERCEDAPP`.

   Follow these steps:

   a. Create an Active Directory user with the username and password that you selected.

b. In Active Directory, select **Properties**, go to the **Account** tab, and set the proper configurations for this user.

- Clear the **User must change password on next login** check box.
- Select the **Password never expires** check box.
- Select the **Store password using reversible encryption** check box.
- Select the **Use DES encryption types for this account** check box.
- Select the **Do not require Kerberos preauthentication** check box.

4. Generate a Kerberos keytab file.

From the domain controller's command line, run this command:

`ktpass -princ HTTP/`*HOST*`@`*DOMAIN* `-mapuser` *USER*`@`*DOMAIN*

where *HOST*, *DOMAIN*, and *USER* are the values you obtained during the "Collecting data" process.

NOTE: The *DOMAIN* parameter must be all upper case, and the *HOST* parameter should be all lower case.

5. In Active Directory, re-enter the password for the user on the Users and Computers window.

Active Directory must store the password again after you set the "use reversible encryption" option and generate the Kerberos keytab file.

Configuring the application

You must configure the Merced Performance Suite application to support SPNEGO single sign-on.

➢ To configure Merced Performance Suite to support SPNEGO

1. Configure LoginGatekeeper.

   Set the **notLoggedInHandler** property of the /com/merced/user/servlet/LoginGatekeeper component to point at the SpnegoNotLoggedInHandler component.

   For example, you might create a properties file that modifies the existing component by simply setting this property.

/com/merced/user/servlet/LoginGatekeeper.properties

```
notLoggedInHandler=\
  (creator:/com/merced/user/validation/spnego/SpnegoNotLoggedInHandler)
```

2. Configure the login handler for SPNEGO—the SpnegoNotLoggedInHandler component.

   Use the information obtained when "Collecting data" and "Registering with the domain controller" to set four key properties of the /com/merced/user/validation/spnego/SpnegoNotLoggedInHandler component in your customer layer.

Product component: /com/merced/user/validation/spnego/SpnegoNotLoggedInHandler

Property	Description
controller	The fully qualified domain name of the Active Directory domain controller. This includes the domain portion of the fully qualified name.
domain	The domain where the controller resides. This must match the domain name portion of the controller property.
host	The host portion of the URL that browser sessions use to access the Merced Performance Suite application.
password	The password created for SPNEGO authentication on the domain controller. (See "Registering with the domain controller" on page 79.)

For example, the application project team might create a properties file that configures these properties in the customer layer. The properties file need not set the **$class** and **$scope** properties—the application layer configures these properties. The properties file that configures the SpnegoNotLoggedInHandler component might look like:

/com/merced/user/validation/spnego/SpnegoNotLoggedInHandler.properties

```
controller=adcontroller.LAB.MAGNETIC.COM

domain=LAB.MAGNETIC.COM

host=app.LAB.MAGNETIC.COM

password=password_1234
```

Configuring the client

Microsoft Internet Explorer version 6.0 supports SPNEGO. To enable SPNEGO single sign-on functionality in the browser, you must set some browser configuration options.

➢ To configure Internet Explorer to support SPNEGO single sign-on

1. Make the Merced Performance Suite URL a trusted site:
   a. From the **Tools** menu, select **Internet Options**.
   b. Go to the **Security** tab.
   c. Select the Trusted sites icon, then click the **Sites** button.
   d. In the **Add this Web site to the zone** box, enter the domain for your application:
      - If using SSL, keep the **Require server verification (https:) for all sites in this zone** check box selected, and type:

        `https://*.`*DOMAIN*

        where *DOMAIN* is the domain name for your installation.

- If not using SSL, clear the **Require server verification (https:) for all sites in this zone** check box, and type:

  **.DOMAIN*

  where *DOMAIN* is the domain name for your installation.

e. Click **Add** to insert the entered site in the trusted list.

f. Click **OK** to accept the changes and close the Trusted sites window.

g. Click **OK** to close the Internet Options window.

2. Enable integrated authentication:

a. From the **Tools** menu, select **Internet Options**.

b. Go to the **Advanced** tab.

c. Under **Security**, select the **Enable Integrated Windows Authentication (requires restart)** check box.

d. Click **OK**.

e. Exit Internet Explorer, and start a new Internet Explorer browser.

Debugging SPNEGO authentication

This section presents common error messages that appear in the Merced Performance Suite log files, along with a description of what might be causing the error messages and steps you can take to fix the problem.

IOException error message

```
IOException in Spnego authentication: java.net.UnknownHostException:
```

Cause: Most likely caused by incorrect setup of the SPNEGO authentication user on the Active Directory domain controller.

Solution: Recreate the user on the Active Directory domain controller. See "Registering with the domain controller" on page 79.

Encryption type not supported error message

```
   java.security.PrivilegedActionException:
java.lang.reflect.InvocationTargetException
   rootCause=GSSException: Failure unspecified at GSS-API level (Mechanism
level: KDC has no support for encryption type (14))
```

Cause: IIS or some other application requiring authentication has been registered previously for your host machine.

Solution: Remove other applications' host machine URL registrations from the Active Directory domain controller. See "Registering with the domain controller" on page 79.

Pre-authentication information invalid error message

```
   javax.security.auth.login.LoginException: Pre-authentication
information was invalid (24)
   rootCause=KrbException: Identifier doesn't match expected value (906)
```

- **Cause**: The password in the SpnegoNotLoggedInHandler component is configured incorrectly.

  Solution: Configure the SpnegoNotLoggedInHandler with the correct password. See "Configuring the application" on page 81.

- **Cause**: The clocks on the machines running the Merced Performance Suite engine and the machine running the domain controller are not synchronized. Typically, the clocks must be within five minutes of each other for SPNEGO to work.

  Solution: Synchronize the clocks on the affected machines.

Client not found in Kerberos database error message

```
   javax.security.auth.login.LoginException: Client not found in Kerberos
database (6)
   rootCause=KrbException: Identifier doesn't match expected value (906)
```

Cause: Either the hostname or port is misconfigured in the SpnegoNotLoggedInHandler.

Solution: Configure the SpnegoNotLoggedInHandler with the correct hostname and port. See "Configuring the application" on page 81.

Controller must end with domain error message

```
   javax.servlet.ServletException: init-param "controller" must end with
domain
```

Cause: The controller configured in the SpnegoNotLoggedInHandler component does not use the domain configured in the domain property.

Solution: Configure the SpnegoNotLoggedInHandler with the correct controller and domain. See "Configuring the application" on page 81.

Unknown host error message

```
   javax.security.auth.login.LoginException: your.host.here:
your.host.here
   rootCause=java.net.UnknownHostException: your.host.here:
your.host.here
```

- **Cause**: The SpnegoNotLoggedInHandler component incorrectly configures the controller or domain. Note that 100% of the logins will fail in this case.

  Solution: Configure SpnegoNotLoggedInHandler with the correct controller and domain. See "Configuring the application" on page 81.

- **Cause**: An Internet Explorer 6.0 client is accessing a Merced Performance Suite engine without the **Enable Windows Integrated Authentication** option enabled. If this is the case, some login attempts will fail and produce this error message, while other login attempts (using correctly-configured client browsers) will succeed.

  Solution: Configure the Internet Explorer client correctly. See "Configuring the client" on page 83.

- **Cause**: Your host is not set as a trusted site in the client browser. If this is the case, a login window will pop up when the user first tries to access Merced Performance Suite. Users with correctly configured browsers will be able to log in without problems.

  Solution: Configure the Internet Explorer client correctly. See "Configuring the client" on page 83.

- **Cause**: The SpnegoNotLoggedInHandler component correctly configures the controller and domain, but the Merced Performance Suite engine cannot access the Active Directory domain controller.

  Solution: Restore connectivity between the Merced Performance Suite engine and the Active Directory domain controller.

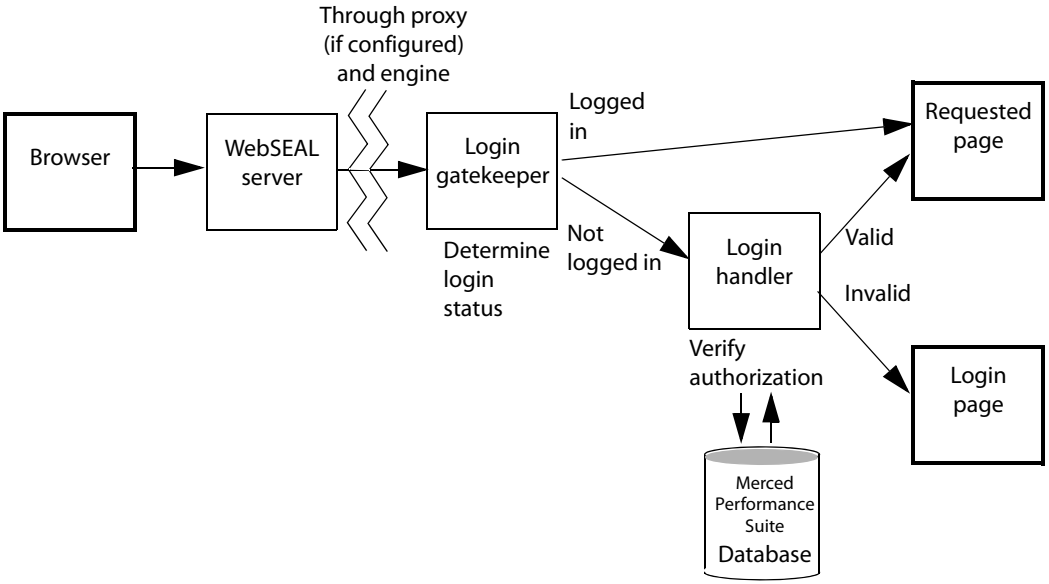
WebSEAL single sign-on authentication

WebSEAL is a Web server that can manage access to all of an organization's Web servers. A component of IBM Tivoli Access Manager, WebSEAL supports multiple authentication methods and provides single sign-on capabilities. WebSEAL acts as a reverse Web proxy—WebSEAL receives HTTP requests from a Web browser and delivers content from its own Web server or from configured back-end Web application servers, like Merced Performance Suite.

When implemented in an environment managed by WebSEAL, Merced Performance Suite assumes that all requests it receives are authenticated; Merced Performance Suite only authorizes the requests.

WebSEAL sends the user name in HTTP request headers. When you enable WebSEAL authentication in Merced Performance Suite, the login gatekeeper calls the WebSealLoginHandler component. This login handler verifies the user's authorization by comparing the user name in the request header to those in

the Merced Performance Suite database. If authorized, users are directed to the Home page; if not, they are directed to the Login page.



HTTP request headers are trivial to forge. You must secure your network to prevent forgery, ensuring the validity of the authentication mechanism and hence the security of the Merced Performance Suite application.

➢ To enable WebSEAL authentication

1. Secure your network to prevent forgery of HTTP headers.
2. Configure the /com/merced/user/servlet/LoginGatekeeper component to use WebSEAL.

   Set the **notLoggedInHandler** property to point to the WebSealLoginHandler component. For example, you might create a properties file that modifies the existing component by simply setting this property.

/com/merced/user/servlet/LoginGatekeeper.properties

```
notLoggedInHandler=\
  (creator:/com/merced/user/validation/webseal/WebSealLoginHandler)
```

3. If required, change the authentication token that the login handler searches for in the HTTP request header.

   The authentication token specifies the user name. By default, the authentication token is `iv-user`, the token used in most WebSEAL configurations.

   If WebSEAL uses a different authentication token, configure the same authentication token in Merced Performance Suite by changing the **webSealKey** property in the /com/merced/user/validation/webseal/WebSealLoginHandler component.

5 Monitoring

This chapter describes the maintenance and monitoring tasks that you must complete to ensure that Merced Performance Suite runs properly. This chapter discusses:

- Maintenance
- Automated monitoring and maintenance
- Debugging
- Troubleshooting

Maintenance

System administrators maintain the environment that supports Merced Performance Suite—this involves ensuring that server machines enable optimal performance, that connectivity is maintained between key processes and machines, and that any system maintenance does not conflict with scheduled processes.

This section discusses:

- Understanding system processes
- Maintaining application server machines
- Maintaining database server machines
- Maintaining connectivity

Understanding system processes

Merced Performance Suite is a complex system with many scheduled jobs that interact with other systems in your organization. An application server machine might run several processes, including:

- Application engines
- Application proxies
- CubePop jobs
- NextPop jobs
- DICE conversion jobs

By understanding what processes run, you learn what log files the system generates and the dependencies that exist between various server machines. For example, your organization might have a DICE conversion job that extracts data from your Avaya database at a particular time. With this knowledge, you can choose to reboot the Avaya database server at a time that does not impact the Merced Performance Suite DICE job.

Collect information from your application project team about which jobs run at which time. Helpful information includes:

- Machine
- User
- Task or job name
- Start time
- Approximate duration
- Frequency
- Command

For example, you might collect a file that looks like:

Sample data collected on system processes

Machine	Task Name	User	Start Time	Duration	Freq	Command
appsrv1	stopEng1	MENG	2:20 AM	1 min	Daily	m mserver stop eng1
	startEng1	MENG	2:30 AM	1 min	Daily	m mserver start eng1
	stopEngb	MENG	3:20 AM	1 min	Daily	m mserver stop engb
	startEngb	MENG	3:30 AM	1 min	Daily	m mserver start engb
appsrv4	stopEng2	MENG	10:30 PM	1 min	Daily	m mserver stop eng2
	startEng2	MENG	10:40 PM	1 min	Daily	m mserver start eng2
	OrgChartConv	MDATA	7:00 AM	30 min	Daily	m dice /com/ merced/dice/jobs/ conversions/ OrgChartConv
	OptimizeDB	MDATA	2:00 PM	30 min	Daily	m optimize-db
appsrv5	SamsonCubeJob	MDATA	12:15 PM	1 hour	Daily	m dice /com/ merced/dice/jobs/ SamsonCubeJob
	AudChunkConv	MDATA	8:15 PM	1 hour	SUN	m dice /com/ merced/dice/ conversions/ AudChunkConv

Sample data collected on system processes *(Continued)*

Machine	Task Name	User	Start Time	Duration	Freq	Command
	HrsChunkConv	MDATA	9:15 PM	3 hour	SUN	m dice /com/ merced/dice/ conversions/ HrsChunkConv
	SAPAttrConv	MDATA	1:00 AM	2 hour	THU	m dice /com/ merced/dice/jobs/ SAPAttrConv
	PDSJob	MDATA	1:05 AM	1 hour	Daily	m dice /com/ merced/dice/jobs/ PDSJob
	RemedyCubeJob	MDATA	1:10 AM	3 hour	Daily	m dice /com/ merced/dice/jobs/ RemedyCubeJob
	stopEng3	MENG	1:15 AM	1 min	Daily	m mserver stop eng3
	startEng3	MENG	1:25 AM	1 min	Daily	m mserver start eng3
	SegCubeJob	MDATA	3:00 AM	20 min	Daily	m dice /com/ merced/dice/jobs/ SegCubeJob
	AudCubePopJob	MDATA	3:30 AM	20 min	Daily	m dice /com/ merced/dice/jobs/ AudCubePopJob
	HrsCubePopJob	MDATA	4:00 AM	30 min	Daily	m dice /com/ merced/dice/jobs/ HrsCubePopJob
	StrmCubeJob	MDATA	4:45 AM	2 hour	Daily	m dice /com/ merced/dice/jobs/ StrmCubeJob

With this information, you know what processes run at what time. You can use this information to determine impacts produced by any server or network issues.

Maintaining application server machines

To maintain the application server machines and ensure an optimal environment for Merced Performance Suite, you must maintain adequate disk space, processing load, and memory.

This involves both monitoring the system and completing routine tasks. You can automate much of this monitoring. See "Automated monitoring and maintenance" on page 105 for suggestions.

This section discusses:

- Disk space
- Processing load
- Memory
- Restarting engines and proxies
- Managing log files

Disk space

Check the available disk space on the host. There should be enough space for all engines, proxies, and tools to write log files for several days. Archiving log files helps control disk space usage. See "Managing log files" on page 98.

Check that the sizes of the current log files are not near any platform-imposed maximum file size. Log files that exceed 1 GB can impact the performance of an engine or proxy. Restarting engines and proxies regularly helps manage log file size. See "Restarting engines and proxies" on page 97.

