IBM Initiate Master Data Service
Version 10 Release 0

Master Data Engine Installation Guide

GI13-2612-00

IBM Initiate Master Data Service
Version 10 Release 0

Master Data Engine Installation Guide

GI13-2612-00

Note

Before using this information and the product that it supports, read the information in “Notices and trademarks” on page 261.

Contents

Figures

Tables

Chapter 1. Introduction

The Master Data Engine is the core component of the IBM® Initiate® Master Data Service. The Master Data Engine comprises the database, business rules, and linking logic that support the functions available in IBM Initiate Master Data Service applications.

The Master Data Engine supports all IBM Initiate Master Data Service application service requests. For example the functions available in IBM Initiate Workbench, IBM Initiate Inspector, or IBM Initiate Enterprise Viewerall use the Master Data Engine logic.

The Master Data Engine is contained within a Java process wrapper (not an application server or servlet engine) that enables greater flexibility of feature development.

Through TCP/IP or HTTP socket connections for the various API calls, the MPINET server receives messages from the Message Broker Suite and clients, performs application interactions, and stores the data in the hub database. MPINET is a multi-threaded process and implements database connection pooling to decrease response time. Information about MPINET interactions is written to `.mlg` log files.

The Master Data Engine writes to the database through Open Database Connectivity (ODBC), a standard database access method. The DXI layer is a database-specific abstraction that is included to handle the variations in accessing databases through ODBC.

The IBM Initiate Master Data Service supports secure encrypted communication between the Master Data Engine and clients through Socket Layer Security (SSL). Two-way SSL is the default configuration. For U.S. federal government installations, the engine can be configured for FIPS compatibility. As well, all authentications to the Master Data Engine are done through an internal Java-based directory server that manages user, group, and permission information. The internal LDAP directory server is installed during the normal Master Data Engine installation process. The directory server can be configured to communicate with external corporate directory servers.

Installing and configuring your Master Data Engine is the first step in implementing the IBM Initiate Master Data Service. The installation process involves pre-installation configuration of your operating system and database platforms, installation of the engine, and various configuration steps.

Chapter 2. Installation worksheets

After the Master Data Engine is installed, you must create an instance of the engine for each runtime environment you plan to use. Depending on your implementation, you might also need instances for LDAP Directory Server, entity managers, event managers, and data source. Completing the installation worksheets before you install the Master Data Engine helps you in planning your installation, as well as saving time and enforcing consistency during installation and instance creation.

Reuse the worksheets for each runtime environment that you plan to implement. For example, you might have a production environment, a test environment, and a training environment.

You use the madconfig utility to create instances. The data source, engine, LDAP, entity manager, and even notification worksheets are organized in the order in which the madconfig utility create_instance prompts are displayed.

Related concept

Chapter 3, “Planning your Master Data Engine installation,” on page 25

Chapter 4, “Preparing your environment,” on page 31

Related reference

“madconfig utility” on page 102

Master Data Engine database connections worksheet

The database connection count is the sum of the connections that are used by the Master Data Engine and by any entity managers. Use the database connection worksheet to estimate the number of database connections required for your installation.

When planning the number of connections required, remember that some Master Data Engine or IBM Initiate Workbench processes require additional database connections. Also, remember that LDAP processes require connections intermittently. Plan to include some additional connections for these processes.

Completing this database connection worksheet before beginning the installation can help you when responding to installation prompts.

Table 1. Database connection worksheet

Connection used by	Guidance	Your value
Engine	The number of JDBC connections, as specified in the jdbc.properties maxActive value.	
Engine	The number of ODBC connections as specified in the com.initiatesystems.hub.mpinet.threads value.	

Table 1. Database connection worksheet (continued)

Connection used by	Guidance	Your value
Engine	The total number of callback threads for all callback handlers, as specified in the com.initiatesystems.hub.handler.cbthreads value. The database connections are used only when a callback handler has been registered and deployed. Callback threads for undeployed callback handlers do not create database connections.	
Engine	Two ODBC connections for the internal administrative processes (dictionary refresh and sequence generation).	2
Engine	If you are using an embedded Entity Manager, have one connection for each entity type managed by the Entity Manager.	
Stand-alone Entity Manager	If you are using stand-alone Entity Managers, have three connections for each stand-alone entity manager.	
Other processes	Include additional connections as needed for IBM Initiate Workbench, Master Data Engine, and LDAP processes. There is no specific value that can be suggested, as requirements vary from installation to installation. However, you might specify a value that represents 10-25 percent of the sum of the connections from the preceding worksheet rows.	
Total number of connections	Enter the sum of your connections here. This number is an approximate count of the database connections your installation requires.	

Data source worksheet

Use the data source worksheet to identify parameters for the data source to which your Master Data Engine instance is connecting.

You use the madconfig utility to create instances. This worksheet is organized in the order in which the madconfig utility create_instance prompts are displayed.

Table 2. Engine data source worksheet

Configuration	Guidance	Your value
Data source name	Identify a name for the data source configuration. The engine instance uses the data source to communicate with the database. The name must consist of 12 or fewer alphanumeric characters. Underscore (_) characters can be used in the name, for example `prod_100`. Other characters are not supported.	

Table 2. Engine data source worksheet (continued)

Configuration	Guidance	Your value
Database type	Identify the database type. The options are: `db2`, `mssqlu`, and `oracle`. The ODBC driver applied by the Master Data Engine installer is determined by the database type you define. The wire driver enables the engine instance to communicate with the database and write data to the schema. However, the engine host requires installation of the applicable database client to enable the engine instance to perform bulk load operations. The Master Data Engine software includes the ODBC drivers listed here. Others are not supported. • Oracle Wire (`oracle` option) • Oracle Net (`oracle` option) • IBM DB2® Wire (`db2` option) • SQL Server Wire (`mssqlu` option). For Oracle databases, the Master Data Engine determines whether to install the wire or net driver based on the database host and port values. If you apply empty values for the database host and port, the engine installs the Oracle Net driver, which requires installation of the Oracle client on the engine host. IBM AIX®, Linux, or Solaris data source information is stored in an `odbc.ini` file located in the `MAD_ROOTDIR/conf` directory.	
Database server name	Identify the fully qualified address of the host on which the database is installed, for example: *prod.customer.com*. For IBM DB2 or Oracle, provide the database host and database port.	
Database name	Identify the name of the database instance created for the engine, for example: `prod`. The database name and data source name can be the same. For Oracle databases, provide the database system ID (SID).	
Microsoft Windows authentication	For the credentials that the Master Data Engine uses to authenticate to the SQL Server database, choose whether to use Microsoft Windows credentials or SQL Server credentials. Choose `y` to use Microsoft Windows authentication. The database authentication mechanism does not prompt for user name and password, but uses the Microsoft Windows credentials. Choose `n` to use SQL Server credentials. You are prompted for the SQL Server credentials provided during the madconfig utility create_instance process.	

Related reference

"madconfig utility" on page 102

Related tasks

"Creating a data source" on page 45

Master Data Engine installation worksheet

Use the engine installation worksheet to define the root directory (MAD_ROOTDIR) values for the host intended for the initial Master Data Engine runtime environment.

Use the engine installation worksheet to identify the directory in which you are installing the Master Data Engine software. The installer provides a default location, but you can use the directory structure of your choice.

If you install additional runtime environments later, they might not point to the same database as that of the initial environment. If installing multiple runtime environments, re-use the installation worksheet to define the unique directory values for each environment.