Processing load

Check the CPU load on the host. The CPU load should be close to historically typical loads. Often, this typical load is no more than 10 to 20% of the machine's maximum load. The machine must be able to accommodate the load at peak periods, such as the load generated by multiple queries when call center agents log in at the start of their shift. If Merced Performance Suite shares the host with other applications (not recommended), remember that those applications may also contribute to system load.

Memory

Check for memory usage at the OS level and at the Java Virtual Machine (JVM) level. You can use Heap Monitor to trace the application's memory usage.

OS memory

Check that there is enough memory available for all Merced Performance Suite engines and proxies to reach their configured maximum size. If there is insufficient memory, the OS may thrash and drastically reduce performance.

JVM memory

Check the memory consumption of the Java process as a percentage of the total allowed by your Merced Performance Suite configuration.

Heap Monitor can collect statistics about the amount of memory the system uses. Heap Monitor writes statistics to a metabean table, called MemoryProfile.

Heap Monitor statistics show used and available memory:

- Max heap size — Indicates the most memory the engine will use to store program objects (its heap)
- Alloc heap size — Indicates the amount of memory the engine is using presently to store program objects
- Free heap size — Indicates the amount of memory currently available on the application server machine

  NOTE: If "free heap size" is consistently less than 100 MB, you might have inadequate memory on the application server machine. If "free heap size" is zero, there is no memory available.

By default, Heap Monitor is disabled. If you encounter memory issues with your application, enable Heap Monitor and monitor memory usage. To enable Heap Monitor, set the **enabled** property to true in the /com/merced/service/vm/monitor/HeapMonitor component.

Restarting engines and proxies

In a production environment, Merced recommends nightly restarts of each engine and weekly restarts of each proxy. Restarting engines and proxies produces new log files for those engines and proxies. Regular restarts, therefore, can help maintain application performance and provide an automatic way to manage log files and keep log file size in check.

NOTE: Proxies continue writing to existing Apache log files after restarting. Therefore, you must prune Apache log files manually before restarting a proxy. See "Managing log files" on page 98.

In a cluster with a load balancer and multiple proxies and engines, regular restarts need not produce any system down time. You can stagger the restart of each engine and proxy. The load balancer can direct requests to proxies still running, and proxies direct requests to engines still running. Staggering restarts at least ten minutes apart allows the restarted engine or proxy to reach a steady state before the next shuts down and restarts.

In a cluster with a single proxy or a single engine, a regular restart produces a brief period of system inaccessibility while the engine or proxy restarts.

Restart engines and proxies during periods of light use, typically at night. After stopping an engine, wait ten minutes before starting it again. You can schedule automatic jobs that restart engines and proxies, using cron in Unix systems, or scheduled

tasks in Windows. See "Running engines and proxies" on page 24. You can also use a third-party scheduling tool, like Autosys.

NOTE: Do not schedule restart of the engine designated as Admin Server when key tasks, like Temporal Property Cleaner, are scheduled to run.

Managing log files

You must manage the log files that Merced Performance Suite generates. When managing your log files, you must consider:

- File size
- Archiving
- Proxy log files

File size

Maintaining reasonably sized log files improves the usefulness of the log files and can improve application performance. Smaller log files are easier to search, for example.

Restarting engines and proxies regularly is one way to maintain the size of individual log files. If log files become too large, consider restarting the engine or proxy more frequently, or altering the **shouldEnforceLogFileMaxSize** and **logFileMaxSize** properties in your engine or proxy configuration to create new log files automatically (see "Cluster configuration" on page 38).

Archiving

Archiving log files ensures that old log files do not fill your disk. You must determine a strategy for removing and backing up files in the log directory. You might complete the task manually or you might schedule a job that automatically moves or removes log files after a certain number of days.

Merced recommends retaining log files for several months. These log files are often useful for system planning. For example, you can use these log files to determine trends in job duration and estimate data growth.

Proxy log files

In addition to the standard proxy log file, a proxy writes to Apache log files. Apache does not create unique log files each time the proxy starts. Therefore, you must proactively prune Apache log files. Merced recommends that you prune Apache log files during the proxy's regular restart. While the proxy is stopped, back up and delete the current Apache log files, preferably with an automated script.

➢ To prune Apache log files

1. Stop the proxy.
2. Zip the Apache log files.

   Found in the MERCED_HOME/var/*proxyX*/logs directory, where *proxyX* is the name of the proxy, there are four Apache log files:

   - access.log
   - error.log
   - mod_jk.log
   - system_service.log
3. Delete the original files.
4. Start the proxy.

Maintaining database server machines

Proper maintenance of database server machines can be complex. Refer to your DBMS documentation for complete requirements. At a minimum, you must maintain adequate tablespace and must back up the database at regular intervals to ensure an optimal environment for Merced Performance Suite.

For general database administration, Merced recommends that you use the tools your database vendor provides:

- For DB2 UDB, use IBM DB2 Control Center.
- For Microsoft SQL Server, use Enterprise Manager for administrative tasks and Query Analyzer for queries.
- For Oracle, use Enterprise Manager for administrative tasks. Many Oracle users find the third-party application, Golden, useful for queries.

Tablespace

Check that the account that Merced Performance Suite uses to log in to the database has sufficient tablespace. Data size will grow as more data is imported. Growth rates are tied to the DICE conversions that the cluster uses.

Also check engine logs for errors related to insufficient tablespace, especially temp and redo spaces in Oracle databases.

Prune the m_diceinvocation and m_execution system tables regularly to ensure proper system performance.

Database back up

Merced strongly recommends that you back up the database frequently. Though you can often reload data from source systems, you cannot reload data entered through forms or the Employee Manager—and some source systems only store several months of recent data, making reloading the entire history impossible. Specifically, Merced recommends nightly hot

backups with more frequent backups of the redo logs (called transaction logs in SQL Server databases). For example, you might back up the redo logs once per hour.

If possible, archive pre-imported data—that is, the data that your DICE conversions use as input. For example, if you have flat-file conversions that pre-aggregate data, archive the flat files. If DICE pulls data from a database, simply maintain that database. In addition, you might create backups that enable you to maintain a staging database.

Maintaining connectivity

You must maintain connectivity to all elements in your Merced Performance Suite cluster:

- Engines
- Proxies
- Database
- Load balancer

You can see an overview of your cluster's layout from the Admin page of the user interface. **Under Merced System Administration**, select **Merced Cluster Administration**, or go to the application URL, .../merced/admin/system-admin/cluster-admin.

Engines

Engines listen on one or more network ports. Verify connectivity to those ports. The ports configured for an engine might vary.

To find an engine's ports, open a command window on the machine running the engine and use the m bel command. Type:

```
$MERCED_HOME/bin/m bel "(creator:/system/engineX.portMap)"
```

where */system/engineX* is the engine component.

This command prints the configured ports. If the engine is configured using the base port strategy, this command might output:

```
{base=5801, http=5871, serverOffset=0, telnet=5851,
ajp13=5861, webHttp=5851, shutdown=5811}
```

See "Setting ports" on page 49 for a complete description of the base port strategy. Frequently used ports for engines include:

- HTTP — Used to communicate directly with browser sessions
- AJP — Used to communicate with proxies
- Shutdown — Used to shut down engine with m mserver command

To verify connectivity to a port, simply connect to the port. Use a command like telnet:

```
telnet host1 1234
```

To verify connectivity to an engine's HTTP port, you can use a browser session to connect directly to the engine, bypassing any proxies or load balancers. For example, if hostX is the engine's hostname and 1234 is the engine's Http port, open this URL in your browser:

```
http://hostX:1234
```

Proxies

By accessing the Merced Performance Suite application through a proxy, you can check connectivity to a proxy. There are four possible conditions:

- Proxy up, all engines up — Normal access
- Proxy up, all engines down — Custom error page that indicates the engines are unavailable

- Proxy up, at least one engine unresponsive — Normal access, no response, or error page indicating page cannot be displayed
- Proxy down — Error page indicating page cannot be displayed

By visiting special URLs not sent to engines, you can view detailed information about the proxies, regardless of the state of the engines.

URL[a]	Shows
/proxy-status	Load and status of individual threads of the proxy
/proxy-info	Detailed raw, potentially sensitive configuration information, usually only available when requested from localhost

a. The complete URL is http://*myproxy*:####/... where *myproxy* is the name of the proxy host and #### is the proxy's Web HTTP port.

Before visiting these URLs, make sure no other proxies or load balancers sit between your browser and the target proxy.

As with engines, you can use the telnet command to verify connectivity to all the ports that the proxy uses. To find a proxy's ports, open a command window on the machine running the proxy and use the m bel command to find the configured ports.

Type:

```
$MERCED_HOME/bin/m bel "(creator:/system/proxyX.portmap)"
```

where */system/proxyX* is the proxy component.

See "Engines" on page 101 for more information about this command and using the telnet command to verify port connection.

Database

Check the connectivity with the cluster database using the same credentials that Merced Performance Suite uses. You can use the jsql program, included with Merced Performance Suite. See "jsql" on page 135 for more information about using the jsql command.

Load balancer

Check that you can connect to Merced Performance Suite through the load balancer that browser sessions in your organization use to connect to Merced Performance Suite.

Automated monitoring and maintenance

You can set up jobs to automate Merced Performance Suite monitoring and maintenance tasks. This section provides suggestions for the types of jobs you might set up. This section discusses:

- Monitoring jobs
- Maintenance jobs

Monitoring jobs

To monitor the Merced Performance Suite cluster, you might set up automated jobs that help with:

- Verifying connectivity
- Checking database connections
- Searching log files

NOTE: Disabling automated monitoring jobs during routine maintenance tasks, like engine and proxy restarts, can prevent unwarranted warnings.

Verifying connectivity

You can create an automated job that verifies connectivity to the load balancer, proxies, and engines. See "Maintaining connectivity" on page 101 for details about how to check connectivity.

When creating the job to verify connectivity, you must determine:

- How often to run — Merced recommends that you run the automated job every ten minutes.
- What constitutes failure — Failure for some checks is indicated by an error page. Failure for other checks is indicated by no response for a selected amount of time.

- When to escalate — Failure might require immediate escalation to a person, such as if connection to a critical system like the load balancer fails. Or, you might require multiple failures before escalating, such as if connection to a single engine fails in a cluster with multiple engines.

Checking database connections

You can create an automated job that checks the number of database connections an engine uses. A large number of connections is often concurrent with slow response times, indicating overloaded engines and other problems.

When creating the job to check database connections, you must determine:

- How often to run — Merced recommends that you run this automated job every ten minutes.
- What constitutes failure — The maximum number of database connections that is acceptable for an engine might vary by your database configuration. See "Connection pools" on page 56.
- When to escalate — Merced recommends that the automated job escalate this issue to a person for attention following two failures.

Searching log files

You can create an automated job that searches log files for connectivity problems with the database. In particular, this job must search DICE logs (for conversion jobs and NextPop jobs) and CubePop logs. One of two strings indicate connectivity problems:

- `java.sql.DriverManager.getConnection` — Indicates a problem connecting with JDBC or ODBC
- `java.sql.SQLException: No more data to read from socket` — Indicates a lost connection

See "Debugging" on page 108 for more information about locating and reading log files.

When creating the job to check database connections, you must determine:

- How often to run — Merced recommends that you run this automated job every ten minutes.
- What constitutes failure — Database connectivity problems during DICE jobs or CubePop jobs are critical and often require manual intervention to enable the system to continue updating data. See "Troubleshooting tasks" on page 113. Database connectivity problems at other times impact the experience of those using the application.
- When to escalate — Merced recommends that the automated job escalate any database connectivity failures during DICE jobs or CubePop jobs to a person for attention.

Maintenance jobs

To support routine maintenance of Merced Performance Suite, you might create automated jobs that:

- Stop and restart engines each night. See "Restarting engines and proxies" on page 97 for implementation details.
- Stop proxies, prune Apache logs, and restart proxies each week. See "Restarting engines and proxies" on page 97 and "Managing log files" on page 98.
- Archive log files based on a strategy appropriate for your organization. See "Managing log files" on page 98.

Debugging

This section contains information useful when trying to debug an issue with Merced Performance Suite. In particular, it describes log files and techniques you might use for analyzing log files. Sections include:

- Locating log files
- Reading log files
- Analyzing log files

Locating log files

Log files are associated with the command used to run the job or process. Key log files that you might frequently use include:

- Engine and proxy — The m mserver command starts and stops each engine and proxy. Locate logs for the engine or proxy in the MERCED_HOME/var/*serverX*/logs directory, where *serverX* is the name of the engine or proxy. Proxies also have several Apache log files.
- DICE — The m dice command runs DICE jobs (DICE conversions and NextPop processes). Locate logs for this command in the MERCED_HOME/var/dice/logs directory.
- CubePop — The m cubepop command runs CubePop processes. Locate logs for this command in the MERCED_HOME/var/cube/logs directory.
- System logs — The system writes errors during startup and messages directed to standard output in the merced_std_err.log and merced_std_out.log files, found in the MERCED_HOME directory.

See Appendix C, "Commands," for a list of commands you might use and the location of their log files.

NOTE: This document describes standard locations for log files. The application writes most log files to the var directory. If necessary, you can configure the application to write the var directory in an alternate location, such as a directory outside of MERCED_HOME. However, Merced recommends that you retain the standard configuration.

To configure an alternate location for the var directory, set the **varDirectory** property in the /Configuration product component to the desired directory. For example, to set the var directory to D:/merced in an environment with Windows-based application server machines, you might create a properties file that looks like:

/Configuration.properties

```
varDirectory=file:///D:/merced/var
```

Reading log files

All Merced Performance Suite tools produce log files containing diagnostic information. The log format is configurable, but in the typical format, an entry contains four lines:

```
info|GLBL000001|2004 08 19 12:31:14.864|Thread-16|null
     |/system/Engine9
     |
     |Stopped Merced Server
```

This entry contains several key pieces of data:

- `info` — Indicates log level. Either Error, Warning, info, debug, trace, or other more verbose levels useful for development.
- `GLBL000001` — Indicates error code; not currently used.

- `2004 08 19 12:31:14.864` — Time stamp; indicates when error was written.
- `Thread-16` — Thread name.
- `null` — Request information. Not contained in this example because this message was not generated by a thread serving a request. If present, it would look like this:

  ```
  http://cobalt:10570/merced/regress/
  index.jsp|192.168.1.16|[3f13g0c8ellei]|SESSIONID=3f13
  g0c8ellei
  ```

  where:

  - `http://cobalt:10570/merced/regress/index.jsp` — The requested URL
  - `192.168.1.16` — The client's IP address.
  - `[3f13g0c8ellei]` — Cookie values relevant to Merced Performance Suite.
  - `SESSIONID=3f13g0c8ellei` — Session identifier.
- `/system/Engine9` — The name of the component that produced this message; in this case, an engine.
- `Stopped Merced Server` — The free-form message.

  The free form message can be difficult to parse with a program because it may contain text that looks like a log message.

Analyzing log files

When analyzing log files look at:

- First error — An error often triggers subsequent errors. The first error is usually the most relevant.

  NOTE: Search for "Error|" to find the beginning of the entry rather than searching for "Error," which can appear repeated in the free-form portion of messages. If debugging an engine log file, search from the bottom of the file.

- Timestamps — Use timestamps to correlate events in various logs.

  To use this technique effectively, you must synchronize the clocks of all hosts in the cluster. For example, you could run an NTP client on all hosts in a cluster, your database server, and maybe even your client workstations.

- Single thread — Search for messages logged by a single thread. Merced Performance Suite is a multi-threaded product. Trying to understand all the log messages at once can be difficult, and they may not tell a coherent story.

- A session — Search for messages logged in an individual session ID. You might need to correlate logs from multiple servers.

Troubleshooting

This section describes issues you might face when maintaining the system that supports Merced Performance Suite and tasks that you might complete when troubleshooting these issues:

- Common system issues
- Troubleshooting tasks

Common system issues

This section list common system issues that might arise and suggests tasks you might complete to resolve these issues.

System issue	Suggested task to resolve
Load balancer is unresponsive or slow-responding	"Resolving engine issues" on page 113
Someone manually kills Java process running CubePop or NextPop	"Manually recovering from failed CubePop or NextPop jobs" on page 114
Someone manually kills Java process running DICE job	"Manually recovering from failed DICE conversions" on page 115
Application server machine running CubePop or NextPop goes down	"Manually recovering from failed CubePop or NextPop jobs" on page 114
Application server machine running DICE job goes down	"Manually recovering from failed DICE conversions" on page 115
Database connection lost during CubePop or NextPop	"Manually recovering from failed CubePop or NextPop jobs" on page 114
Database connection lost during DICE job	"Manually recovering from failed DICE conversions" on page 115

Troubleshooting tasks

This section describes tasks you might complete when troubleshooting common issues that can arise when maintaining the Merced Performance Suite system:

- Resolving engine issues
- Manually recovering from failed CubePop or NextPop jobs
- Manually recovering from failed DICE conversions

Resolving engine issues

Problems with engines might appear during routine monitoring tasks or they might manifest as a slow-responding or unresponsive load balancer. To resolve engine issues, you must identify the problem engines, stabilize the system, and analyze the problem.

➢ To resolve engine issues

1. Identify engines that are not running properly.
2. Stop the load balancer or proxies if currently healthy engines are unable to handle the user load.

   Routing more requests to healthy engines might cause those engines to malfunction, also.
3. Stop and restart malfunctioning engines.

   Depending on how you run your system, you might use the m mserver command or you might use a scheduled task. See "Running engines and proxies" on page 24.

   When stopping an engine manually, collect several pieces of information:

   - Symptoms experienced
   - Concurrent activities and events
   - Relevant log files

4. If you stopped the load balancer or proxies, restart after a sufficient number of engines are online.
5. Analyze log files to determine source of problem.
   a. Search log files for the string `Error|`, beginning at the bottom of the file.
   b. After finding the error, search down for `rootCause`.

      Common errors include:

      - java.lang.OutofMemory
      - java.lang.StackOverflowError
      - Max number of connections

      Typically, these errors indicate that an engine is overloaded. If all engines in a cluster are running when this error surfaces, your cluster might have an inadequate number of engines.

Manually recovering from failed CubePop or NextPop jobs

CubePop and NextPop can handle exceptions that arise when populating the database. When such exceptions arise, the job stops naturally, usually without populating the database, and writes an error to the log file. In this case, the application project team can fix the problem and run subsequent CubePop or NextPop processes normally.

Occasionally, CubePop or NextPop stops unnaturally. When stopped unnaturally, the job is unable to update its status in the database; subsequent CubePop or NextPop processes cannot run until you manually correct the status of the failed job.

You must manually recover from failed CubePop or NextPop jobs when:

- Someone manually kills the Java process running the job
- Application server machine running job goes down
- Database connection was lost during the job

➢ To recover manually from failed CubePop or NextPop jobs

1. Open a command window on the node where the failed CubePop or NextPop job was running.
2. Run the cubepop -cleanup command to restore the database to its pre-CubePop or NextPop state and enable new CubePop or NextPop processes to run.

   Type:

   ```
   $MERCED_HOME/bin/m cubepop -cleanup
   ```

Manually recovering from failed DICE conversions

DICE conversion jobs can handle exceptions that arise when extracting data from source systems and loading the data into the database. When such exceptions arise, the job stops naturally and writes a message to the log file. The application project team can fix the problem and run a subsequent conversion job normally.

Occasionally, a conversion job stops unnaturally. When stopped unnaturally, the job is unable to release the semaphores it took to prevent other jobs from writing to the same tables. Other conversion jobs that use the same tables cannot run until you manually release semaphores that the failed job holds.

You must manually recover from failed DICE conversions when:

- Someone manually kills the Java process running the job
- Application server machine running job goes down
- Database connection was lost during the job

➢ To recover manually from a failed DICE conversion

1. Open a command window in the cluster.
2. List the state of the application's semaphores. Type:

   ```
   m semaphore list
   ```

   The command returns a list of the semaphores and indicates whether each is taken and which process has taken it.
3. Find any semaphores held by the failed conversion.

   Search the "Acquired" column for the failed conversion. The namespace column identifies the semaphore.
4. Release any semaphores that the failed conversion holds.

   For example, if the failed conversion holds the semaphore identified by the SPLIT_ID namespace, type:

   ```
   m semaphore forcibly-release NAMESPACE:SPLIT_ID
   ```

A Glossary

This glossary defines terms that Merced Performance Suite documentation uses, including common terms that have a special meaning in Merced Performance Suite. Within a definition, other terms that are defined in the glossary appear in **bold type**.

access control list

A set of **roles** allowed access to specified Merced Performance Suite objects, such as a **report**, **dashboard**, or particular page.

base metric

A numerical measurement, or statistic, for a specific set of dimension labels. A base metric has a rule, or formula, that describes how to extract and calculate that statistic. **CubePop** and **NextPop** populate **fact tables** in the **OLAP data mart** with base metrics. CubePop and NextPop calculate base metrics by aggregating rows from a table in the **OLTP data mart**.

calculated metric

A numerical measurement, or statistic, for a specific set of dimension labels. A calculated metric has a rule, or formula, that describes how to compute that statistic, both the inputs and the computation method. Calculated metrics can use **base metrics** and other calculated metrics as inputs, and they can use any

computation method. Merced Performance Suite computes calculated metrics when required by a **report** and stores calculated metrics in memory.

Cbuild

A tool to deploy Merced Performance Suite.

CMS

Content Management System. A set of properties files that controls what information appears in the user interface.

cube

A **fact table** in the **OLAP data mart**.

CubePop

An executable that creates or updates data in every existing **dimension table** and **fact table**. Fact tables that CubePop loads reference data in newly-created or updated dimension tables.

dashboard

A set of **report** summaries, which allow users to quickly view information presented by various reports and easily access a full report for more details. Typically, reports contained in a dashboard address a similar theme.

data restriction policy

A list of **roles** and a data restriction relationship for each role. The restriction relationship indicates about whom persons in that role can access data. For example, a data restriction policy might give directors access to information about everyone, managers access to information about their team members, and agents no access at all.

data mart

A highly-focused relational **database** that stores data about a particular subject or function. There are two distinct data marts—sets of tables—in the Merced Performance Suite database: an **OLTP data mart** and an **OLAP data mart**.

database

A collection of tables managed by a **DBMS**, such as Microsoft SQL Server or Oracle. The Merced Performance Suite database contains two **data marts**: an **OLTP data mart** and an **OLAP data mart**.

DBMS

Database management system. A software system that builds and maintains database tables.

DICE

Data Integration and Cleansing Engine. DICE is the process that extracts data from data sources and loads the data into the Merced Performance Suite **OLTP data mart**. DICE is an ETL (extract transform load) engine.

DICE conversion

A specification for extracting, loading, and transforming data from source systems into the **OLTP data mart**.

DICE job

A set of DICE conversions and **NextPop** processes run together, either in sequence or in parallel.

dimension

A space that an organization uses to measure values, such as the time space. Dimensions correspond to axes. Values on those axes—values in the dimension—are called labels. Labels describe a point or area in the dimension. Labels imply units, such as the units in a graph or on a report.

dimension table

Database table that stores all possible values for a particular **dimension**.

Employee Manager

A tool accessible from the Admin page that allows you to create and maintain **org chart** data for Merced Performance Suite.

engine

An instance of a Merced Performance Suite application server. An engine is the core application that renders reports and Web pages, and authenticates users.

fact table

Database table that stores values measured for combinations of relevant **dimensions**. Fact tables are also called cubes. In Merced Performance Suite, you configure fact tables with **base metrics**.

group

A particular employee or collection of employees who work at a specific level and have a specific function in an organization's hierarchy.

JDBC

Java database connectivity. The part of the Java Development Kit that defines an application programming interface to give Java programs standard **SQL** access to **databases**.

localconfig

A subdirectory of the **MERCED_HOME** directory. Properties files found in the localconfig directory override properties files built into the application configuration. Therefore, you can temporarily customize your application with files placed in the localconfig directory.

MERCED_HOME

The single directory on the application server machine that contains the files and executables that support the Merced Performance Suite application, including proxies, engines, and jobs. You might give the actual directory a different name and create a system variable called MERCED_HOME that references the actual directory.

metric

A **base metric** or **calculated metric**.

NextPop

A process that updates an individual **fact table** in the **OLAP data mart** with fresh data from the **OLTP data mart**. Fact tables that NextPop loads continue to reference current dimension tables. You run NextPop from a **DICE job**.

ODBC

Open database connectivity. A standard for accessing different database systems. An application can submit statements to ODBC using the ODBC flavor of **SQL**. ODBC translates these statements to whatever flavor the **database** understands.

org chart

The **persons** and **groups**, and the relationships between the persons and groups, that Merced Performance Suite stores data for and reports on.

OLAP data mart

Online analytical processing **data mart**—a set of relational database tables optimized for analysis. In Merced Performance Suite, the OLAP data mart organizes data into **fact tables** and **dimension tables** that allow end user modules to generate **reports** and **dashboards** quickly and reliably.

OLTP data mart

Online transaction processing **data mart**. A set of relational database tables optimized for operational or transactional use. In Merced Performance Suite, the OLTP data mart contains captured source system data, offering fast and reliable access to the data.

person

An employee in the **org chart**. Each person must have a unique identifier, **role**, and parent **group**, in addition to characteristics like name.

proxy

An instance of a Web server. When end-users request the Merced Performance Suite application in their browser, a proxy determines what **engine** to run their browser sessions on, and sends browser requests to that engine.

report

A presentation of related **metrics** (data from the **OLAP data mart**), which allows users to understand and analyze select issues.

role

An employee type, usually related to the employee's function and level in the organization.

SQL

Structured query language. A language supported by many **DBMSs** that allows you to pose complex questions of a **database**.

transactional report

A set of data retrieved from a table in the **OLTP data mart**. Transactional reports retrieve data that match preconfigured criteria.

transactional search

A set of data retrieved from a table in the **OLTP data mart**. Transactional searches retrieve data that match user-entered criteria.

B System Requirements

This appendix lists hardware and software requirements for application server, database server, and client machines running Merced Performance Suite.

Sections include:

- Application server machine
- Database server machine
- Client machine

Application server machine

Software

Certified operating systems
Windows Server 2003
Windows Server 2003 Enterprise Edition, SP1
Windows 2000 Server, SP3+
Windows 2000 Advanced Server, SP3+
Red Hat Enterprise Linux 3 AS and ES[a]

a. X server must be installed to support NetCharts

Required software
Python version 2.2.1
Java 2 Platform, Standard Edition, v 1.4.2_05 SDK [a]

a. Previous versions of Java must be uninstalled

NOTE: Merced Performance Suite is bundled with Apache Web server software (version 2.0.53). Merced Performance Suite does not integrate with other Web servers. You can integrate your own Apache server or derivative of an Apache server; contact CSG for details.

Hardware

Server-class machines with a *minimum* of:

- Two 3.2 GHz Xeon CPUs
- 4 GB RAM
- 18GB Hard disk, SCSI

When determining the number of application server machines, remember these cluster requirements:

- Minimum 1.5 GB RAM per engine
- Minimum of 1 engine per 1250 licensed users (assuming 300 concurrent active users)
- Minimum of 2 application server machines for failover

Also consider your organization's preferences:

- Number of engines per application server machine
- Running proxies and engines on the same machines
- Running DICE and CubePop jobs on dedicated machines

Database server machine

Software

Certified database applications
IBM DB2 Universal Database Enterprise Server Edition V8.2.2 (FixPak 9)
Oracle 9i (9.2.0.7 with patch 3967276)
SQL Server 2000, SP3

Hardware

For Oracle, Merced Performance Suite requires a database server machine with a *minimum* of

- Two 3.2 GHz Xeon CPUs
- 6 GB RAM
- 600 GB storage device; 0+1 RAID
- Maximum possible disk RPM speed

Client machine

Client machines require:

- Microsoft Internet Explorer 6.0 (or higher versions). JavaScript must be enabled. No plug-ins are required.

  NOTE: Merced Performance Suite can support other browsers, but no others are currently certified.

- Microsoft Excel to open files created with the "Export to formatted Excel (xls)" function.
- Adobe Reader 5.0 or Adobe Acrobat 5.0 (or higher versions) to view and print PDF files generated with the "Create PDF" function.

C Commands

This appendix lists commands you might use when administering Merced Performance Suite. Specific sections include:

- Using quotes in commands
- dumpcc
- jsql
- m
- m cubepop
- m dice
- m encrypt
- m export-cbeans
- m file-splitter
- m import-cbeans
- m load-metrics
- m mserver
- m optimize-db
- m optimize-repository
- m repository-upgrader

- m schemaload-main
- m schemaload-reports
- m semaphore

Using quotes in commands

Commands require instructions in precise syntax. Some commands, for example, require components specified as BEL expressions:

```
(creator:/com/merced/component_name)
```

Other commands require the fully-specified component name:

```
/com/merced/component_name
```

You must ensure that the operating system sends the proper syntax to the command executable. Operating system shells parse command arguments before sending them to executables. These shells apply special interpretation to some characters, such as parentheses or spaces. To avoid this special interpretation during parsing, you must escape the character—you must mark the character or sequence of characters that you want sent without any interpretation. In many shells, you can use single quotes or double quotes to escape a sequence of characters.

Escaping requirements vary by operating system and shell. See the documentation for your shell for precise information. Many Unix shells prefer single quotes (') for escaping, and many Windows shells prefer double quotes (") for escaping. Unix shells often accept double quotes, however. And Windows shells do not require, but accept, escaping for some characters, like parentheses.

This documentation uses double quotes to escape parentheses characters in commands. Your shell may require no quotes, or may require single quotes.

BEL expressions, for example, usually require escaping:

```
"(creator:/com/merced/component_name)"
```

dumpcc

The dumpcc command lists the properties files that Creator used to construct a component and the values for the component's properties.

SYNTAX

To run the dumpcc command, open a command window on a cluster node and type:

`$MERCED_HOME/bin/dumpcc` *component_name*

where *component_name* is the fully-specified name of a component.

This command is useful for debugging and inspection. For example, to see which database your application connects to, examine the /Configuration component. Type:

```
$MERCED_HOME/bin/dumpcc /Configuration
```

NOTE: Do not use this command to debug and inspect components with scopes other than global. The dumpcc command examines components defined throughout the entire application (that is, components with the **$scope** property set to "global"). Other components are only defined within a specific context. For example, the UserDataStatusConfiguration component only exists within the context of a browser session. Attempting to examine non-global components with the dumpcc command might produce an error.

jsql

The jsql command starts a JDBC SQL client. Once connected, you can run arbitrary SQL commands, as long as the JDBC driver understands those commands.

SYNTAX

You can call the jsql command in two ways:

- By using an available JDBC driver. Type:

  `jsql --driver` *driver-class* `--username` *username* `--password` *password* `--url` *jdbc-url*

  You specify the JDBC driver's class name, login credentials, and a JDBC URL.

  The driver class for IBM DB2 UDB is:

  `com.ibm.db2.jcc.DB2Driver`

  The driver class for Microsoft SQL Server is:

  `com.microsoft.jdbc.sqlserver.SQLServerDriver`

  The driver class for Oracle is:

  `oracle.jdbc.driver.OracleDriver`

- By using the JDBC configuration component. Type:

  `jsql --config` *jdbc-configuration-component*

  The JDBC configuration component contains the necessary connection information. This method must start Creator, and is therefore a little slower. However, you can use this method to test your JDBC configuration components.

  NOTE: The jsql command cannot read encrypted passwords. If your configuration uses encrypted database passwords, you must also specify the password with the jsql command:

  `jsql --config` *jdbc-configuration-component* `--password` *clear-text-password*

Therefore, when using encrypted passwords you cannot use the jsql command to test the JDBC configuration component.

The jsql client passes SQL commands to the database through the specified JDBC driver. The database engine executes the SQL command. The jsql client does little parsing of the SQL commands. There are several conventions to follow:

- Use SQL commands specific to the DBMS. For example, in SQL Server you might type `sp_help`, but in Oracle you might type `describe`.
- Terminate every command with a semicolon (;). Use this jsql termination character even if the native DBMS command tool uses a different terminator.
- Exit jsql by typing:

  ```
  quit;
  ```

m

The m command runs executables in the MERCED_HOME directory.