Table 3. Engine installation worksheet

Configuration	Guidance	Your value
Installation home directory	Identify the MAD_ROOTDIR, which is the path name for the directory in which to install the Master Data Engine. Use this example: **Microsoft Windows** default: C:\Program Files\IBM\Initiate\Engine10.0.0.*x* **IBM AIX, Linux, or Solaris** default: /opt/IBM/Initiate/Engine10.0.0.*x*	

Related concepts

"Master Data Engine directory structure - MAD_ROOTDIR and MAD_HOMEDIR" on page 32

"Master Data Engine elements and high-level interdependencies" on page 26

"Master Data Engine and instances installed on different drives (Microsoft Windows)" on page 39

Related tasks

Chapter 5, "Installing the Master Data Engine," on page 41

Master Data Engine instance worksheet

Use the Master Data Engine instance worksheet to define your instance values to save time when you respond to instance creation prompts.

Use the engine instance worksheet to identify the parameters for each engine instance you plan to create.

You use the madconfig utility to create instances. This worksheet is organized in the order in which the madconfig create_instance prompts are displayed.

If you install additional runtime environments later, they might not point to the same database as that of the initial environment. If installing multiple runtime environments, re-use the installation worksheets to define the unique values for each environment.

Table 4. Engine instance worksheet

Configuration	Description	Your value
Instance name	Identify a name for the engine instance, for example, `prod100_1` . The product supports creating and running multiple engine instances within a single home directory.	
Instance home directory	Identify the path to the directory (`MAD_HOMEDIR`) in which to create the engine instance. One example is: **Microsoft Windows** default: `C:\group\name` **IBM AIX, Linux, or Solaris** default `/opt/group/name` where *group* is the name of the directory created for the associated runtime instances (for example, `prod` or `qa`), and *name* is the instance name specified. The product supports creating and running multiple engine instances within a single home directory.	
SSL enablement or FIPS compliance	If your engine communicates over SSL, or is being installed in a U.S. government environment (FIPS-compliant), identify the: • Full path and name of the JSSE truststore (the madconfig utility provides a default) and password for accessing the JSSE truststore • Full path and name of JSSE keystore (the madconfig utility provides a default) and password for accessing the JSSE keystore • JSSE keystore type (default `PKCS12`) You are not prompted for this information unless you use the madconfig utility command documented in the enabling FIPS compliance task. **Restriction:** You cannot have a FIPS-compliant instance and a non-FIPS compliant instance that share the same engine `MAD_ROOTDIR`.	
Instance locale	Identify the primary locale for instance data. You can also identify a list of applicable secondary locales. If defining more than one locale, specify the primary locale first and then separate locale values with a comma (,). For example, you can type, `en_US` or `fr_FR` or `en_US,fr_FR`.	
Database data source name	Identify the name of the data source created for this instance. You define the name in the data source worksheet.	

Table 4. Engine instance worksheet (continued)

Configuration	Description	Your value
Database user	Identify the user name for the database user account. The database user account was created as a part of completing database configuration. This prompt is not displayed if you chose Microsoft Windows authentication for SQL Server during the madconfig utility create_datasource process.	
Database user password scheme	Identify the password scheme. Options are: • plain - no encryption • pwd2 - for passwords encrypted from the madpwd2 utility • pwd3 - for password encrypted from the madpwd3 utility To encrypt the password, you must use the madpwd2 or madpwd3 utility. This prompt is not displayed if you chose Microsoft Windows authentication for SQL Server during the madconfig utility create_datasource process.	
Database password	Identify the password for the database user account that was created as a part of completing the database configuration. If you are applying an encrypted password from the madpwd2 utility, identify the encrypted password string. Leave the value in the worksheet blank for now. You can return to this section of the worksheet after completing the remainder of the worksheet and creating the initial runtime environment. If you are applying an encrypted password from the madpwd3 utility (used for AES encryption), you must first generate the AES key and initialization vector (iv). Use the generation output here to identify the full path and name of the AES-CBC (128, 192, or 256 bit) key file and iv file in hex-encoded format. Return to this section of the worksheet to identify the encrypted string after completing the rest of this worksheet and creating the initial runtime environment. This prompt is not displayed if you chose Microsoft Windows authentication for SQL Server during the madconfig create_datasource process.	
Database host	Identify the fully qualified address of the host on which the database is installed, for example, *prod.customer.com* . The madconfig utility pre-populates the corresponding value from the specified data source. Press Enter to apply the default value.	
Database port	Identify the database port number. The default values are: • **Oracle**: `1521` • **SQL Server**: `1433` • **IBM DB2**: `50000`	

Table 4. Engine instance worksheet (continued)

Configuration	Description	Your value
MPINET over TCP/IP	Identify whether the Master Data Engine instance provides MPINET services (standard socketing). Answering `y` to this prompt results in standard MPINET TCP/IP communication behavior, meaning that the engine listens for the standard MPINET protocol over port 16000. If your implementation requires MPINNET over HTTP, respond `n` and see the MPINET over HTTP prompt in the next row.	
Engine instance host name	Identify the fully qualified address of this host that is intended for this instance of the Master Data Engine, for example, *myhub.customer.com*.	
Engine instance port number	Identify the port number intended for communication with the Master Data Engine. The default value is: `16000` In a multi-engine environment, each Master Data Engine (MPINET port) must listen on a different port (for example, MPINET1 on 16000, MPINET2 on 16001, and so on).	
MPINET over HTTP	Identify whether this engine instance provides MPINET over HTTP services. You can configure an instance to run over both TCP/IP and HTTP. In some government environments, this method is required. See the *IBM Initiate Master Data Service Security Technical Implementation Guide (STIG)* for details.	
Embedded application services	Identify whether the engine instance provides embedded application services (IBM Initiate Enterprise Service Oriented Architecture (ESOA) Toolkit RESTful services).	
Embedded application services port number	If using the embedded application services, indicate the web server port number. The default value is: `7378`.	
Full-text index	Identify whether the instance is to use a full-text index. A response of `y` enables IBM Initiate Flexible Search.	
Engine instance thread count	Identify the number of concurrent connections (context pool objects) the instance can handle. The default value is: 1.	
Embedded entity manager	Identify whether the engine instance is to use the embedded entity manager. The valid options are: • `y` to use the embedded entity manager, which is typical for most implementations. • `n` to not use the embedded entity manager. The embedded entity manager runs within the engine instance. Another type, a stand-alone entity manager, runs as a separate instance to manage a single entity type.	
Embedded event manager	If you plan to implement event notification, enter `y` to use the embedded event manager. Event notification allows external systems to receive messages for a subset of internal events that are generated within the IBM Initiate Master Data Service.	

Table 4. Engine instance worksheet (continued)

Configuration	Description	Your value
Embedded LDAP directory server	Identify whether to embed the required IBM Initiate LDAP directory server. The valid values are: • `y` to embed within the engine instance • `n` to not embed within the engine instance For an engine instance to start, it requires an IBM Initiate LDAP directory server. You can use an embedded server, use a server that is embedded in a different engine instance, or create a stand-alone server instance. If you are not embedding the IBM Initiate LDAP directory server within this engine instance, you must specify within the corresponding configuration prompts to use a stand-alone IBM Initiate LDAP directory server or to use the embedded IBM Initiate LDAP directory server in another Master Data Engine runtime environment. If the applicable stand-alone or embedded server does not exist yet, return here to specify the corresponding values after creating that server. In addition, your implementation might require integration with an external corporate LDAP directory server.	
LDAP directory server host	Identify the fully qualified host name for the IBM Initiate LDAP directory server, for example, *initiateldap.customer.com*. This prompt is not displayed if you chose `y` at the previous prompt in order to embed an LDAP directory server.	
LDAP directory server port number	Identify the port number for the IBM Initiate LDAP directory server. The default value is: `1389`	
LDAP server admin port number	Identify the LDAP server administration port number. The replication technology used in the embedded and stand-alone LDAP directories is specific to the vendor used (OpenDS) and cannot be used with other directory server implementations. OpenDS requires this port number when performing administrator tasks on the LDAP server.	
LDAP directory server in a cluster with existing LDAP servers	Identify whether the IBM Initiate LDAP directory server is intended for membership in a cluster with existing LDAP servers. The valid values are: `y` for yes or `n` for no. If you enter `y`, you are prompted for additional information such as the IBM Initiate LDAP Server replication port number and the cluster peer IBM Initiate LDAP Server host, port number, and replication port number. Enter `y` only after you create at least one embedded or stand-alone IBM Initiate LDAP instance. This prompt is not displayed if you chose not to embed an IBM Initiate Directory Server.	