SYNTAX

To use the m command, open a command window on a node and type:

`$MERCED_HOME/bin/m` *command* [*command-args ...*]

where:

- command — A valid command or component (specified as a BEL expression) that is configured as a command. See the remaining sections in this appendix for an explanation of many useful commands.
- command-args — Zero or more arguments that the operating system passes to the command.

Many commands have a `--help` option. Running this option provides information about how to use the command.

m cubepop

The m cubepop command starts CubePop, the process that loads (or populates) the entire contents of the OLAP data mart from the data in the OLTP data mart. See "Details" on page 142 for more information.

LOG FILES

/var/cube/logs

SYNTAX

To run the m cubepop command, open a command window on the cluster node designated to run jobs, and type:

```
$MERCED_HOME/bin/m cubepop [option_name]
```

where *option_name* is a run option. Possible options include:

- no option
- -offline option
- -force-optimize option
- -retry option
- -cleanup option
- -only-pack-cubes option

no option

Do not specify an option to run CubePop online; application engines can continue to run and produce reports based on existing data until CubePop completes data population. When run online, CubePop populates new fact tables and dimension tables and requests the DBMS (except for SQL Server) to compute database statistics based on the configured strategy for your application. See *Mini-Guide to Database Statistics Management* for information about database statistics strategies.

To run CubePop online, open a command window and type:

```
$MERCED_HOME/bin/m cubepop
```

-offline option

Use the -offline option to build an original set of metadata, dimension, and fact tables. When run with the -offline option, CubePop eradicates all of the tables in the OLAP data mart, including metadata, and rebuilds an original set of tables from scratch. CubePop also requests the DBMS (except for SQL Server) to compute database statistics for fact tables and dimension tables and for tables in the OLTP data mart containing stale statistics.

Run CubePop using the -offline option when:

- You load the OLAP data mart for the first time (e.g., a new deployment).
- You make a change, other than adding rows of data, to any dimension.

  After you add or remove grains or custom grains, the application cannot render any reports that reference those dimensions until you run CubePop. Merced recommends running CubePop with the -offline option in this case.

NOTE: Stop all engines in the cluster before executing CubePop with the -offline option. Failure to stop engines can produce strange application behavior, though no lasting damage will result.

To use the m cubepop -offline command, open a command window and type:

```
$MERCED_HOME/bin/m cubepop -offline
```

-force-optimize option

Use the -force-optimize option when you want the DBMS (other than SQL Server) to compute database statistics for tables in the OLAP data mart. When run with the -force-optimize option,

CubePop populates fact tables and dimension tables online, and requests the DBMS to compute new statistics for these tables, regardless of when statistics were last updated. CubePop also requests the DBMS to compute new statistics for OLTP data mart tables containing stale statistics.

To use the m cubepop -force-optimize command, open a command window and type:

```
$MERCED_HOME/bin/m cubepop -force-optimize
```

-retry option

Use the -retry option after correcting a configuration error that caused the previous CubePop job to fail. When run with the -retry option, CubePop populates tables that it could not populate in its previous attempt and deploys all the fresh tables.

The -retry option is useful when a previous CubePop exited abnormally due to a configuration error that you have since corrected. The previous CubePop may have left many staged tables—tables loaded with fresh data but not deployed—because CubePop does not deploy fresh tables when at least one table cannot be populated.

For example, you might use this option when an incorrectly configured base metric prevented CubePop from populating one or more fact tables. After noticing and correcting the metric, run `cubepop -retry` and CubePop will populate only the fact tables that it could not populate before and will deploy the entire fresh set of dimension and fact tables.

If recovering from a CubePop run with the -force-optimize option, you must run the -retry option with the -force optimize option.

To use the m cubepop -retry command, open a command window and type:

```
$MERCED_HOME/bin/m cubepop -retry
```

To run with the -force-optimize option, type:

```
$MERCED_HOME/bin/m cubepop -retry -force-optimize
```

-cleanup option

Use the -cleanup option if CubePop does not terminate normally (see "Manually recovering from failed CubePop or NextPop jobs" on page 114). When run with the -cleanup option, CubePop restores the OLAP data mart to its state before the failed CubePop attempt.

NOTE: Only use this option if you are certain that a CubePop process is not currently running.

To use the m cubepop -cleanup command, open a command window and type:

```
$MERCED_HOME/bin/m cubepop -cleanup
```

When called with the -cleanup option, CubePop:

- Releases the CubePop semaphores

  See "Details" on page 142 for a description of semaphores. This option releases any semaphores still held.

- Retires any dimension table that is not deployed
- Retires any fact table that refers to deactivated dimension tables

Essentially, running CubePop with the -cleanup option removes any staged tables—tables with new data loaded but not deployed. This option returns the set of tables and sibling tables to a "clean" state.

-only-pack-cubes option

Use the -only-pack-cubes option to move any unassigned base metrics to cbean-based cubes without loading data. When run with the -only-pack-cubes option, CubePop assigns unassigned base metrics to cbean-based cubes and writes the newly assigned cube for each base metric in the log file; CubePop does not populate the database.

You can use Cube Manager to move the metrics to other compatible cubes or to create Kimball aggregate cubes. See *Mini-Guide to Automatic Cube Management* for details. After assigning new base metrics to cubes with this option, you must run CubePop with a different option to load data into the tables.

To use the m cubepop -only-pack-cubes command, open a command window and type:

```
$MERCED_HOME/bin/m cubepop -only-pack-cubes
```

DETAILS

CubePop loads (or reloads) all the tables in the OLAP data mart—the dimension tables and the fact tables. CubePop loads data into a set of staged sibling tables. After populating all the tables, CubePop deploys the staged sibling tables, making them the active tables that reports use.

Normally, CubePop can execute while application engines continue to run. Executing CubePop places significant load on the database and can cause noticeable performance degradation of the application. However, reports will continue to be available unless you use the -offline option.

Configuration of the CubeUniverse component (/com/merced/cube/CubeUniverse) affects CubePop. In particular, this component controls the number of tables a job can populate simultaneously and whether CubePop can run while NextPop runs.

CubePop uses semaphores (rows in specific tables that indicate a process is running) to ensure that only one CubePop process runs at a given time. CubePop uses three semaphores:

- M_UNIVERSEMETADATA — CubePop enters a row in this table to indicate the process is running
- M_DIMENSIONMETADATA — CubePop enters a row while populating dimension tables

- M_CUBEMETADATA — CubePop enters a row while populating fact tables

When CubePop completes normally, CubePop releases the semaphores, allowing the next CubePop to run. When CubePop exits abnormally, one or more semaphores might remain in place, preventing subsequent CubePop processes from executing. In this case, the log messages for those subsequent CubePop processes contain error messages suggesting this condition.

m dice

The m dice command can start a DICE conversion, a NextPop population, or a DICE job that combines multiple conversions or NextPop populations. A DICE conversion extracts data from a source system and loads the data in the OLTP data mart. A NextPop process updates a fact table in the OLAP data mart with fresh data from the OLTP data mart.

LOG FILES

/var/dice/logs

SYNTAX

To run the m dice command, open a command window on the cluster node designated to run jobs, and type:

`$MERCED_HOME/bin/m dice` *arguments*

where *arguments* specify run-time parameters that vary by your task. You might use the m dice command for several tasks:

- Getting help
- Running a job or conversion
- Running conversion for specific dates

Getting help

To get help with the m dice command, type:

```
m dice --help
```

Add the optional keyword "jobs" to see a list of all jobs and conversions currently available. Type:

```
m dice --help jobs
```

Running a job or conversion

To run a job or conversion, type:

`m dice` *job_name*

where *job_name* is one of three types of components, specified as a BEL expression:

- DICE conversion — A component that describes how to extract, transform, and load data from a particular data source.
- NextPop population — A component that specifies the cubes to load with fresh data from the OLTP data mart during NextPop along with other required parameters.
- DICE job — A component that defines one or more DICE conversions, NextPop populations, or other DICE jobs to run together. A DICE job can specify an execution order and dependency among components.

For example, to initiate a DICE job called AvayaJob, you might type:

```
m dice "(creator:/com/merced/dice/jobs/AvayaJob)"
```

Running conversion for specific dates

To run a single conversion but override the dates the conversion specifies, use the time-range option. Type:

`m dice --time-range` *from-time to-time job_name*

where *from-time* is the start date and *to-time* is the end date for changed data capture purposes. Typically, the from-time is inclusive and the to-time is exclusive. The changed data capture method determines how DICE uses these dates for capturing data.

For example, to run AvayaJob for two specific dates, you might type:

```
m dice --time-range 1/29/2004 1/30/2004 "(creator:/com/
merced/dice/jobs/AvayaJob)"
```

Enter `BOT` in the from-time to specify the beginning of time, and enter `EOT` in the to-time to specify the end of time. Specifying these dates reloads all data in the system.

If the conversion is configured to understand grains finer than an hour and you want to run AvayaJob for a specific set of hours, you might type:

```
m dice --time-range "1/29/2004 01:00:00" "1/30/2004
03:00:00" "(creator:/com/merced/dice/jobs/AvayaJob)"
```

In this case, you must escape the dates with quotes because each date includes a space.

m encrypt

The m encrypt command generates encrypted text for a password. You must enter encrypted passwords in the properties files for JDBC configuration components when you configure your application to require encrypted database passwords. See "Database password encryption" on page 52 for more information.

SYNTAX

To run this command, open a command window and type:

```
$MERCED_HOME/bin/m encrypt password
```

where *password* is the clear text (decrypted) version of the password.

In the command window, the command returns the encrypted version of the password.

NOTE: You cannot encrypt passwords exceeding 512 characters. For such passwords, the command generates an error.

m export-cbeans

The m export-cbeans command writes the cbeans defined in the cbean repository to an external file. This command writes a file that contains the data stored in the cbean tables and the structure of the cbean tables.

LOG FILES

/var/export-cbeans/logs

SYNTAX

To run this command, open a command window and type:

`$MERCED_HOME/bin/m export-cbeans [`*arguments*`]`

When you run the command without an argument, the command writes the contents of the cbean repository to stdout. There are two options:

- -f *file* — Use the -f flag to specify a file to export cbeans to. Specify the *file* as a URI (uniform resource identifier), using either an absolute reference, such as file:/usr/local/test.cbeans, or a reference relative to the MERCED_HOME directory, such as file:cbeans/test.cbeans.
- -a — Use the -a flag to create an archive file of the existing state of the cbean repository. With this option, the command writes the information to a time-stamped file in the MERCED_HOME/cbeanadmin/archives directory.

EXAMPLES

```
$MERCED_HOME/bin/m export-cbeans
$MERCED_HOME/bin/m export-cbeans -f file:my-export.cbeans
$MERCED_HOME/bin/m export-cbeans -f file:/usr/local/my-export.cbeans
$MERCED_HOME/bin/m export-cbeans -a
```

m file-splitter

The m file-splitter command divides very large files into a set of smaller files, and rejoins the smaller files back into the single, large file.

SYNTAX

Splitting files

To split a file, type:

```
$MERCED_HOME/bin/m file-splitter --split filename
```

where *filename* is the name of the file, either fully qualified or relative to MERCED_HOME.

Using the --split option divides the file into separate files of half a gigabyte each, adding the .MER file extension.

For example, if you split the mytest.log file by typing:

```
$MERCED_HOME/bin/m file-splitter --split mytest.log
```

the command creates three files:

```
mytest.log.MER1
mytest.log.MER2
mytest.log.MER3
```

Rejoining files

To rejoin a file split with this command, place the split files into the directory where you want to create the joined file, and from that directory, type:

```
$MERCED_HOME/bin/m file-splitter --rejoin filename
```

where *filename* is the name of the original file before the split.

The m file-splitter command is useful when you need to send a multi-gigabyte file across the Internet, and your current tools cannot handle such a large file.

m import-cbeans

The m import-cbeans command loads cbean specifications into the cbean repository, the tables in the database that pertain to cbeans. This command loads cbeans specified in a particular file or in a set of files defined by a registry. Before loading cbean specifications, the command can archive the repository's existing cbean specifications or purge existing cbeans from the repository.

LOG FILES

/var/import-cbeans/logs

SYNTAX

To run this command, open a command window and type:

`$MERCED_HOME/bin/m import-cbeans` *arguments*

There are two optional arguments and one required argument.

Optional arguments

- -a — Use the -a flag to archive the existing state of the cbean repository before importing new cbean specifications. Having an archive file allows you to restore to the present (pre-import) state later, if necessary.

  With this option, the command generates an archive in a time-stamped file in the MERCED_HOME/cbeanadmin/ archive directory.

  In the default product configuration, the -a flag is implicit in every import (that is, without specifying this flag the command archives the current repository).

  You can disable implicit archiving by setting the **autoArchive** property to "false" in the /com/merced/cbeans/repository/ ImportCBeansMainService component.

- -p — Use the -p flag to purge the current cbean repository before importing new cbean specifications.

  With this option, the command deletes all data in the cbean repository table.

Required arguments

You must specify exactly one of three required arguments:

- -f *file_name* — Use the -f flag to import cbeans from a file. Requires at least one *file_name*. Separate additional names with spaces. Specify the *file_name* as a URI (uniform resource identifier), using either an absolute reference, such as file:/usr/local/test.cbeans, or a reference relative to the MERCED_HOME directory, such as file:cbeans/test.cbeans.
- -r [*registry_name*] — Use the -r flag to import cbeans from all files defined in a registry. The registry must be of class com.merced.cbeans.CBeansRegistry. Specify the registry_name using a BEL expression. You can specify any number of registries, separated by spaces. If you specify no registries, this command imports cbeans from the files specified by the default registry, /CBeans.
- -init — Use the -init flag to initialize the cbean repository. With this option, the command imports cbeans from the files defined in the /CBeans registry. Running this option is the same as running the -r option without specifying a registry.

EXAMPLES

```
$MERCED_HOME/bin/m import-cbeans -init

$MERCED_HOME/bin/m import-cbeans -p -init

$MERCED_HOME/bin/m import-cbeans -a -f file:my-cbeans.cbeans

$MERCED_HOME/bin/m import-cbeans -f file:my-cbeans.cbeans file:/usr/
local/my-cbeans2.cbeans

$MERCED_HOME/bin/m import-cbeans -r
```

```
$MERCED_HOME/bin/m import-cbeans -r (creator:/CBeans) (creator:/com/
merced/OtherRegistry)

$MERCED_HOME/bin/m import-cbeans --help
```

m load-metrics

The m load-metrics command loads the application database with the definitions of metrics found in properties files. This command only loads new metric definitions. This command does not load definitions of metrics that already exist in the database, even if the properties file definition is different. After you load a metric definition into the database, you can only change that definition through the user interface. This command loads the definitions of base metrics, calculated metrics, goal metrics, and level-restricted calculated metrics (LRCMs).

You must run this command to load metric definitions before running the m schemaload-reports command to load report definitions that reference those metrics.

LOG FILES

/var/LoadMetricsMainService/logs

SYNTAX

To run this command, open a command window and type:

```
$MERCED_HOME/bin/m load-metrics
```

OTHER

You can delete all the metrics defined in the database and use the m load-metrics command to reload metrics based on properties file definitions. To delete metrics defined in the database, issue the following SQL commands in order:

```
drop table m_metricgroup_childmetrics_rel;
drop table m_metricgroup;
drop table m_metriclevelrestricted;
drop table m_metriccalculated;
drop table m_metricbase;
drop table m_metriccommon_childmet_gw_rel;
drop table m_metricgoal;
```

```
drop table m_metriccommon;
drop table m_metricdimension;
drop table m_unitformat;
drop table m_datasource;
```

NOTE: These commands delete all defined metrics, including those created with the user interface. Metrics created with the user interface do not exist in properties files; once you delete those metrics, you must recreate them manually.

m mserver

The m mserver command controls Merced Performance Suite servers, either engines or proxies.

LOG FILES

/var/*server_name*/logs

SYNTAX

To use the m mserver command, open a command window on a node and type:

`$MERCED_HOME/bin/m mserver` *operation* [*server_name ...*]

where *operation* is a task and *server_name* is the value of the **name** property for a particular engine or proxy (a particular "server"). The command initiates the operation for the specified servers.