Table 4. Engine instance worksheet (continued)

Configuration	Description	Your value
Embedded task manager	Identify whether the engine instance uses an embedded task manager (process manager). The engine task manager is a method by which the engine polls certain events or processes and runs the associated logic on a scheduled basis. If you plan to implement dynamic frequency-based bucketing (DFBB), you must enable the embedded task manager. DFBB identifies frequently occurring bucket values at runtime Do not confuse the use of the term "task manager" with the concept of identity, relationship, or custom tasks that are worked by data stewards in applications like IBM Initiate® Inspector.	
Task manager polling interval	If you use the task manager, indicate the interval (in seconds) that the task manager polls for events or processes to run. The default is 60 seconds.	
Task manager poll for expired tasks	Identify whether the task manager polls for expired tasks. Custom tasks (tasks that are worked by data stewards) can be configured to expire in the hub if the custom task has not been worked or resolved within a defined number of days.	
Task expiration polling interval	Indicate, in seconds, the interval in which the task manager polls for expired tasks. The default is 60 seconds.	
Task status for expiring tasks	Identify the task status for expired tasks. The default is a status of 6.	
Frequency string poll	Identify if the task manager polls for frequency strings. If enabled, the task manager looks for newly added frequency strings and re-buckets those members that might be affected. This functionality is required for DFBB.	
Frequency string polling interval	If the task manager polls for frequency strings, indicate the polling interval in seconds. The default is 60 seconds. This functionality is used by DFBB.	
Frequency string maudrecno	Identify the maudrecno used to identify a new frequency string. Entering a value of 0 processes any frequency strings added since the startup of this engine instance. The default is 0. This functionality is used by DFBB.	
Master Data Engine queue poll interval	If you are implementing event notification, specify the number of seconds in which you want the event manager to poll the engine for event notifications. The default is 10 seconds.	
Master Data Engine queue work unit	If you are implementing event notification, specify the row count that is pulled from the entoque record in the engine database. The default is 500.	
Master Data Engine queue worker thread count	If you are implementing event notification, specify the number of threads used to process notifications in parallel. The default is 1.	

Table 4. Engine instance worksheet (continued)

Configuration	Description	Your value
Enable entity manager relationship linker	Identify whether to enable the relationship linker. If you are managing relationships, like households, enter `y`.	
Entity manager database poll interval	Enter the number of seconds that the queue manager waits to requery when no records are in the database, the internal queue is full, or if a database error has reached the maximum number of retry attempts. The default is 10 seconds.	
Entity manager database work unit	Enter the number of entique records the queue manager retrieves from the database per database query. The default is 500. An entique record is created when the engine processes new members or modifications to existing members.	
Entity manager worker thread count	Enter the number of threads that process entique records in parallel per entity type. The default is 1.	
Engine management port number	Identify the port number used by the JMX browser to monitoring the Master Data Engine from a remote system. The default value is: `1199`. For more information about the JMX browser, see the "Operational monitoring" chapter within the *IBM Initiate Workbench User's Guide*.	
Callback thread count	Identify the callback thread count. The default value is: `2` If callback handler functionality (callouts to a vendor application) is not applicable, apply the default value. If you are planning to use callouts, consider applying a higher value depending on the amount of callback activity expected. **Callback handlers**. As needed, you can create custom callback handlers by using the IBM Initiate Java or .NET SDKs. For development details, see the *IBM Initiate Master Data Service SDK Reference for Java and Web Services*. Also, after creating a custom handler, you must: • Set the `MAD_CALLBACKLIB` variable. • Register the handler in IBM Initiate Workbench (see *IBM Initiate Workbench User's Guide*). • Deploy the handler by using the madconfig utility.	

Table 4. Engine instance worksheet (continued)

Configuration	Description	Your value
Bootstrap the database	Identify whether to have the instance creation process bootstrap the database. The valid values are: • `y` to load the data model into the database. **Attention:** Applying `y` overwrites all data in the database with no way to recover. Do not apply `y` unless setting up an initial runtime environment that points to an empty database instance. • `n` if you do not want to load the data model or if you want to delay loading it. For example, do not bootstrap (select `n`) when creating an instance to point to an existing database for upgrade or an additional instance that points to an already-bootstrapped database. If delaying the bootstrapping process for an empty database, use the madconfig utility to load the data model at a later time: **Microsoft Windows**: `madconfig bootstrap_instance` **IBM AIX, Linux, or Solaris**: `madconfig.sh bootstrap_instance`	

Related concepts

"Master Data Engine directory structure - MAD_ROOTDIR and MAD_HOMEDIR" on page 32

"Master Data Engine elements and high-level interdependencies" on page 26

"Master Data Engine and instances installed on different drives (Microsoft Windows)" on page 39

Appendix K, "FIPS compliance," on page 255

"Master Data Engine database configuration" on page 36

"System and software users for the Master Data Engine" on page 36

"Database user account password encryption" on page 43

"Master Data Engine installation in a high-availability environment" on page 27

Appendix D, "Thread count settings," on page 219

Chapter 8, "Entity managers," on page 73

Chapter 9, "Event notification," on page 81

Appendix A, "LDAP Directory Server for the Master Data Engine," on page 197

"IBM Initiate LDAP directory server stand-alone instance" on page 29

"Master Data Engine environment variables" on page 87

Related reference

"madconfig utility" on page 102

"Data source worksheet" on page 4

"LDAP directory server worksheet"

"Stand-alone entity manager worksheet" on page 16

Related tasks

"Enabling FIPS compliance in the Master Data Engine" on page 255

LDAP directory server worksheet

Before creating the stand-alone instance of the LDAP directory server, use the LDAP directory server worksheet to define the required values.

You use the madconfig utility to create instances. This worksheet is organized in the order in which the madconfig utility create_instance prompts are displayed.

Table 5. LDAP directory server worksheet

Configuration	Description	Your value
Instance name	Identify the name for the stand-alone instance of the LDAP directory server. For example: `prod100`	
Instance home directory	Identify the path to the directory in which to create the specified server instance. One example is: **Microsoft Windows**: `C:\home\group` **IBM AIX, Linux, or Solaris**: `/home/group` where *group* is the name of the directory created for the associated runtime instances (for example: `prod` or `qa`).	
Instance locale	Identify the primary locale for instance data. You can also identify a list of applicable secondary locales. If defining more than one locale, specify the primary locale first and separate locale values with a comma (,). For example, you can type `en_US` or `fr_FR` or `en_US,fr_FR`.	
Database data source name	Identify the name of the data source created for this LDAP directory server instance. You define the name in the data source worksheet.	
Database type	Identify the type of database for the specified data source.	
Database user	Identify the user name for the database account. The database user account was created as a part of completing the database configuration. This prompt is not displayed if you chose Microsoft Windows authentication for SQL Server during the madconfig utility create_datasource process.	

Table 5. LDAP directory server worksheet (continued)

Configuration	Description	Your value
Encrypted format for database user password	Identify whether to specify the password in encrypted format. The valid values are: `y` for yes or `n` for no. If `y`, the password must have been encrypted and you must know its encrypted value . This prompt is not displayed if you chose Microsoft Windows authentication for SQL Server during the madconfig utility create_datasource process.	
Database password	Identify the password for the database user account. If the password is encrypted and you opted to apply an encrypted value, define the encrypted password. This prompt is not displayed if you chose Microsoft Windows authentication for SQL Server during the madconfig utility create_datasource process.	
Database host	Identify the fully qualified address of the host on which the database is installed. For example: *prod.customer.com* The madconfig utility prepopulates the corresponding value from the specified data source. To apply the default value, press Enter.	
Database port	Identify the database port number.	
LDAP server host	Identify the fully qualified host name for this server instance. For example: *initiateldap.customer.com*	
LDAP server port number	Identify the port number for this server instance. The default value is: `1389`	
LDAP server in a cluster with existing LDAP servers	Identify whether this server instance is intended for membership in a cluster with existing LDAP servers.	
Windows service account	Identify the account name for the Microsoft Windows service to use when connecting. This prompt is not displayed unless you chose Microsoft Windows authentication during the madconfig utility create_datasource process.	
Windows service password	Identify the password for the Microsoft Windows service to use when connecting. This prompt is not displayed unless you chose Microsoft Windows authentication during the madconfig utility create_datasource process.	
LDAP server management port number	Identify the management port number for the server. The default value is: `1198`	

Related concepts

Appendix A, “LDAP Directory Server for the Master Data Engine,” on page 197

“Master Data Engine directory structure - MAD_ROOTDIR and MAD_HOMEDIR” on page 32

"Master Data Engine database configuration" on page 36

"System and software users for the Master Data Engine" on page 36

"Database user account password encryption" on page 43

Related reference

"Master Data Engine instance worksheet" on page 6

"madconfig utility" on page 102

Related task

"Creating a stand-alone IBM Initiate LDAP directory server instance" on page 46

Stand-alone entity manager worksheet

To configure a stand-alone entity manager, you create an instance that manages a single entity type or one that uses more threads than another. Use the stand-alone entity manager worksheet to record information for each stand-alone entity manager that you plan to configure.

You use the madconfig utility to create instances. This worksheet is organized in the order in which the madconfig utility create_instance prompts are displayed. Entity configuration parameters are stored in the `com.initiate.server.queue.cfg` file.

Table 6. Stand-alone entity manager worksheet

Configuration	Description	Your value
Entity manager instance name	Identify a name for the new entity manager instance. For example, you might name it `prod100_id` if the entity type is identity. The example name is based on the suggested syntax of combining the group and version along with the entity identifier (*groupvers*[*_identifier*])	
Instance home directory	Identify the path to the directory in which to create the specified entity manager. One example is: **Microsoft Windows**: `C:\ibm\initiate\home\`*group* **IBM AIX, Linux, or Solaris**: `/ibm/initiate/home/`*group* where *group* is the name of the directory created for the associated runtime instances (for example, `prod` or `qa`).	
Instance locale(s)	Identify the locale for the entity manager. If more than one locale applies, specify the primary locale first, and separate locale values with a comma (,). For example, you can type `en_US, fr_FR`	

Table 6. Stand-alone entity manager worksheet (continued)

Configuration	Description	Your value
Entity type	Identify the entity type that this entity manager is to manage (for example, `identity`). The entity type must be configured for asynchronous processing through IBM Initiate Workbench. To identify the current entity types defined, query the `mpi_enttype` table in the database. Also, you can view the **Entity Types** tab from the **Member Types** tab in IBM Initiate Workbench. Asynchronous management is further discussed in the entity manager topics and in the "Algorithms" and "Configuration editor" chapters in the *IBM Initiate Workbench User's Guide*.	
Database data source name	Identify the name of the database data source this entity manager is to use.	
Database user	Identify the user name for the database account. This prompt is not displayed if you chose Microsoft Windows authentication during the madconfig utility create_datasource process.	
Encrypted format for database user password	Identify whether to specify the password in encrypted format. If `y`, the password must be encrypted according to and you must know the encrypted value. This prompt is not displayed if you chose Microsoft Windows authentication during the madconfig utility create_datasource process.	
Database password	Identify the password for the database user account. If password is encrypted and you opted to apply an encrypted value, define the encrypted password. This prompt is not displayed if you chose Microsoft Windows authentication during the madconfig utility create_datasource process.	
Database host	Identify the fully qualified address of the host on which the database is installed. For example: *prod.customer.com* The madconfig utility pre-populates the corresponding value from the specified data source. Press Enter to apply the default value.	
Database port	Identify the database port number.	
LDAP directory server host	Identify the fully qualified host name for the IBM Initiate LDAP directory server. For example: *initiateldap.customer.com*	
LDAP directory server port number	Identify the port number for the IBM Initiate LDAP directory server.	

Table 6. Stand-alone entity manager worksheet (continued)

Configuration	Description	Your value
LDAP server admin port number	Identify the LDAP server administration port number. The replication technology used in the embedded and stand-alone LDAP directories is specific to the vendor used (OpenDS) and cannot be used with other directory server implementations. OpenDS requires this port number when performing administrator tasks on the LDAP server.	
Enable relationship linker	Identify whether to enable the relationship linker for this entity manager.	
Entity manager database poll interval	Enter the number of seconds that the queue manager waits to re-query when no records are in the database, the internal queue is full, or if a database error has reached the maximum number of retry attempts. The default is 10 seconds.	
Entity manager database work unit	Enter the number of entique records the queue manager retrieves from the database per database query. The default is 500. An entique record is created when the engine processes new members or modifications to existing members.	
Entity manager worker thread count	Enter the number of threads that process entique records in parallel per entity type. The default is 1.	
Entity manager management port number	Identify the port number for this entity manager. The default value is: `1200`	
Callback thread count	Identify the callback thread count for this entity manager. The default value is: `2` If callback handler functionality (callouts to a vendor application) is not applicable, apply the default value. If you are planning to use callouts, consider applying a higher value depending on the amount of callback activity expected.	
Microsoft Windows service account	Identify the account name for the Microsoft Windows service to use when connecting. This prompt is not displayed unless you chose Microsoft Windows authentication for SQL Server during the madconfig utility create_datasource process.	
Microsoft Windows service password	Identify the password for the Microsoft Windows service to use when connecting. This prompt is not displayed unless you chose Microsoft Windows authentication for SQL Server during the madconfig utility create_datasource process.	

Related concepts

"Master Data Engine directory structure - MAD_ROOTDIR and MAD_HOMEDIR" on page 32

Chapter 8, “Entity managers,” on page 73

“Master Data Engine database configuration” on page 36

“Database user account password encryption” on page 43

Related reference

“Master Data Engine instance worksheet” on page 6

“Entity manager configuration parameters” on page 76

“madconfig utility” on page 102

Related task

“Creating stand-alone entity managers” on page 46

Event notification worksheets

Use the event notification worksheet to define the values for your event manager and event handler.

You can embed your event manager in your Master Data Engine instance or you can configure a stand-alone manager.

- Embedded event manager - complete the embedded event manager worksheet and the embedded event handler worksheet.
- Stand-alone event manager - complete the stand-alone event manager worksheet and the stand-alone event handler worksheet.

The madconfig utility is used to create event manager instances and event handlers. These worksheets are organized in the order in which the madconfig utility prompts are displayed.

The values defined in the embedded event manager worksheet are the same event notification values defined in the engine instance worksheet.

Table 7. Embedded event manager worksheet

Configuration	Guidance	Your value
Embedded event manager	If you plan to implement event notification, enter y to use the embedded event manager. Event notification allows external systems to receive messages for a subset of internal events that are generated within the IBM Initiate Master Data Service.	
Master Data Engine queue poll interval	If you are implementing event notification, specify the number of seconds in which you want the event manager to poll the engine for event notifications. The default is 10 seconds.	
Master Data Engine queue work unit	If you are implementing event notification, specify the row count that is pulled from the entoque record in the engine database. The default is 500.	
Master Data Engine queue worker thread count	If you are implementing event notification, specify the number of threads used to process notifications in parallel. The default is 1.	