Valid operations for particular servers are:

- start[1] — Starts the server in the background. Can enter multiple server names.
- stop[1] — Stops the server. Can enter multiple server names. After stopping an engine, wait ten minutes before starting it again.
- restart[1] — Stops and then restarts the server.
- run — Starts the server in the foreground. Prints logs from stdout to the screen.
- halt[1] — Stops the server immediately. Only use if the stop operation fails.

1. Do not use this command for proxies that run on Windows-based application server machines.

If you specify a server not on the node, that server is not affected. If you do not specify any server, the command initiates the operation for all engines and proxies on the node.

Valid operations for a node are:

- list — Prints a list of servers on this node.
- register-system-service — Registers a system service for all the engines that run on the node. By default, the system service name is com_merced_server_MServerMainService.
- unregister-system-service — Unregisters the system service for the engines that run on the node.

  If you unregister a system service in Windows, reboot the node before you re-register that system service.

NOTE: Do not specify a server if you use these operations.

m optimize-db

The m optimize-db command computes statistics for the entire database schema. This command is only valid for Oracle databases.

SYNTAX

To run this command, open a command window and type:

```
$MERCED_HOME/bin/m optimize-db
```

NOTE: Merced recommends that you discontinue use of the m optimize-db command to compute database statistics. Proper use of the m optimize-repository --stale only and the m cubepop -force-optimize commands is a more effective strategy for keeping database statistics up-to-date. The m optimize-db command is inconsistent with the algorithm CubePop uses. See *Mini-Guide to Database Statistics Management* for details.

m optimize-repository

The m optimize-repository command estimates statistics for all the tables, indexes, and columns in a named metabean repository or in all production repositories. This command is only valid for metabean repositories that represent connections to DB2 UDB or Oracle databases.

SYNTAX

To run this command, open a command window and type:

```
$MERCED_HOME/bin/m optimize-repository [component_name]
```

where *component_name* is a repository component specified as a BEL expression. For example, to estimate statistics for tables in the primary metabean repository, type:

```
m optimize-repository "(creator:/com/merced/datamart/
repository/DataMartRepository)"
```

If you do not specify a component, the command requests the DBMS to estimate statistics for all tables in the production repositories—all tables in the OLTP data mart.

You can also run the m optimize-repository command with the --stale-only option.

--stale-only option

Use this option to update previously calculated statistics that are out-of-date, or "stale." This command requests the DBMS to update stale statistics for tables in a specified OLTP data mart repository. Oracle determines whether the statistics are stale for any tables in that repository and estimates new statistics for those tables. DB2 UDB estimates new statistics for tables without any statistics, but computes no statistics for tables with existing statistics, even stale statistics.

To use this option, type:

```
m optimize-repository --stale-only [component_name]
```

where *component_name* is a repository component specified as a BEL expression. To apply this command to the tables in all production repositories, omit a component name; type:

```
m optimize-repository --stale-only
```

NOTE: This command does not always detect stale statistics immediately in Oracle. The database and its configuration affect when this command detects stale statistics. Run this command periodically to keep statistics up-to-date over long periods of time. Do not run this command to detect tables newly in need of updated statistics, such as in the midst of a DICE job.

m repository-upgrader

The m repository upgrader command modifies a metabean repository to ensure the repository complies with product configuration. This command modifies the schema without dropping data. As necessary, the command initializes the database, creates tables, adds columns to existing tables, adds indexes and constraints, and loads stored procedures, based on the application's configuration files. The command also loads seed data.

NOTE: This command updates or initializes a single metabean repository. The m schemaload-main command updates or initializes all registered metabean repositories other than the reports repository. The m schemaload-reports command updates or initializes the reports repository.

SYNTAX

To run this command, open a command window on a cluster node and type:

```
$MERCED_HOME/bin/m repository-upgrader component_name
```

where *component_name* is the repository component that you want to initialize.

For example, to initialize the primary repository, type:

```
m repository-upgrader /com/merced/datamart/repository/
DataMartRepository
```

m schemaload-main

The m schemaload-main command modifies the database to ensure that the OLTP data mart complies with the product configuration not related to reports. This command modifies the schema without dropping data. As necessary, the command initializes the database, creates tables, adds columns to existing tables, adds indexes and constraints, and loads stored procedures. The command also loads seed data.

This command runs the m repository-upgrader command for all registered metabean repositories other than the reports repository. Run the m schemaload-reports command for the reports repository.

You must run this command to initialize the data mart during a new deployment and you must run this command to update the database whenever you make a configuration change that alters a metabean type, such as when you create a form. Run this command before loading data for a new configuration into the tables.

NOTE: Changing the data type of an existing column through product configuration causes this command to fail. If required, you must implement such changes in the database manually.

LOG FILES

/var/Bulk*/logs

SYNTAX

To run the m schemaload-main command, open a command window on a cluster node, and type:

```
$MERCED_HOME/bin/m schemaload-main
```

m schemaload-reports

The m schemaload-reports command modifies the database to ensure that the OLTP data mart complies with the product configuration for reports. This command modifies the schema without dropping data. As necessary, the command initializes the database, creates tables, adds columns to existing tables, adds indexes and constraints, and loads stored procedures. The command also loads seed data.

This command runs the m repository-upgrader command for the reports repository. Run the m schemaload-main command for the other repositories.

You must run this command to initialize the data mart during a new deployment and you must run this command to update the database whenever you make a configuration change that alters a report metabean type. Before running the m schemaload-reports command, run the m load-metrics command to populate the database with new metric definitions. Run the m schemaload-reports command before loading data for a new configuration into the tables.

NOTE: Changing the data type of an existing column through product configuration causes this command to fail. If required, you must implement such changes in the database manually.

LOG FILES

/var/Bulk*/logs

SYNTAX

To run the m schemaload-reports command, open a command window on a cluster node, and type:

```
$MERCED_HOME/bin/m schemaload-reports
```

m semaphore

The m semaphore command either lists the current state of registered DICE semaphores or forces the release of a currently held semaphore.

LOG FILES

/var/semaphore/logs

SYNTAX

To see a list of DICE semaphores, open a command window in the cluster and type:

```
$MERCED_HOME/bin/m semaphore list
```

The command returns a list of the semaphores and indicates whether each is taken and which process has taken it.

To release a currently-taken semaphore, open a command window and type:

```
$MERCED_HOME/bin/m semaphore forcibly-release name
```

where *name* is the name of the semaphore.

NOTE: Be certain that the process that has taken the semaphore is no longer running.

DETAILS

To ensure that two jobs do not write conflicting information regarding an identity, DICE uses semaphores. A semaphore is a row in the M_SEMAPHORE table. Each namespace, or system containing a unique identifier, has a semaphore. A semaphore has a unique name based on a namespace, a flag that indicates whether the semaphore is in use or taken, and a string that indicates which DICE job is using the semaphore.

DICE jobs take appropriate semaphores, based on the identities the jobs use. When finished, jobs release semaphores. If jobs terminate abnormally, such as with a critical error, they do not release semaphores. In this case, subsequent jobs that require the same semaphore cannot obtain the semaphore and cannot start until you forcibly release the semaphore using the m semaphore command.

D Data Dictionary

This appendix lists core tables common to all Merced Performance Suite data marts. Each listing shows the columns in the table with a brief description of those columns. Because applications are highly customized, your tables might contain additional columns added for your application; this dictionary does not include those columns. Similarly, your data mart will also contain custom tables that support your unique application; this dictionary does not include those tables.

This appendix contains the following sections:

- Categories
- Data types
- List of tables

Categories

This section lists tables by different categories:

- Alerts, messages, and tasks
- Forms
- Head count job
- Identities
- Job executions
- Login
- Org chart
- Scheduled jobs
- Time grains
- Transactional reports
- Other tables

Alerts, messages, and tasks

Tables related to alerts, messages, and tasks include:

- M_ALERTBULLETIN
- M_ALERTBULLETINCOPY
- M_BULLETINPRIORITY
- M_MESSAGEBULLETIN
- M_MESSAGEBULLETINCOPY
- M_TASKBULLETIN
- M_TASKBULLETINCOPY
- M_TASKSTATUS

Forms

Tables related to forms include:

- M_CUSTOMERFORM
- M_CUSTOMERFORMSTATUSTYPE
- M_CUSTOMERFORMTEMPLATE

Head count job

Tables related to the head count job include:

- M_DATECOUNT
- M_HEADCOUNT

Identities

Tables related to identities include:

- M_IDENTITY
- M_IDENTITYASSIGNMENT
- M_NAMESPACE

Job executions

The data mart stores information about many types of executables. M_EXECUTION stores root information, common to all executables; other tables, which are linked to M_EXECUTION, store information appropriate to a particular type of execution. Tables related to job execution include:

- M_CUBEEXECUTION
- M_CUBEPOPEXECUTION
- M_CUBEPOPULATION
- M_DICECONVERSIONEXECUTION

- M_DIMENSIONPOPULATION
- M_EXECUTION
- M_EXECUTIONMESSAGE
- M_EXECUTIONMESSAGETYPE
- M_EXECUTIONSTATUSSUMMARY
- M_MAINSERVICEEXECUTION
- M_NEXTPOPEXECUTION
- M_POPULATION

Login

Tables related to login include:

- M_LOGINRECORD
- M_LOGINTICKET

Org chart

Tables related to the org chart include:

- M_ATTRITIONREASONTYPE
- M_CAREEREVENT
- M_CAREEREVENTREASON
- M_CAREEREVENTTYPE
- M_CONTACTINFO
- M_COUNTRY
- M_EMPLOYMENTSTATUSTYPE
- M_GROUPTYPE
- M_INSTITUTION
- M_JOBTITLE

- M_LOCATION
- M_MGROUP
- M_MGROUPATTR
- M_MGROUP_PARENTGROUP_HIST
- M_MUSER
- M_NONPLAYERCHARACTER
- M_PERSON
- M_PERSON_CAREEREVENTS_REL
- M_PERSON_EMPLOYMENTSTATUS_HIST
- M_PERSON_PARENTGROUP_HIST
- M_PERSON_TITLE_HIST
- M_SALARYGRADE
- M_TITLETYPE

Scheduled jobs

NOTE: Do not alter values in these tables. These tables must remain in sync with running processes.

Tables related to scheduled jobs include:

- M_DURABLEPARAMETERSMAP
- M_DURABLEPARAMETERSMAPENTRY
- M_DURABLEPARAMETERSMAP_ENTRIES_REL[1]
- M_DURABLEPARAMETERSMETABEANREFERENCE[1]
- M_DURABLESCHEDULABLEWRAPPER

1. Subject to name mangling.

- M_SCHEDULEDJOB[1]
- M_SCHEDULEDJOB_TASKS_REL[1]
- M_SCHEDULEDTASK[1]
- M_SCHEDULEDTASKLOG
- M_SCHEDULEDTASKLOGENTRY
- M_SCHEDULEDTASKLOG_ENTRIES_REL

Time grains

Tables related to time grains include:

- M_CUSTOMTIMEGRAININFO
- M_CUSTOMTIMEGRAINSEED
- M_CUSTOMTIMEGRAINTYPE

Transactional reports

Tables related to transactional reports include:

- M_TRANSACTIONALREPORT
- M_TRANSACTIONALREPORTDIMENSIONMAPENTRY[2]
- M_TRANSACTIONALREPORT_ATTRIBUTES_REL[2]
- M_TRANSACTIONALREPORT_DIMENSIONMAPENTRIES_REL[2]
- M_TRANSACTIONALREPORT_PARENTDIRECTORIES_REL [2]
- M_TRANSACTIONALREPORT_ROLES_REL[2]

1. Deprecated from landingpad (DICE).
2. Subject to name mangling.

Other tables

Other core tables include:

- M_METABEANREFERENCE
- M_PAGEFETCHER
- M_TIMEZONE
- M_UNITFORMAT

Data types

The list of tables shows the data type for each column. The list shows the Merced Performance Suite data type. Actual database data types may differ, depending on the database application.

Possible Merced Performance Suite data types include:

- BigString — A large text string.
- Boolean — A true or false value.
- Component — Text containing a component name.
- Date — A date field.
- Integer — An integer number.
- Long — A long number.
- Pointer — A pointer to an ID in another table.
- Service — Text containing a component name.
- String — A text string.
- Timestamp — A date including the hour and minute.

List of tables

This section contains an alphabetical listing of tables.

M_ALERTBULLETIN

Stores all the alerts in the system.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
sender	FK	Pointer	Any principal: currently a person, group, or non-player character (the system). ID in M_METABEANREFERENCE.
recipients		String (4000)	Text indicating recipients.
priority	FK	Pointer	Identifier indicating priority of this alert bulletin. ID in M_BULLETINPRIORITY.
subject		String (256)	Text of the alert's subject line.
body		String (4000)	Text in the alert's message.
link		String (4000)	A URL.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_ALERTBULLETINCOPY

Stores a copy of each alert as received by the individual recipients. Contains one record for each recipient.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
bulletin	FK	Pointer	Alert bulletin that is sent as a copy. ID in M_ALERTBULLETIN.
recipient	FK	Pointer	Employee who receives copy of alert bulletin. ID in M_PERSON.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_ATTRITIONREASONTYPE

Defines possible reasons for employee attrition.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Text that describes a reason for employee attrition.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_BULLETINPRIORITY

Defines possible priorities for bulletins.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String (40)	Text that describes a bulletin priority.
displayName		String (40)	Text that appears for this priority in the user interface.
ordinalValue		String (40)	Sort order for this priority relative to others.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CAREEREVENT

Stores events applicable to employees careers.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
salary		Long	Employee's salary.
notificationDate		Date	Date employee is notified of the event.
acknowledgeDate		Date	Date employee acknowledged event.
effectiveDate		Date	Date this event is effective.
person	FK	Pointer	Employee for whom this event is applicable. ID in M_PERSON.
eventType	FK	Pointer	Type of event stored by this row. ID in M_CAREEREVENTTYPE.
reason	FK	Pointer	Reason for this event. ID in M_CAREEREVENTREASON.
title	FK	Pointer	Employee's title. ID in M_JOBTITLE.
salaryGrade	FK	Pointer	Employee's salary grade. ID in M_SALARYGRADE.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Number	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Number	Integer indicating row version. Incremented each time system modifies row.

M_CAREEREVENTREASON

Defines valid reasons for career events.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Text describing career event.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CAREEREVENTTYPE

Defines valid types of career events.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Text describing career event type.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CONTACTINFO

Stores employee contact information.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
emailAddress		String	Employee's e-mail address.
homePhone		String	Employee's home phone number.
emgContactPerson		String	Employee's emergency contact person.
emgContactRelationship		String	Employee's relationship to emergency contact person.
emgContactPhone		String	Emergency contact's phone number.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_COUNTRY

Defines set of countries valid in an employee's location.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
code		String	Text containing abbreviation of country.
m_name		String	Text containing name of country.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CUBEEXECUTION

Stores job execution information common to different kinds of cube jobs. Designed to be joined with M_EXECUTION and either M_CUBEPOPEXECUTION or M_NEXTPOPEXECUTION. Version 2.7 does not store useful data in this table.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mubid	FK	Pointer	Job execution identifier. ID in M_EXECUTION.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CUBEPOPEXECUTION

Stores information about specific CubePop executions.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mubid	FK	Pointer	Job execution identifier. ID in M_EXECUTION.
dimensionStartTime		Timestamp	Timestamp indicating start of overall dimension population.
dimensionEndTime		Timestamp	Timestamp indicating completion of overall dimension population.
cubeStartTime		Timestamp	Timestamp indicating start of overall cube population.
cubeEndTime		Timestamp	Timestamp indicating completion of overall cube population.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CUBEPOPULATION

Stores cubes populated during specific CubePop executions.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mubid	FK	Pointer	Job execution identifier. ID in M_EXECUTION.
cube		String	Name of cube populated.
createdDate		Date	Timestamp indicating row creation.