Use the embedded event handler configuration worksheet to define the values for your embedded event handler configuration. Use the madconfig utility configure_instance_event_handler command.

Table 8. Embedded event handler configuration worksheet

Configuration	Guidance	Your value
Master Data Engine instance	Identify the name of your engine instance.	
Event Manager publishing destination	Specify the destination queue name. This entry must be the same name specified in IBM Initiate Workbench.	
Event Manager publishing destination user	Specify the destination user name.	
Password scheme	Identify the password scheme. Options are: • plain - no encryption • pwd2 - for passwords encrypted from the madpwd2 utility • pwd3 - for password encrypted from the madpwd3 utility To encrypt the password, you must use the madpwd2 or madpwd3 utility.	
Password	Identify the password for the publishing. If the password is encrypted and you opted to apply an encrypted value, define the encrypted password.	
Add another publishing destination	Indicate whether you plan to send notifications to multiple destinations. If yes, you are prompted to enter the necessary information for the new destination.	

Use the stand-alone event manager worksheet to define values for a stand-alone event manager. Use the madconfig utility create_eventmgr command.

Table 9. Stand-alone event manager worksheet

Configuration	Guidance	Your value
Event Manager instance name	Identify a name for the instance.	
Event Manager instance home directory	Identify the path to the directory in which to create the event manager instance.	
Engine instance locale	Identify the locale for the engine instance associated with this event manager. If more than one locale applies, specify the primary locale first, and separate locale values with a comma (,). For example, you can type `en_US, fr_FR`.	

Table 9. Stand-alone event manager worksheet (continued)

Configuration	Guidance	Your value
Entity type	Identify the entity type for this event manager (for example, `id` for identity entity). To identify the entity types defined for your implementation, query the mpi_enttype table in the database. Also, you can view the Entity Types tab from the Member Types tab in IBM Initiate Workbench.	
Database data source name	Identify the name of the database data source this event manager is to use.	
Database password scheme	Identify the password scheme. Options are: • plain - no encryption • pwd2 - for passwords encrypted from the madpwd2 utility • pwd3 - for password encrypted from the madpwd3 utility To encrypt the password, you must use the madpwd2 or madpwd3 utility.	
Database user	Identify the user name for the database user account. The database user account was created as a part of completing database configuration. This prompt is not displayed if you chose Microsoft Windows authentication for SQL Server during the madconfig utility create_datasource process.	
Database password	Identify the password for the database user account. If the password is encrypted and you opted to apply an encrypted value, define the encrypted password.	
LDAP directory server host	Identify the fully qualified host name for the LDAP directory server, for example, `initiateldap.customer.com`.	
LDAP directory server port number	Identify the port number for the LDAP directory server.	
LDAP server admin port number	Identify the LDAP server administration port number. The replication technology used in the embedded and stand-alone LDAP directories is specific to the vendor used (OpenDS) and cannot be used with other directory server implementations. OpenDS requires this port number when performing administrator tasks on the LDAP server.	
Event Manager queue poll interval	Specify the number of seconds in which you want the event manager to poll the engine for event notifications. The default is 10 seconds.	
Event Manager queue work unit	Specify the row count that is pulled from the entoque record in the Master Data Engine database. The default is 500.	
Event Manager queue worker thread count	Specify the number of threads used to process notifications in parallel. The default is 1.	

Table 9. Stand-alone event manager worksheet (continued)

Configuration	Guidance	Your value
Event Manager management port number	Identify the port number for this event manager. The default value is: 1201.	

Use the stand-alone event handler worksheet to define the values for your event handler. You can configure multiple event handlers to publish notifications to multiple destinations. Re-use this worksheet for each destination required. Use the madconfig utility configure_eventmgr_eventhandler command.

Table 10. Stand-alone event handler worksheet

Configuration	Guidance	Your value
Event Manager instance name	The name of event manager specified in the stand-alone event manager worksheet.	
Event Manager JMS connection factory name	The JMS connection factory name is defined in your JMS queue provider and JNDI LDAP environment. A ConnectionFactory object encapsulates a set of connection configuration parameters that are defined by an administrator. A client uses it to create a connection with a JMS provider. A ConnectionFactory object is a JMS administered object and supports concurrent use. The JMS API establishes the convention that JMS clients find administered objects (for example, Queues) by looking them up in a JNDI namespace.	
Event Manager publishing destination name	Specify the destination queue name. This entry must be the same name specified in IBM Initiate Workbench.	
Event Manager publishing destination user	Specify the destination user name.	
Event Manager publishing destination	Specify the destination queue name. This entry must be the same name specified in IBM Initiate Workbench.	
Publishing destination password scheme	Identify the password scheme. Options are: • plain - no encryption • pwd2 - for passwords encrypted from the madpwd2 utility • pwd3 - for password encrypted from the madpwd3 utility To encrypt the password, you must use the madpwd2 or madpwd3 utility.	
Publishing destination password	Identify the password for the publishing. If the password is encrypted and you opted to apply an encrypted value, define the encrypted password.	

Table 10. Stand-alone event handler worksheet (continued)

Configuration	Guidance	Your value
Add another publishing destination	Indicate whether you plan to send notifications to multiple destinations. If yes, you are prompted to enter the necessary information for the new destination.	

Related concept

Chapter 9, “Event notification,” on page 81

Related reference

“madconfig utility” on page 102

“madpwd2 utility” on page 128“madpwd3 utility” on page 128“Master Data Engine instance worksheet” on page 6

Chapter 3. Planning your Master Data Engine installation

Before you install the Master Data Engine, review theses checklist items and determine how to address those items in your implementation.

Use this checklist as a guide through the planning process. Links to related topics are provided at the end.

1. Review the Master Data Engine elements and high-level interdependencies topic.
2. Identify any high-availability needs. While there are no specific steps you need to take during installation, it is suggested that you review the high-availability topic.
3. If you are installing in a U.S. federal government environment, review the "Master Data Engine installation in U.S. government environments" and "FIPS compliance" topics. You provide information about this item in the engine instance worksheet.
4. Plan your required instances by referring to the Master Data Engine instance worksheet.
5. Review the LDAP directory structure topics and determine whether your implementation will use an embedded LDAP server, a stand-alone LDAP server, an external corporate LDAP server, or a combination of options. If you are using an embedded server, note this in the Master Data Engine instance worksheet. If you are using an embedded or external server, complete the stand-alone LDAP server worksheet.
6. Review the entity management topics and determine your strategy. If you are using an embedded entity manager, note this on the Master Data Engine instance worksheet. If you are using a stand-alone entity manager, complete the stand-alone entity manager worksheet.
7. Review the event notification topics and determine whether your implementation requires this feature. If so, you can opt to use an embedded event manager or a stand-alone event manager. If you are using an embedded event manager instance, complete the engine instance worksheet and the embedded event manager and embedded event handler configuration worksheets (these two worksheets are found in the event notification worksheets topic). If you are using a stand-alone event manager, complete the stand-alone event manager and stand-alone event handler worksheets (also found in the event notification worksheets topic).
8. Complete all other installation worksheets.

Important: In versions prior to version 7.5 of the software, each Master Data Engine instance created was appended to an `.ini` configuration file (for example, `services.ini` or `madman.ini`). If upgrading from version 7.5 or earlier, you must store a backup of the existing `.ini` file. While this file is no longer used by the Master Data Engine, it is required for the Message Broker Suite components (see the *IBM Initiate Master Data Service Message Broker Suite Reference*).

Topics in the *IBM Initiate Master Data Service Engine Installation Guide* are intended for individuals responsible for installing and configuring the core Master Data Engine. Specific operational procedures, such as starting, stopping, and monitoring software, and reprocessing messages, are provided in the *IBM Initiate Master Data*

Service Software Operations Guide. Information about configuring a hub is provided in the *IBM Initiate Workbench User's Guide*.

Related concepts

Appendix A, "LDAP Directory Server for the Master Data Engine," on page 197

Appendix K, "FIPS compliance," on page 255

Chapter 8, "Entity managers," on page 73

Chapter 9, "Event notification," on page 81

"SSL security" on page 191

Chapter 4, "Preparing your environment," on page 31

Master Data Engine elements and high-level interdependencies

The Master Data Engine is the core component of the IBM Initiate Master Data Service. Before beginning the installation of the Master Data Engine, it is helpful to understand the engine elements and their interdependencies.

Engine database. The engine operating data is stored in the engine database. You bootstrap the database by using the madconfig utility. The bootstrapping process creates the core tables and indexes and loads a baseline dictionary for starting the engine instances that are associated with the database.

Master Data Engine runtime environment. At a minimum, the runtime environment includes an engine installation, data source configuration, and an engine instance. The runtime environment communicates with the same engine database, which can be located on a separate host (server). Depending on the implementation, the runtime environment can also include additional engine instances and one or more other runtime instance types. For example, you can have stand-alone entity managers and LDAP directory servers.

- **Engine installation**. The Master Data Engine software is the infrastructure that provides the core logic and functionality of the IBM Initiate Master Data Service. The engine compares member records and scores that data based on the algorithms and other deployment-specific logic configuration.

  You use an operating system-specific installer to create the engine through a graphical user interface (GUI) or command-line interface (CLI) process. The installer creates the engine in the directory that you specify.
- **Engine instance**. On the engine host, you create one or more engine instances. Each running instance of the Master Data Engine uses various utilities from the engine installation directory to complete requests sent by client applications. For example, the madconfig utility in the engine installation `\scripts` directory is used to create engine instances, data source configurations, and stand-alone instances of entity managers and LDAP directory servers.

  Each instance has a home directory and an instance directory. When running the madconfig utility, you specify a home directory for the engine instance. This directory is where the processes of a created runtime instance are located.
- **Engine data source**. The engine instance and other types of runtime instances depend on a data source configuration to communicate with and write to the database. All supported OBDC drivers for the specified database type are

automatically installed when you create the data source. As with creating engine instances, the madconfig utility is used to create the data source.

Each engine host requires a data source configuration. In creating subsequent engine instances on the same host, you can use the same data source (that is, avoid creating additional data sources) if the instances use the same database. If connecting to different databases, you must create a corresponding data source for each database (for example, one for the `prod` database and one for the `qa` database).

- **Entity management**. The entity manager is the logic within the Master Data Engine that controls when comparison takes place after member data is derived. You can configure your engine to use an embedded entity manager or a stand-alone entity manager.
- **LDAP directory server**. All authentication to the Master Data Engine is done through a bundled LDAP directory server which is installed during the engine installation. There are multiple ways to set ups a Master Data Engine environment with the bundled LDAP directory server. You use an embedded IBM Initiate LDAP server, a stand-alone IBM Initiate LDAP server, an external LDAP server, or a combination of these options.

Related concepts

"Master Data Engine directory structure - MAD_ROOTDIR and MAD_HOMEDIR" on page 32

Chapter 8, "Entity managers," on page 73

Appendix A, "LDAP Directory Server for the Master Data Engine," on page 197

Chapter 2, "Installation worksheets," on page 3

Related reference

"Master Data Engine installation worksheet" on page 6

"Master Data Engine instance worksheet" on page 6

"madconfig utility" on page 102

Master Data Engine installation in a high-availability environment

To support installation of the IBM Initiate Master Data Service in high-availability environments, you can configure multiple Master Data Engine instances on multiple host severs. By doing so, if one server or instance goes down, the others can continue to process traffic.

No special configuration is needed at the Master Data Engine level; the Master Data Engine instances can run independently of one another.

The Master Data Engine does not provide the front-end layer that can spread transactions across the multiple engines. Transaction can be spread across engine by using either a hardware or software IP router that provides a virtual IP port for client connections to connect. In order to have a high-availability database server, IBM Initiate Master Data Service relies on the RDBMS vendors to provide a clustered database setup that provides a virtual connection for database failover.

If you are installing the IBM Initiate Master Data Service software in a vertical clustering environment (multiple Master Data Engine instances running on the same server), you must make sure that each MPINET (engine) instance is listening on a different port number. For horizontal clustering (multiple Master Data Engine instances with one instance per physical server), the ports can be the same across all servers. This figure shows the IBM Initiate® Master Data Service® software in a vertical clustering environment.

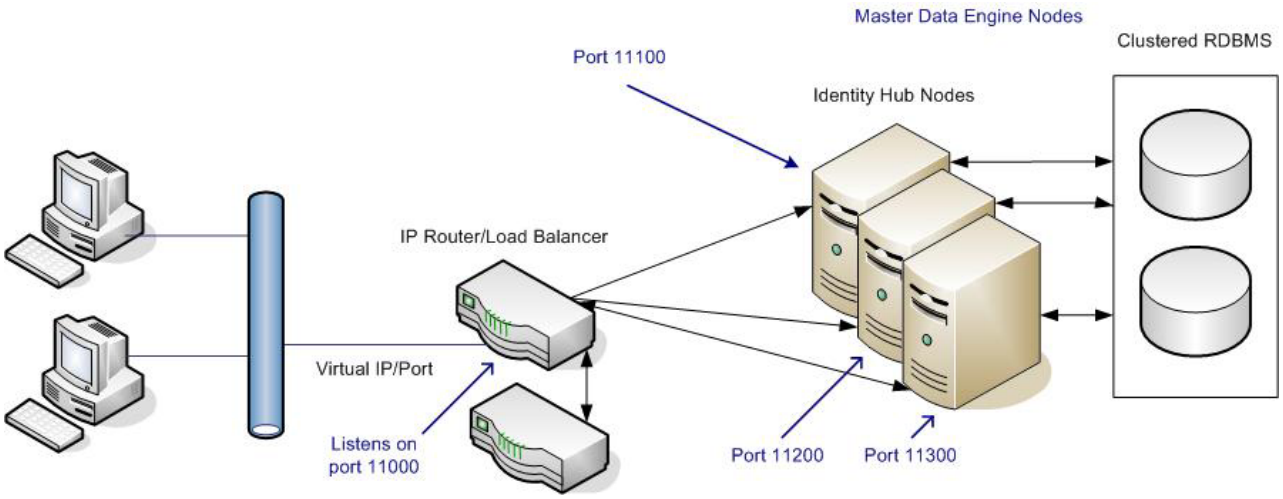

Figure 1. IBM Initiate Master Data Service vertical clustering configuration

You can run multiple instances of the Master Data Engine within a single home directory. Each instance of a Master Data Engine has its own `mpinet_name` directory, where *name* is the engine instance name. Each `mpinet_name` directory has `conf`, `data`, and `deploy` directories.

Related task

"Using ping requests to monitor Master Data Engine and database availability" on page 59

Master Data Engine installation in U.S. government environments

Many U.S. federal government environments require that certain security standards be met before the IBM Initiate Master Data Service can be used.

The Federal Information Processing Standards (FIPS) is a security standard developed by the U.S. federal government. To be FIPS-compliant, the Master Data Engine and command-line utilities that communicate over SSL must be FIPS140-2 enabled. When creating your Master Data Engine instance, you use a specific madconfig utility command that shows prompts specific to FIPS configuration. For command-line utilities, this means setting the MAD_SSLFIPSMODE variable in your `com.initiate.server.system.cfg` configuration file.