Column	Key	Data Type	Description
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CUSTOMERFORM

Stores information about who created and modified customer forms.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
customerFormTemplate	FK	Pointer	Template used for this customer form. ID in M_CUSTOMERFORMTEMPLATE.
createdBy	FK	Pointer	Employee who created form record. ID in M_PERSON.
modifiedBy	FK	Pointer	Employee who last modified form record. ID in M_PERSON.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CUSTOMERFORMSTATUSTYPE

Stores possible entries for the status of a customer form template.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_value		String	Text describing form status.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CUSTOMERFORMTEMPLATE

Stores information about the form templates created for users to enter data on a customer form.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Text identifying the form template
description		String	Text providing descriptive information about the form template.
formDescriptor		String	Text containing the form descriptor component.
status	FK	Pointer	Status of the form template. ID in M_CUSTOMERFORMSTATUSTYPE.
createdBy	FK	Pointer	Employee who created the form template. ID in M_PERSON.

Column	Key	Data Type	Description
modifiedBy	FK	Pointer	Employee who last modified the form template. ID in M_PERSON.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CUSTOMTIMEGRAININFO

All custom time grain intervals, both start and end point. Contains one row per interval, such as for each distinct "FiscalQuarter." Running m cubepop -offline builds this table based on the information in M_CUSTOMTIMEGRAINSEED and M_CUSTOMTIMEGRAINTYPE.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
grainName		String	Text that defines the custom time grain name.
dimValueDescription		String	Description of this interval, such as "FiscalQ2-2006."
firstIncludedDate		Timestamp	First instant in the interval.
firstExcludedDate		Timestamp	First instant no longer in the interval.
dimValueSymKey		String	Token that represents the time dimension entity for this custom grain interval.
dimValueNumKey		Long	Number that represents the time dimension entity for this custom grain interval.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CUSTOMTIMEGRAINSEED

Defines the starting point of each custom time grain interval. Contains one row per interval, such as for each distinct "FiscalQuarter."

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
grain	FK	Pointer	Type of the specified custom time grain. ID in M_CUSTOMTIMEGRAINTYPE.
dimValueDescription		String	Description of this interval, such as "FiscalQ2-2006."
firstIncludedDate		Timestamp	First instant in the interval.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_CUSTOMTIMEGRAINTYPE

Defines the names and descriptions of the custom time grains in the deployment, such as "Season" or "FiscalQuarter." Contains one row per custom grain.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Text that defines the custom time grain name, such as "FiscalQuarter." Must be unique within this table.
description		String	Prose description of the custom grain, such as "Fiscal Quarters are like normal quarters, but more remunerative."
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_DATECOUNT

Stores one row for every date of interest. Used as a "data chunk" table for the DateCount base metric.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_day		Date	Date of interest.
m_count		Integer	Number of days occurring on that date, such as 1.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_DICECONVERSIONEXECUTION

Stores information about executions of DICE conversions, such as changed-data-capture parameters, number of rows affected, and number of phantoms generated. Join with M_EXECUTION for root execution information.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mubid	FK	Pointer	Job execution identifier. ID in M_EXECUTION.
dataSetName		String	Text describing the data set processed during this conversion.
dataFromTime		Timestamp	First date requested by conversion.
dataToTime		Timestamp	Last date requested by conversion.
stableDataFromTime		Timestamp	The first date in the stable period, the period over which data is not changed during future conversions.
stableDataToTime		Timestamp	The last date in the stable period, the period over which data is not changed during future conversions.
rowsInserted		Long	Number of new rows inserted during this execution.
rowsUpdated		Long	Number of rows updated during this execution.
rowsDeleted		Long	Number of rows deleted during this execution.

Column	Key	Data Type	Description
rowsRejected		Long	Number of rows rejected during this execution, not resulting in an insertion or deletion.
phantomsGenerated		Long	Number of phantom identifiers created during this execution.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_DIMENSIONPOPULATION

Stores dimensions populated during specific executions.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mubid	FK	Pointer	Job execution identifier. ID in M_EXECUTION.
dimension		String	Name of dimension populated.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_DURABLEPARAMETERSMAP

Stores a map of parameters used with a durable job.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_DURABLEPARAMETERSMAPENTRY

Stores an individual key-value pair for a durable parameter map.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mapKey		String	Name of the parameter.
primValueStr		String	String form of the value, if applicable.
primValueTypeStr		String	Type of the string form, if applicable
refValue	FK	Pointer	Reference to the complex value, if applicable. The value will either be a primitive or a reference value. ID in M_DURABLEPARAMETERSMETABEANREFERENCE.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_DURABLEPARAMETERSMAP_ENTRIES_REL

NOTE: This table name is subject to name mangling. With the version 2.5 name mangler, the table name is M_DURABLEPARA_O7F3_ENTRIES_REL. With the version 2.2 name mangler, the table name is M_DURABLEPARAME_5B_ENTRIES_REL.

Stores an array of entries for each durable parameters map.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
durableParametersMap	FK	Pointer	Identifier for a particular durable parameters map. ID in M_DURABLEPARAMETERSMAP. Cannot be NULL.
entries_val	FK	Pointer	Valid entry for a durable parameters map. ID in M_DURABLEPARAMETERSMAPENTRY.
entries_val_ind		Integer	Index for this entry. Cannot be NULL.

M_DURABLEPARAMETERSMETABEANREFERENCE

NOTE: This table name is subject to name mangling. With the version 2.5 name mangler, the table name is M_DURABLEPARAMETERSMETABE_2FJU. With the version 2.2 name mangler, the table name is M_DURABLEPARAMETERSMETABEAN_1B.

Stores references to a metabean of any type from a durable parameter map.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
referenceType		String	Metabean type referenced
referenceId		Long	ID of referenced metabean.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_DURABLESCHEDULABLEWRAPPER

Stores individual durable scheduled jobs.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
originatingScheduler		Service	Scheduler that created this job.
durableSchedulable		Service	Template for running this job.
taskName		String	Name of this job.
m_schedule		String	Schedule for running this job.
scheduledTaskLog	FK	Pointer	History of when this job has been run. ID in M_SCHEDULEDTASKLOG.

Column	Key	Data Type	Description
durableParameters	FK	Pointer	Parameters to pass the **durableSchedulable** when running it. ID in M_DURABLEPARAMETERSMAP.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_EMPLOYMENTSTATUSTYPE

Defines possible types of employee status.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Name that describes an employee status.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_EXECUTION

Stores information about each run of an executable. M_EXECUTION is the root table, designed to be joined with other execution-related tables that provide more specific information about an execution, based on the type of job. Other tables include:

- M_MAINSERVICEEXECUTION
- M_DICECONVERSIONEXECUTION
- M_CUBEEXECUTION
- M_CUBEPOPEXECUTION
- M_NEXTPOPEXECUTION
- M_POPULATION
- M_CUBEPOPULATION
- M_DIMENSIONPOPULATION

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier (a run of an executable).
executable		Service	Name of an executable component.
parentExecution	FK	Pointer	Executable run that launched this executable. ID in M_EXECUTION. Runs not launched by an executable, contain a NULL value.
startTime		Timestamp	Time that this executable run started.
endTime		Timestamp	Time that this executable run finished. Runs not completed contain a NULL value.
statusSummary	FK	Pointer	Status of this executable run. ID in M_EXECUTIONSTATUSSUMMARY.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_EXECUTIONMESSAGE

Stores messages produced by audited job executions. Version 2.7 only stores error messages.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
execution	FK	Pointer	Associated execution run. ID in M_EXECUTION.
messageType	FK	Pointer	Type of message. ID in M_EXECUTIONMESSAGETYPE
m_timestamp		Timestamp	Time message was written.
ordinality		Long	Number that indicates the relative order of the message, compared to other messages generated during the same execution.
body		BigString	Text containing message.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_EXECUTIONMESSAGETYPE

Defines types of execution messages.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
displayname		String	Name indicating message type. There are three default types: • Error • Warning • Info **NOTE:** This version only uses Error messages.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_EXECUTIONSTATUSSUMMARY

Defines possible execution statuses.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
displayname		String	Name summarizing status of an execution. There are five default statuses: • Initializing • Running • Failed • Cancelled • Completed
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_GROUPTYPE

Defines types of groups.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Name of this group type.
hasPeople		Boolean	Indicates whether this is a group of employees.
subgroupOf	FK	Pointer	Parent group type in the group hierarchy. ID in M_GROUPTYPE (this table).

Column	Key	Data Type	Description
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_HEADCOUNT

Stores one row for every person-day of interest. Used as a "data chunk" table for the HeadCount base metric.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
who	FK	Pointer	Person being tallied. ID in M_PERSON.
m_when		Date	Date of interest.
m_count		Integer	Tally of person-days for this person and date (e.g., 1).
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_IDENTITY

Defines valid identity values in particular namespaces.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
namespace	FK	Pointer	Namespace where identity value is valid. ID in M_NAMESPACE.
m_value		String	Text containing valid identity value in specified namespace.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Not used in this table.
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_IDENTITYASSIGNMENT

Links a source system identifier to a resource in the data mart repository, such as a person or group, during a particular time period.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_identity	FK	Pointer	Source system identifier. ID in M_IDENTITY.
resourceType		String	Component name of a metabean type. This metabean type specifies the resource with this identity assignment. Typical values are: • /com/merced/datamart/repository/MGroup • /com/merced/datamart/repository/Person

Column	Key	Data Type	Description
resourceId		Long	Identifier in the metabean type specified in **resourceType**. Usually a person or group identifier from M_MGROUP or M_PERSON.
fromTime		Timestamp	Time when this identity is first applicable to this person or group.
toTime		Timestamp	Time when this identity is no longer applicable to this person or group.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Not used in this table.
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_INSTITUTION

Defines a set of institutions. Not commonly used in Merced Performance Suite applications.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Name of institution.
location	FK	Pointer	Location of institution. ID in M_LOCATION.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_JOBTITLE

Defines a list of valid job titles.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Text identifying a job title.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_LOCATION

Stores information about an employee's location.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
addressLine1		String	First line in address.
addressLine2		String	Second line in address.
addressLine3		String	Third line in address.
addressLine4		String	Fourth line in address.
postcode		String	Postal code.
country	FK	Pointer	Country name. ID in M_COUNTRY.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_LOGINRECORD

Tracks login events in the Merced Performance Suite application. Each user login creates a row in this table.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mUser	FK	Pointer	User who logged in. ID in M_MUSER.
loginTime		Timestamp	Time user logged in.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_LOGINTICKET

Tracks user login and session status. Used during failover to restore a user's login and session when the proxy routes the user to a different application engine. Login Ticket Sweeper periodically removes unneeded old rows to maintain the table's size.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
user	FK	Pointer	User who logged in. ID in M_MUSER.
sessionKey		String	Key that identifies session.
lastAccessTime		Timestamp	Time user last accessed session.
loginTime		Timestamp	Time user logged in.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_MAINSERVICEEXECUTION

Stores information common to all executions of main services (services started with the m command, such as application engines, CubePop, or DICE).

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mubid	FK	Pointer	Job execution identifier. ID in M_EXECUTION.
args		String	Command line arguments passed to the command.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_MESSAGEBULLETIN

Stores all the messages in the system.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
sender	FK	Pointer	Any principal: currently a person, group, or non-player character (the system). ID in M_METABEANREFERENCE.
recipients		String (4000)	List of recipients as described by the sender. **NOTE:** This can include groups and overlapping sets of people.
priority	FK	Pointer	Priority of message bulletin. ID in M_BULLETINPRIORITY.
subject		String (256)	Subject line of message
body		String (4000)	Body text of message.
link		String (4000)	A URL.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_MESSAGEBULLETINCOPY

Stores a copy of each message as received by the individual recipients. Contains one record for each recipient.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
bulletin	FK	Pointer	Message bulletin that is sent as a copy. ID in M_MESSAGEBULLETIN.
recipient	FK	Pointer	Employee who receives copy of message bulletin. ID in M_PERSON.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_METABEANREFERENCE

Stores references to any defined metabean. By referencing rows in this table, another column can possibly reference one of multiple tables rather than always referencing single tables. For example, if a column references a person, you can create a foreign key to the M_PERSON table. However, if a column sometimes references a person but sometimes references a group, you can create a reference to this table, which can store references to rows in both M_PERSON and M_MGROUP.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
referenceType		String	Name of metabean type referenced, such as Person or MGroup.
referenceId		Long	ID of the referenced row, such as the ID in M_PERSON or M_MGROUP.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_MGROUP

Stores groups defined in Merced Performance Suite org chart.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Name that identifies group.
groupType	FK	Pointer	Type of this group. ID in M_GROUPTYPE.
parentGroup	FK	Pointer	Identifier of this group's parent group. ID in M_MGROUP (this table).
attr	FK	Pointer	Attribute associated with this group. ID in M_MGROUPATTR.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_MGROUPATTR

Defines attributes available to groups.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Name that identifies this attribute.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_MGROUP_PARENTGROUP_HIST

Stores a group's parent group at any date in time.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mgroup	FK	Pointer	Group identifier. ID in M_MGROUP.
parentGroup	FK	Pointer	Group identifier of parent group. ID in M_MGROUP.
from_date		Date	First date that this row is in effect.
end_date		Date	Last date that this row is in effect.

M_MUSER

Stores user names employees use to log in to Merced Performance Suite.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
username		String	User name to log in.
password		String	Password to log in.
person	FK	Pointer	Person associated with this login name. ID in M_PERSON.

Column	Key	Data Type	Description
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_NAMESPACE

Defines namespaces (usually source systems) used in Merced Performance Suite.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Name that identifies namespace.
displayName		String	Name that appears in user interface.
appliesTo		String	Metabean type that the identities in this namespace apply to.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_NEXTPOPEXECUTION

Stores information about execution of NextPop jobs.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mubid	FK	Pointer	Job execution identifier. ID in M_EXECUTION.
waitForCubePop		Boolean	Flag indication whether the NextPop execution is configured to wait for CubePop.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_NONPLAYERCHARACTER

Defines principals that do not correspond to an actual person or group. For example, you might define a "System" principal, which is sends certain alerts.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Text that describes this principal.
emailAddress		String	E-mail address.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_PAGEFETCHER

Defines tasks for Page Fetcher. Page Fetcher periodically retrieves and caches specified URLs, usually reports that require lengthy computation times. Cached versions of these pages load more quickly.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
url		String (4000)	A URL that Page Fetcher caches.
username		String	Username that Page Fetcher uses to log in when retrieving the URL.
m_schedule		String	Text that indicates the schedule at which Page Fetcher caches this page. See Merced Support (https://support.mercedsystems.com/wiki/Schedule_Property_Converter_Reference) for more information about the syntax.
enabled		Boolean	Flag that indicates whether Page Fetcher implements this task.
notes		String	Optional text that describes the task.

Column	Key	Data Type	Description
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_PERSON

Stores list of employees defined in Merced Performance Suite org chart.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
firstName		String	First name.
middleName		String	Middle name.
lastName		String	Last name.
birthDate		Date	Birth date.
startDate		Date	First date of employment.
endDate		Date	Last date of employment.
jobTitle		String	Job title.
salary		String	Salary.
proposedSalary		String	Proposed salary.
salaryIncreasePct		Float	Salary increase percentage.
contactInfo	FK	Pointer	Record containing contact info for employee. ID in M_CONTACTINFO.
employmentStatus	FK	Pointer	Employment status of this employee. ID in M_EMPLOYMENTSTATUSTYPE.
nextReviewDate		Date	Date of next review.
lastReviewInterval		Integer	Length between reviews.

Column	Key	Data Type	Description
attritionReason	FK	Pointer	Reason for employee attrition. ID in M_ATTRITIONREASONTYPE.
location	FK	Pointer	Location of this employee. ID in M_LOCATION.
parentGroup	FK	Pointer	Employee's group. ID in M_MGROUP.
title	FK	Pointer	Employee's title type. ID in M_TITLETYPE
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive • 2 indicates phantom • 3 indicates inactive phantom
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_PERSON_CAREEREVENTS_REL

Stores a list of career events for each employee.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
person	FK	Pointer	Employee. ID in M_PERSON. Cannot be NULL.
careerEvents_val	FK	Pointer	Career event. ID in M_CAREEREVENT.
careerEvents_val_ind		Integer	Index for this entry. Cannot be NULL.

M_PERSON_EMPLOYMENTSTATUS_HIST

Stores an employee's employment status at any date in time.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
person	FK	Pointer	Employee. ID in M_PERSON.
employmentStatus	FK	Pointer	Employment status. ID in M_EMPLOYMENTSTATUSTYPE.
from_date		Date	First date that this row is in effect.
end_date		Date	Last date that this row is in effect.