In addition, Message Broker Suite IBM Initiate Inspector, IBM Initiate Web Reports, and IBM Initiate Workbench must be FIPS-compliant before being used in a U.S. federal government implementation. For full details about installing the IBM Initiate Master Data Service and applications in a U.S. federal government environment, see *IBM Initiate Master Data Service Security Technical Implementation Guide (STIG)* .

Related concept

Appendix K, “FIPS compliance,” on page 255

Related tasks

“Enabling FIPS compliance in the Master Data Engine” on page 255

“Enabling FIPS compliance for command-line utilities” on page 257

IBM Initiate LDAP directory server stand-alone instance

All engine and entity manager instances require communication with an IBM Initiate LDAP directory server. The instances cannot start without an IBM Initiate LDAP directory server. If applicable to your environment, you can choose to implement a stand-alone instance of the LDAP directory server.

When creating an engine instance, your first option is to embed the IBM Initiate LDAP directory server in the engine instance itself. The second option is to configure the instance to use a stand-alone IBM Initiate LDAP directory server. The third option is to use an LDAP directory server that is embedded in another engine instance. If you are not embedding the IBM Initiate LDAP directory server in at least one of the engine instances created for the initial Master Data Engine environment, you must create a stand-alone instance of the server. In that case, complete the LDAP worksheet and then use the creating an IBM Initiate LDAP directory server instance procedure. Afterward, for any runtime instance that uses the stand-alone server, apply the corresponding IBM Initiate LDAP directory server configuration values.

Even if you opt to embed the IBM Initiate LDAP directory server in one or more engine instances, you might want to create a stand-alone instance for other runtime instances. Another option is to integrate runtime instances with an external corporate LDAP directory server.

Before configuring your Master Data Engine environment, review all of the LDAP directory server topics.

Related concept

Appendix A, “LDAP Directory Server for the Master Data Engine,” on page 197

Entity manager stand-alone instance

Some implementations choose to use stand-alone entity managers in order to have a tighter control over resource allocation. Others might choose to use separate entity managers for multiple entity types (for example, one for identity entities and one for household entities).

If resource allocation is a concern for your environment, you can specify more threads for a slower entity manager, and less threads for a faster entity manager.

Before configuring your Master Data Engine, review the entity manager topics.

Related concept

Chapter 8, “Entity managers,” on page 73

Event notification - stand-alone instance

Event notification allows external sources to receive messages for a subset of internal events that are generated within the IBM Initiate Master Data Service. Implementations that use event notification might choose to use stand-alone event managers in order to have a tighter control over resource allocation.

Using a stand-alone event manager allows you to scale your implementation for performance reasons. Adding additional stand-alone event managers can help to increase system performance. Before configuring your Master Data Engine, review the entity manager topics.

Related concept

Chapter 9, “Event notification,” on page 81

Chapter 4. Preparing your environment

Before you install the Master Data Engine, verify that you have completed the applicable environment preparation steps.

1. Verify that the server, or servers, on which the Master Data Engine and engine instances are being installed meet the requirements listed in *IBM Initiate Master Data Service System Requirements* and in the "Server prerequisites" topic. Also verify that all vendor software noted in the *IBM Initiate Master Data Service System Requirements* is installed on servers and workstations.
2. Review the directory structure and guideline topics. You need this information to complete the Master Data Engine installation worksheet.
3. Review the system and software users topic and identify your user and group requirements.
4. If you use the IBM AIX, Linux, or Solaris operating system, set your user limits.
5. Review the database configuration topic and complete the database connections and engine data source worksheets. Configure your database according to your database platform instructions. During the configuration process, make sure your configuration conforms to the information in the database configuration topic.

Server prerequisites

Verify that the server, or servers, on which theMaster Data Engine and engine instances are being installed meet the requirements listed in *IBM Initiate Master Data Service System Requirements* and in this topic.

The supported ODBC and JDBC drivers require that the server on which the software in being installed can connect to the relational database management system (RDBMS). For performance reasons, database client software is required to use the database vendor bulk-load tool. The use of ODBC drivers other than the drivers included with the IBM Initiate Master Data Service is not supported.

The database client drivers included and installed with the Master Data Engine are the only supported drivers.

If you are using Oracle, you must install the full Oracle Client rather than the Oracle Instant Client. The full client contains the OCI (or native) drivers that are required by the Master Data Engine.

On IBM AIX, the Master Data Engine requires the C Runtime Library for AIX, version 8.0.0.3 or later. To identify the version you have installed, use this command: `lslpp –L "xlC.aix*.rte"`.

On Red Hat Linux, you must install `compat-libstdc++-33`.

Important: In pre-7.5 versions of the software, each Master Data Engine instance created was appended to an .ini file (for example, services.ini or madman.ini). If upgrading from version 7.5 or earlier, you must store a backup of the existing .ini file. While this file is no longer in use by the Master Data Engine, it is required for the Message Broker Suite components (see the *IBM Initiate Master Data Service Message Broker Suite Reference*).

Master Data Engine directory structure - MAD_ROOTDIR and MAD_HOMEDIR

Your installation requires a directory for the installed software, and one or more directories for instances of the Master Data Engine. Although the IBM Initiate Master Data Service software does not require any specific directory location for installation, you need to plan a directory structure up front.

From one installation of the Master Data Engine, you can configure and run multiple engine instances (for example, a production instance, a test instance, and a training instance). At the end of the installation and configuration process, you have an installation directory where the software is installed and one or more instance directories from where a Master Data Engine instance runs. Because of the multiple directories, you want to choose a directory structure that meets your requirements and complies with any local network standards. The Master Data Engine uses environment variables set during installation to point to the installation and instance locations. We refer to these locations, or directories, as `MAD_ROOTDIR` and `MAD_HOMEDIR`. There are also `MAD_ROOTDIR` and `MAD_HOMEDIR`environment variables.

The `MAD_ROOTDIR` directory is where the Master Data Engine software is installed. This is the directory that you specify when you run the installer. The value of the `MAD_ROOTDIR` environment variable is the full directory path name to the installed software. This table shows examples for a host that includes two engine installations.

Table 11. Engine installation directory examples (MAD_ROOTDIR)

Engine host	MAD_ROOTDIR examples

Microsoft Windows	`C:\Program Files\IBM\Initiate\Engine10.0.x_A`
	
	`C:\Program Files\IBM\Initiate\Engine10.0.x_B`

IBM AIX, Linux, or Solaris	`/opt/IBM/Initiate/Engine10.0.x_A`
	
	`/opt/IBM/Initiate/Engine10.0.x_B`

The `MAD_HOMEDIR` directory is where the engine instances are located. A `MAD_HOMEDIR` is specified for every instance at the time the instance is created. For each engine instance, the `MAD_HOMEDIR` has an `inst` directory. The `inst` directory further contains additional directories for each instance of a Master Data Engine component (Master Data Engine, entity managers, and LDAP directory servers).

The `MAD_HOMEDIR` directory can be a single directory. The information in this table describes the directory paths for an instance home directory and an instance directory.

Table 12. Instance directory paths

Directory	Description

Instance home directory	The fully qualified directory you specify for runtime instances that communicate with the same database. For example:
	
	Microsoft Windows:
	
	`C:\MAD_HOMEDIR\group\`
	
	IBM AIX, Linux, or Solaris:
	
	`/MAD_HOMEDIR/group/`
	
	where *group* is the name of the directory created for the associated runtime instances (for example: `prod` or `qa`).