M_PERSON_PARENTGROUP_HIST

Stores an employee's parent group at any date in time.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
person	FK	Pointer	Employee. ID in M_PERSON.
parentGroup	FK	Pointer	Parent group. ID in M_MGROUP.
from_date		Date	First date that this row is in effect.
end_date		Date	Last date that this row is in effect.

M_PERSON_TITLE_HIST

Stores an employee's title type at any date in time.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
person	FK	Pointer	Employee. ID in M_PERSON.
title	FK	Pointer	Title type. ID in M_TITLETYPE.
from_date		Date	First date that this row is in effect.
end_date		Date	Last date that this row is in effect.

M_POPULATION

Stores common information about executions that populate dimensions or cubes in the OLAP data mart repository. Designed to join with M_CUBEPOPULATION and M_DIMENSIONPOPULATION.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
mubid	FK	Pointer	Job execution identifier. ID in M_EXECUTION.
tableName		String	Name of table populated.
forceClear		Boolean	Flag that indicates whether **forceClear** parameter was true for this table population.
rowsInserted		Number	Number of rows inserted into table. **NOTE:** Only recorded for cubes; stores NULL for all dimensions.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_SALARYGRADE

Defines salary grades.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Name that identifies a salary grade.
minSalary		Long	Minimum salary at this grade.
maxSalary		Long	Maximum salary at this grade.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_SCHEDULEDJOB

This table supports landingpad, a deprecated feature.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
displayName		String	
primaryKeyName		String	
startTime		Timestamp	
endTime		Timestamp	
status		Integer	
taskStatus		Integer	
runType		Integer	
runBy	FK	Pointer	ID in M_MUSER.
errorMessage		String	
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_SCHEDULEDJOB_TASKS_REL

This table supports landingpad, a deprecated feature. Stores an array of tasks for a scheduled job.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
scheduledJob	FK	Pointer	Scheduled job. ID in M_SCHEDULEDJOB. Cannot be NULL.
tasks_val	FK	Pointer	Task included in scheduled job. ID in M_SCHEDULEDTASK.
tasks_val_ind		Integer	Index value for the row's scheduled task. Cannot be NULL.

M_SCHEDULEDTASK

This table supports landingpad, a deprecated feature. Stores scheduled tasks.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
taskName		String	Name that identifies this task.
primaryKeyName		String	
taskKey		String	
startTime		Timestamp	
endTime		Timestamp	
scheduledFromTime		Timestamp	
scheduledToTime		Timestamp	
numBeans		Long	
status		Integer	
fromTime		Timestamp	
toTime		Timestamp	
lastCommittedToTime		Timestamp	
errorMessage		String	
sourceFile		String	

Column	Key	Data Type	Description
settleDuration		Long	
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_SCHEDULEDTASKLOG

Stores a history of scheduled jobs.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
scheduledTime		Timestamp	Time at which the job was scheduled.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_SCHEDULEDTASKLOGENTRY

Stores records of individual runs of scheduled jobs.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
scheduledExecutionTime		Timestamp	Time task was scheduled for execution.
executionStartTime		Timestamp	Time task execution started.
executionEndTime		Timestamp	Time task execution finished.
status		Integer	Integer indicating status of task execution.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_SCHEDULEDTASKLOG_ENTRIES_REL

A join table that associates the entire history of a scheduled job with the records of its individual runs.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
scheduledTaskLog	FK	Pointer	Scheduled task log. ID in M_SCHEDULEDTASKLOG.
entries_val	FK	Pointer	Log entry for a scheduled task. ID in M_SCHEDULEDTASKLOGENTRY.
entries_val_ind		Integer	Index entry for the row's scheduled task log entry. Cannot be null.

M_TASKBULLETIN

Stores all the tasks in the system.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
sender	FK	Pointer	Any principal: currently a person, group, or non-player character (the system). ID in M_METABEANREFERENCE.
recipients		String (4000)	Text indicating recipients.
priority	FK	Pointer	Priority of this task bulletin. ID in M_BULLETINPRIORITY.
subject		String (256)	Subject line.
body		String (4000)	Body text of task bulletin.
link		String (4000)	A URL.
dueDate		Timestamp	Date task is due for completion.
status	FK	Pointer	Status of task. ID in M_TASKSTATUS.
notifyOnComplete		Boolean	Flag indicating whether to notify sender when task is completed.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_TASKBULLETINCOPY

Stores a copy of each task as received by the individual recipients. Contains one record for each recipient.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
bulletin	FK	Pointer	Task bulletin. ID in M_TASKBULLETIN.
recipient	FK	Pointer	Person who receives copy of task bulletin. ID in M_PERSON.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_TASKSTATUS

Defines possible statuses of tasks.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String (40)	Name that identifies task status.
displayName		String (40)	Name user interface displays for this task status.
ordinalValue		Integer	Sort order for this task relative to others.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_TIMEZONE

Stores list of valid time zones.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
displayName		String	Name that user interface displays for this time zone.
javaName		String	Time zone ID supported by Java, such as "America/Los_Angeles" for the U.S. Pacific time zone. See http://java.sun.com/j2se/1.4.2/docs/api/java/util/TimeZone.html.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_TITLETYPE

Defines possible title types (employee roles) in Merced Performance Suite org chart.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Name that identifies this title type (employee role).
bossWithin	FK	Pointer	Primary group type of employees with this title type. ID in M_MGROUP.
viewAnyGroupType		Boolean	
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_TRANSACTIONALREPORT

Defines transactional reports.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
metaBeanType		Component	Metabean type that this transactional report queries.
m_name		String	Unique name that identifies this transactional report.

Column	Key	Data Type	Description
shortName		String	Unique short name for this transactional report.
description		String	Text describing this transactional report.
beanView		String	Map that links report columns to particular properties; determines what data the report shows.
timePhrase		Component	Defined phrase that specifies the time period retrieved by the transactional report.
personPhrase		Component	Defined phrase that specifies the person retrieved by the transactional report.
acl		Component	Access control list that defines a set of users allowed to view this transactional report.
sortColumn		Integer	Integer representing column on which the transactional report sorts data. (0 corresponds to first column, 1 corresponds to second column, etc.)
sortDescending		Boolean	Flag indicates whether transactional report sorts in descending order.
numRows		Integer	Number of data rows that the transactional report initially shows on each page.
useOptimizer		Boolean	Flag indicates whether to use transactional report optimizer.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_TRANSACTIONALREPORTDIMENSIONMAPENTRY

NOTE: This table name is subject to name mangling. With the version 2.5 name mangler, the table name is M_TRANSACTIONALREPORTDIME_OEAU. With the version 2.2 name mangler, the table name is M_TRANSACTIONALREPORTDIMENS_75.

Stores valid map entries for the transactional report dimension map.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_key		String	Text that indicates type of field specified by entry. Either: • person • time
m_value		String	Property in source metabean type that contains the person or time that the transactional report searches for.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

M_TRANSACTIONALREPORT_ATTRIBUTES_REL

NOTE: This table name is subject to name mangling. With the version 2.5 name mangler, the table name is M_TRANSACT_ATDM_ATTRIBUTES_REL. With the version 2.2 name mangler, the table name is M_TRANSACTIO_7F_ATTRIBUTES_REL.

Stores an array of attribute components for transactional reports.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
transactionalReport	FK	Pointer	Transactional report identifier. ID in M_TRANSACTIONALREPORT. Cannot be NULL.
attributes_val		Component	Group attribute component of users shown the link to this transactional report in certain directory views.
attributes_val_ind		Integer	Index for this attribute. Cannot be NULL.

M_TRANSACTIONALREPORT_DIMENSIONMAPENTRIES_REL

NOTE: This table name is subject to name mangling. With the version 2.5 name mangler, the table name is M_TRANSAC_ATDM_DIMENS_PCYH_REL. With the version 2.2 name mangler, the table name is M_TRANSACTI_7F_DIMENSIO_OM_REL.

Stores an array of transactional report dimension map entries, which indicate the person and time fields searched, for transactional reports.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
transactionalReport	FK	Pointer	A transactional report. ID in M_TRANSACTIONALREPORT. Cannot be NULL.
dimensionmapentries_val	FK	Pointer	A transactional report dimension map entry. ID in M_TRANSACTIONALREPORTDIMENSIONMAPENTRY
dimensionmapentries_val_ind	FK	Integer	Index for this dimension map entry. Cannot be NULL.

M_TRANSACTIONALREPORT_PARENTDIRECTORIES_REL

NOTE: This table name is subject to name mangling. With the version 2.5 name mangler, the table name is M_TRANSAC_ATDM_PARENT_IMJK_REL. With the version 2.2 name mangler, the table name is M_TRANSACTI_7F_PARENTDI_RJ_REL.

Stores an array of parent directory components for transactional reports.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
transactionalReport	FK	Pointer	A transactional report. ID in M_TRANSACTIONALREPORT. Cannot be NULL.
parentdirectories_val		Component	A parent directory component for this transactional report.
parentdirectories_val_ind	FK	Integer	Index for this parent directory. Cannot be NULL.

M_TRANSACTIONALREPORT_ROLES_REL

NOTE: This table name is subject to name mangling. With the version 2.5 name mangler, the table name is M_TRANSACTIONAL_ATDM_ROLES_REL. With the version 2.2 name mangler, the table name is M_TRANSACTIONALRE_7F_ROLES_REL.

Stores an array of role components for transactional reports.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
transactionalReport	FK	Pointer	Transactional report identifier. ID in M_TRANSACTIONALREPORT. Cannot be NULL.
roles_val		Component	Role component of users shown the link to this transactional report in certain directory views.
roles_val_ind		Integer	Index for this role component. Cannot be NULL.

M_UNITFORMAT

Stores definitions of metric units.

Column	Key	Data Type	Description
id	PK	Integer	Unique row identifier.
m_name		String	Text that identifies this unit.
format		String	Code that describes how the application presents data assigned this unit in reports and dashboards.
excelFormat		String	Code that describes how the application formats data assigned this unit when exporting to Microsoft Excel xls worksheet files.
createdDate		Date	Timestamp indicating row creation.
modifiedDate		Date	Timestamp indicating last modification.

Column	Key	Data Type	Description
isInactive		Integer	Integer indicating whether row is active: • 0 indicates active • 1 indicates inactive
siComponentName		String	Indicates component name if row loaded as a static instance (seed data).
versionId		Integer	Integer indicating row version. Incremented each time system modifies row.

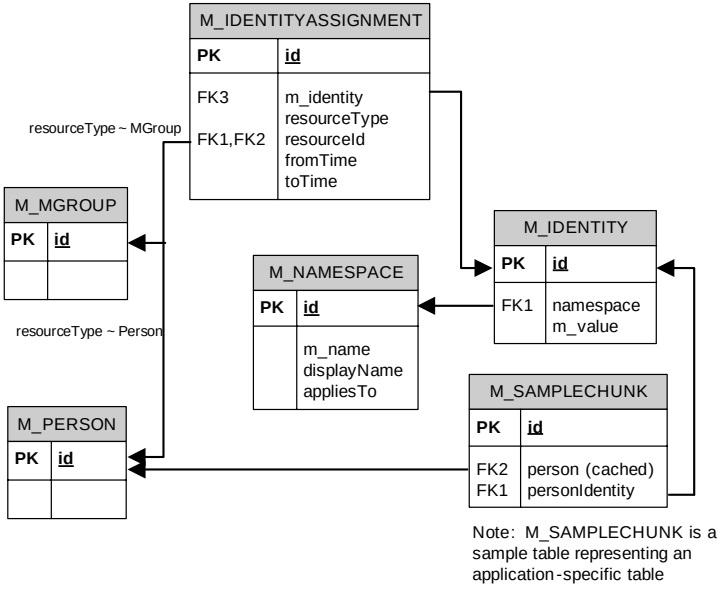
E Schema Diagrams

This appendix contains schema diagrams showing the relationships between some core tables in the Merced Performance Suite data mart. Use these schema diagrams in conjunction with the table descriptions in Appendix D, "Data Dictionary." Schema diagrams are not available for all tables in the dictionary.

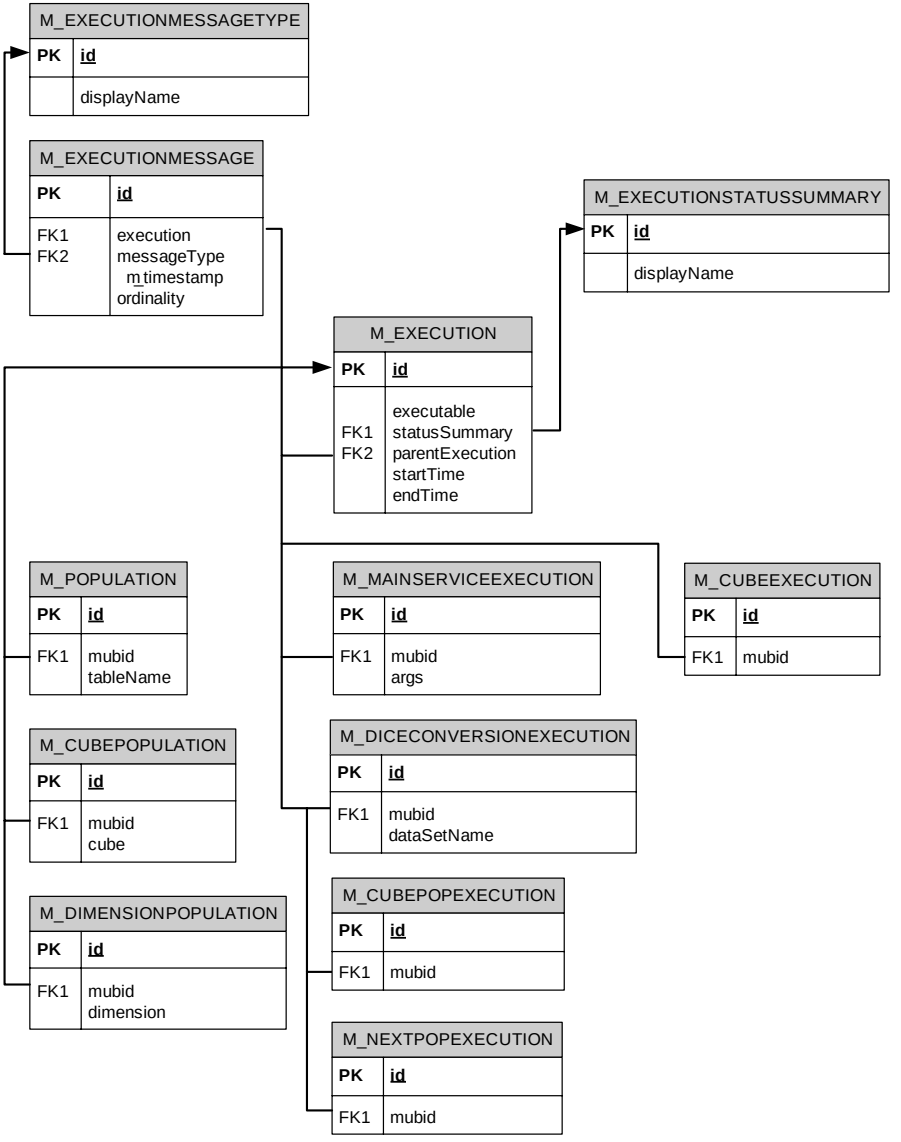
This appendix contains schema diagrams for tables in these categories:

- Identities
- Job executions
- Org chart

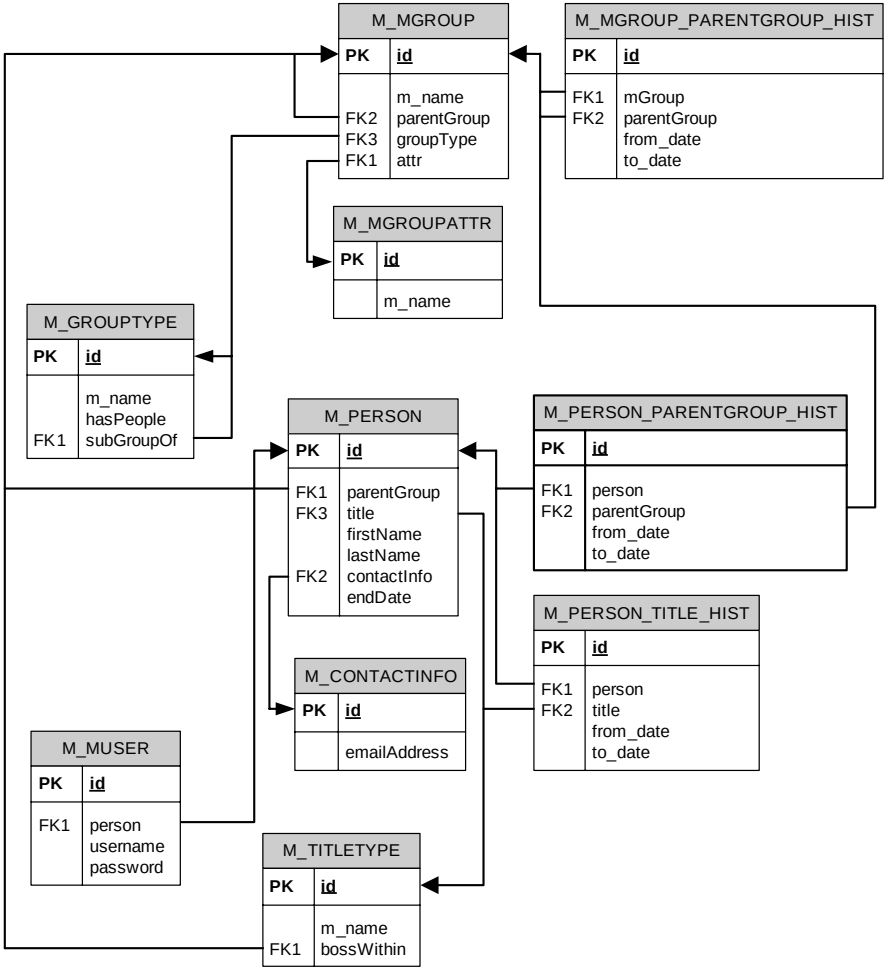
Identities


M_IDENTITYASSIGNMENT
PK
id
FK3
FK1,FK2
m_identity
resourceType
resourceId
fromTime
toTime
resourceType ~ MGroup
M_MGROUP
PK
id
M_IDENTITY
PK
id
FK1
namespace
m_value
M_NAMESPACE
PK
id
m_name
displayName
appliesTo
resourceType ~ Person
M_SAMPLECHUNK
PK
id
FK2
FK1
person (cached)
personIdentity
M_PERSON
PK
id
Note: M_SAMPLECHUNK is a sample table representing an application-specific table

Job executions



Org chart


M_MGROUP
PK id
m_name
FK2 parentGroup
FK3 groupType
FK1 attr
M_MGROUP_PARENTGROUP_HIST
PK id
FK1 mGroup
FK2 parentGroup
from_date
to_date
M_MGROUPATTR
PK id
m_name
M_GROUPTYPE
PK id
m_name
hasPeople
FK1 subGroupOf
M_PERSON
PK id
FK1 parentGroup
FK3 title
firstName
lastName
FK2 contactInfo
endDate
M_PERSON_PARENTGROUP_HIST
PK id
FK1 person
FK2 parentGroup
from_date
to_date
M_PERSON_TITLE_HIST
PK id
FK1 person
FK2 title
from_date
to_date
M_CONTACTINFO
PK id
emailAddress
M_MUSER
PK id
FK1 person
username
password
M_TITLETYPE
PK id
m_name
FK1 bossWithin

Index

A

access control list, definition 117
Active Directory
  creating user for SPNEGO 81
  domain controller registration 79
  NTLM support 74
  SPNEGO support 78
Admin Server
  configuring 49
  property 47
  purpose 48
  requirements 49
  scheduling concern 98
admin user
  configuring for LDAP 73
  validation service for 64
Adobe Reader, requirement 129
AMT system, tables for 166
Apache log files
  customizing format 43
  managing 99
  proxy restart and 97
Apache Web server 2, 126
application
  customizing 10
  SPNEGO authentication, configuring 81
application server. *See* engine
application server machine
  configuration 8
  determining number 127
  disk space, monitoring 95
  engines, maintaining 97
application server machine (*continued*)
  hardware requirements 126
  locating in network 15
  maintaining 94–99
  monitoring CPU load 95
  monitoring memory usage 96
  processes on 92
  proxies, maintaining 97
  RAM allocation 7
  set up 19–29
  software requirements 126
archiving log files 98
authentication
  description 60
  LDAP 67
  NTLM 74
  SPNEGO 78
  standard 63
  WebSEAL 88
authorization
  description 60
  single sign-on 61
automating
  connectivity verification 105
  database checks 106
  engine restart 107
  log file archiving 107
  log file creation 43, 47, 98
  log file monitoring 106
  proxy restart 107
avoiding system conflicts 92

B
backing up database 100
base metrics
  assigning to cubes 141
  definition 117
  deleting 153
  loading 153
base port 50
BEL expressions 133
bin directory 11
browser
  configuring for NTLM authentication 75
  configuring for SPNEGO authentication 83
  pointing session to engine 41
  software requirements 129

C
cache, temporal properties 48
calculated metrics
  definition 117
  deleting 153
  loading 153
case sensitivity 35
cbean repository
  archiving 150
  importing specifications into 150
  initializing 151
  purging 151
cbeans
  exporting to file 148
  importing into repository 150
  registry for 151
Cbuild, definition 118
Certificate Authority 16
certificate for SSL 16
charts, displaying in Unix 21
client. *See* browser
cluster
  configuring 38
  description 2
  firewall configuration 15
  hardware assignment 7
  minimum configuration 3
cluster component 38
CMS, definition 118
collecting data
  SPNEGO configuration 79
  system jobs 93
command syntax 133
commands 131–164
component
  See also individual component names
  defining 9
  finding properties of 134
Configuration component
  JDBC configurations, setting 56
  time zone, setting 20
  var directory 109
  viewing properties of 134
connection pool 56
Content Management System. *See* CMS
Creator 9
cube universe component 142
CubePop
  application server machine for 8
  cleanup command option 141
  command to run 138
  definition 118
  details 142
  failure to start 8
  failures, recovering from 114
  force optimization option 139
  log files 108
  offline option 139
  online option 138
  only-pack-cubes option 141
  semaphores 142
cubes, definition 118
customer layer 11
customizing an application 10

D
data dictionary 165
data mart
  See also OLAP data mart, OLTP data mart
  definition 119
  schema diagrams 229–233
  tables in 173–228
database
  administrative tools 100
  backing up 100
  configuring 54
  connection pools 56
  connectivity issues in log files 106
  connectivity, monitoring 104
  DB2 set up 32
  definition 119
  development, creating for 5
  initializing 160
  installing 30

database (*continued*)
limiting connections 57
locating on network 15
metrics, deleting 153
metrics, loading 153
modifying reports schema 162
modifying schema 161
monitoring connection usage 106
monitoring connectivity 104
monitoring tablespace 100
Oracle initialization file 31
password encryption 52, 135, 147
restoring from failed CubePop 115
running SQL commands 135
schema diagrams 229–233
SQL Server settings 36
staging, creating for 5
statistics, computing 139, 139, 157, 158
tables in 173–228
tablespace, monitoring 100
database server machine
hardware requirements 128
maintaining 100
set up 30
software requirements 128
DB2 UDB
configuration variables 33
gathering statistics 158
registry variables 33
tablespace configuration 34
user privileges 34
DBMS, definition 119
debugging
log files 110
SPNEGO authentication 84–87
deleting metrics 153
development environments
description 5
hardware for 7
validation service for 64
DICE
application server machine for 8
available jobs, listing 144
command to start job 144
definition 119
failure to start 8
failure, recovering from 115
log files 108
semaphores 163
dimension table, definition 120
dimensions
changing 139
definition 120
disk space, monitoring 95
dispatching validation service component 65
DISPLAY variable 22
domain controller 80
dumpcc command 134

E

Employee Manager, definition 120
encryption
command for passwords 147
database passwords 52
network 15
engine
Admin Server designation 48
component for 45
configuring 45
description 2
failure to start 8, 46
hardware configuration 8
inadequate number, symptoms of 114
log files 29, 47
m mserver command 155
memory requirements 8
monitoring connectivity 101
name, specifying 45
pointing browser session to 41
port assignments 49
registering system service for 26, 156
restarting regularly 97
running 24
running as scheduled task 28
running as service 25
running with m mserver command 27
specifying machine for 39
starting with mserver command 155
stopping with mserver command 155
troubleshooting 113
unregistering system service 27
user to run process 24
environments
databases for 5
description 4
hardware differences 7

errors
  database connectivity 106
  engine log files 114
  log files, general 110
  SPNEGO authentication 84
Excel, requirement for 129
exporting cbeans to file 148

F
facility requirements 14
fact table, definition 120
File Sniffer, Admin Server task 48
files, splitting 149
firewall 18
forging headers 89
forms, tables for 167

G
goal metrics
  deleting 153
  loading 153
grain changes, effects of 139

H
hardware requirements
  application server machines 126
  database server machines 128
  discussion 7
  installation environment 14
head count job, tables for 167
Heap Monitor 96
help option in commands 137
host component 40
HTTP headers 88

I
identities
  schema diagram 230
  tables for 167
importing cbeans 150
inadequate memory indicator 96
installing
  database 30
  Java 23
  operating system user 14
  Python 23
  servers in network 15
  X server 21
integration testing 4
Internet Explorer requirement 129

J
Java
  implementation 9
  installing 23
  monitoring memory consumption 96
JAVA_HOME variable 23
JavaScript requirement 129
JDBC configuration component
  configuring 54
  connection pool 57
  encrypted passwords 52
  registering 55
  testing 58
JDBC, definition 120
job executions
  schema diagram 231–??
  tables for 167
jsql command 135

K
Kerberos 78

L
layers in code 10
LDAP
  admin user, configuring 73
  configuring validation 73
  description 67
  SSL, using with 72
LDAP validation service component
  lookup 70
  simple 68
level-restricted calculated metrics. *See* LRCMs
lib directory 11
lightweight directory access protocol. *See* LDAP
Linux 8, 126
load balancers
  configuring standard port 25
  description 2
  monitoring connectivity 104
  troubleshooting 113
loading metrics 153
localconfig directory 11
localconfig layer 10
log files
  analysis techniques 110
  archiving 98
  database connectivity errors 106
  engine errors 114
  file size 98

log files (*continued*)
format 109
locations 108
maintaining size 97
monitoring automatically 106
monitoring size 95
login gatekeeper
description 60
SPNEGO configuration 82
login handler
description 60
SPNEGO 82
WebSEAL 88
login tables 168
Login Ticket Sweeper 48
LoginServletService component 63
LRCMs
deleting 153
loading 153

M

m command 137
m cubepop command 138
m dice command 144
m encrypt command 147
m export-cbeans command 148
m file-splitter command 149
m import-cbeans command 150
m load-metrics command 153
m mserver command
description 155
using to run engines and proxies 27
m optimize-db command 157
m optimize-repository command 158
m repository upgrader command 160
m schemaload-main command 161
m schemaload-reports command 162
m semaphore command 163
maintenance
automating tasks 107
avoiding system conflicts 92
database 100
engine 97
information required 93
proxy 97
memory
engine requirements 8
inadequate, indication of 96
monitoring usage 96
MERCED_HOME
creating directory 19
definition 121
directories in 11
metabean repository
description 55
estimating statistics for 158
matching with properties files 160
metrics 153
monitoring
automating jobs 105
CPU load 95
database connections 106
disk space 95
log files 110
memory usage 96
network connectivity 101
proxy connectivity 102
monitors 48

N

network
firewall configuration 15, 18
monitoring connectivity 101, 105
servers in subnet 15
SSL recommendation 15
NextPop
command to run 144
definition 121
failures, recovering from 114
troubleshooting 114
node
configuring 39
description 3
registering system service for 26
node component 39
NTLM authentication 74
NTLM authenticator component 74

O

ODBC, definition 121
OLAP data mart
cluster element 3
definition 122
loading data into 138, 144
OLTP data mart
cluster element 3
definition 122
loading data into 144
modifying schema 161, 162

OpenSSL tool kit 17
operating systems
  certified, application server 126
  installing application in 14
  monitoring memory 96
Oracle
  gathering statistics 157, 158
  initialization file 31
  physical structure 30
  tablespace 32
org chart
  definition 121
  schema diagram 233
  tables for 168

P

PACKAGE.ctl files 10
password encryption
  database passwords in properties files 52
  generating encrypted passwords 147
permissions, database 32
person definition 122
ports
  disabling 51
  IANA standard 25
  setting 49
  verifying connectivity 101
privileges, database 34
product code layer 10
production environment 4, 7
properties files
  cluster component 39
  configuration component 56
  dispatching validation service component 66
  engine 48
  host component 41
  JDBC configuration component 55
  LDAP validation service component 69, 72
  node component 40
  proxy component 44
  SpnegoNotLoggedInHandler 83
  uses of 9
properties, finding component values 134
proxy
  configuring 41
  description 2
  error page, custom 44
  hardware configuration 8
proxy (*continued*)
  log files 29, 43
  m mserver command 155
  monitoring connectivity 102
  port assignments 49
  registering system service 26
  restarting regularly 97
  running 24
  running as service 25
  running with m mserver command 27
  specifying machine 39
  starting with mserver command 155
  stopping with mserver command 155
  supported software 126
  unregistering system service 27
  user to run process 24
  viewing information 103
proxy component 41
Python, installing 23
PYTHONHOME variable 23

Q

quality assurance environment 4
quotes in commands 133

R

repositories 55
  See also cbean repository, metabean repository
roles, definition 122

S

scheduled jobs, tables for 169
scheduled tasks 28
schema diagrams 229–233
scope 134
secure sockets layer. *See* SSL
semaphores
  CubePop 142
  DICE 163
  DICE, manually releasing 115
server offset port 50
servers
  See also application server machine, database server machine, engine, proxy
  required hardware and software 125
  specifying machine for 39

single sign-on
  authorization 61
  description 59
  NTLM authentication 74
  overview 61
  SPNEGO authentication 78
  WebSEAL authentication 88
software requirements
  application server machines 126
  client machines 129
  database server machines 128
Solaris 8
splitting files 149
SPNEGO authentication
  application configuration 81
  browser configuration 83
  data required 79
  debugging 84–87
  domain controller registration 79
SpnegoNotLoggedInHandler component 82
SQL Server 35–36
SQL, definition 123
SSL
  Certificate Authority 16
  certificates 16
  configuring proxy 43
  definition 15
  generating certificate 17
  LDAP authentication 72
  purpose of 15
staging environment 4
standard authentication 63
system services
  default name 26
  proxy, specifying name 42
  purpose 25
  registering 26
  scheduling restart 28
  specifying user 27
  unregistering 27

T

tablespace
  creating 32
  DB2 UDB 34
  monitoring 100
Temporal Property Cleaner 48
testing applications 4
time grains, tables for 170
time zones, setting 20
TimeZoneManager component 20
transactional reports
  definition 123
  tables for 170
transactional searches
  definition 123
troubleshooting
  common issues 112
  CubePop 114
  DICE conversions 115
  engines 113

U

Universal Database. *See* DB2 UDB
Unix
  command syntax 133
  DISPLAY environment variable 21
  memory limits 8
  monitoring charts 22
  running engines and proxies 24
URL
  accessing engine directly 102
  accessing proxy information 103
  cluster layout 101
user
  database 32, 34
  installation 14
  running engine 27, 29
  running proxy 27
UserDataStatusConfiguration component 134

V

validation service component 63
validation services
  dispatching 64
  LDAP 67
  preconfigured 64
  purpose 60
var directory 11
variable
  DISPLAY 22
  JAVA_HOME 23
  PYTHONHOME 23

W

Web server. *See* proxy
WebSEAL authentication 88, 90

Windows
Admin Server limits 49
command syntax 133
memory limits 8
running engines and proxies 24

X

X server 21

Merced Systems, Inc.
333 Twin Dolphin Drive, Suite 500
Redwood Shores, CA 94065 USA

650-486-4000
fax 650-486-4001
info@mercedsystems.com
www.mercedsystems.com