Instance directory	The fully qualified path to the directory in which the processes of a created runtime instance are located. In these examples, the instance directories are for 2 engine instances for production:
	
	Microsoft Windows:
	`C:\MAD_HOMEDIR\inst\mpinet_prod100_1`
	`C:\MAD_HOMEDIR\inst\mpinet_prod100_2`
	
	IBM AIX, Linux, or Solaris:
	`/MAD_HOMEDIR/inst/mpinet_prod100_1`
	`/MAD_HOMEDIR/inst/mpinet_prod100_2`
	
	In the directory name, the `prod100_`*identifier* string is the name that you specified for the engine instance at creation.
	
	If you use a stand-alone entity manager or LDAP directory server, those would show as:
	`/MAD_HOMEDIR/inst/mpientmgr_prod100`
	`/MAD_HOMEDIR/inst/mpildap_prod100`

The default value for the `MAD_HOMEDIR` is the name of the instance being created, but you do not have to use this convention. Although each instance must have a unique name, you can group multiple instances under a single `MAD_HOMEDIR`.

You can install the software and the instances on different physical drives if necessary. One reason some system administrators prefer this structure is because each instance has activity logs. These logs can grow quite large. The most common reason for using different physical drives is that in many non-production environments, you might have multiple copies of the software, and multiple instances. For example, you might have separate software installations and instances for developers, testers, and training. You might also have two versions of the software that you use during an upgrade or conversion project.

Directory structure guidelines for MAD_ROOTDIR (software) and MAD_HOMEDIR (instances)

Planning the structure for your `MAD_ROOTDIR` and `MAD_HOMEDIR` directories in advance can simplify your installation process.

Use these guidelines when planning the `MAD_ROOTDIR` and `MAD_HOMEDIR` directory structure on your servers. Using these guidelines can also assist you and the IBM Software Support team if you ever require their assistance.

Table 13. IBM Initiate directory structure guidelines

Directory	Guidance
MAD_ROOTDIR directory for the Master Data Engine software	The MAD_ROOTDIR, or installation directory is created when you run the installer. Because the installer creates the directory structure during installation, you do not have to create this directory in advance. You can choose to accept the Master Data Engine installer directory defaults when you run the installer or specify a different directory. The administrative user account (which is the operating system user account used to install the software on the host (for example, initiate) must have read, write, and execute permissions for the directory in which you are installing. For IBM AIX, Linux, or Solaris installations, the associated user group (for example, initgrp) must also have read, write, and execute permissions.
MAD_HOMEDIR directory for instances	The MAD_HOMEDIR directory is the top-level directory in which to organize instances of the software such as engine, entity managers, and LDAP directory server instances. Instance directories are created when you run the madconfig utility create_instance target. Typically this directory uses the convention \ibm\initiate\home*instance_name* \, where *instance_name* is the name of your Master Data Engine instance. You are not required to use this standard. **Remember:** On IBM AIX, Linux, or Solaris operating systems, the ibm/initiate/home directory is not the same as or associated with the user home directory. When you use the madconfig utility to create your instance, these subdirectories are created under each MAD_HOMEDIR: • \bin - Contains scripts and supporting programs • \etc - Reserved for special configuration options that might not be installed at all sites • \inst - Contains runtime instance directories • \log - Contains MPINET output logs (.mlg files) • \sql - Contains SQL scripts • \support - Temporary folder used by IBM Software Support during troubleshooting

Table 13. IBM Initiate directory structure guidelines (continued)

Directory	Guidance
Instance directories (`inst`)	Within the `MAD_HOMEDIR` directory, you have one or more directories in which to group associated instances on the host. These directories are created for you when the instances (Master Data Engine, entity manager, and LDAP directory server) are created and are located in the *MAD_HOMEDIR*`\inst` directory. For example, you might have *C:*`\`*MAD_HOMEDIR*`\inst\mpinet_prod100_1` and *C:*`\`*MAD_HOMEDIR*`\inst\mpientmgr_prod100_1`. After you have created your instances, these subdirectories are created under each `inst` directory: • `\conf` - Contains the instance configuration files. • `\data` - Working directory used for runtime processes • `\ext` - Contains the instance deployment files • `\index` - Contains the full text index if you are using the IBM Initiate Flexible Search feature. If you are not using an index, this directory remains empty. • `\ldap` - Contains LDAP-specific files for the instance. This directory is created only if an LDAP directory server is embedded with the engine instance. • `\tmp` • `\work` - Working directory used for runtime processes

This sample directory structure shows multiple instances associated with the production database. In this example, you see two `MAD_HOMEDIR` directories: prod and test. Within the prod `\inst` directory, you see two entity manager instances (one for household [hh] entity and one for identity [id] entity), a stand-alone LDAP instance, and two engine instances (mpinet_prod_1 and mpin_prod_2).

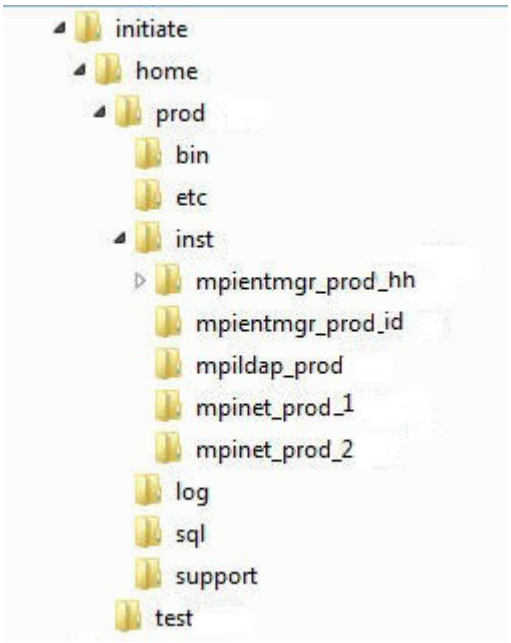

Figure 2. Sample directory structure with multiple instances

System and software users for the Master Data Engine

On IBM AIX, Linux, or Solaris operating systems, a new group and user are suggested for installing the Master Data Engine.

The user is the account you use to run the Master Data Engine installer and other engine utilities to create engine processes. In turn, the Master Data Engine sets that user account as the "owner" of the processes on the host (this user can start and stop the processes). The user must be a member of a group that can provide shared permissions for all users of the IBM Initiate® Master Data Service® on the host. The software does not require any special values for the user and group names. For ease of installation, it is suggested that you use the same names for users and groups on engine hosts that write to the same database. For example, configure these names:

- User: `initiate`
- Group: `initgrp`

For all installations, and as part of configuring the database, you must create at least one database user account. When creating your engine instance, you are prompted to specify a database user.

Related concept

"Master Data Engine database configuration"

User limits on UNIX and Linux platforms

The Master Data Engine is a high performance application and uses large amounts of RAM to provide fast response time. On larger installations, the memory and file requirements can exceed the default ulimit (user limits) setting of the operating system.

Set all ulimit settings related to memory (stack and heap) and the number of open files per process to "unlimited". The supported operating systems of IBM AIX, Linux, or Solaris have different options for setting the memory, stack, and open files settings (for example, through user creation or within a user `.profile` or `.login` file). For complete details, see the vendor-specific documentation for ulimit settings.

Master Data Engine database configuration

Before you begin configuration of your Master Data Engine database, consider your naming convention, user accounts, and platform-specific conventions.

For database creation instructions, see the applicable database product documentation.

Naming: Give the associated database instance, user account, and data source configuration the same name. You might also want to include the IBM Initiate Master Data Service version in your name. For example, you might name each of these elements `prod_100` for the production database. Using this naming convention can help other members of your organization and IBM Software Support understand the mapping between instances, accounts, and databases.

Database user account: All installations require at least one database user account. In order to bootstrap the database (which is typically done when creating the initial Master Data Engine instance), perform an engine upgrade, define new entity types, or create implementation-defined segments, the database user account must have certain permissions. This user account must have permission to:

- Create table and drop table
- Create index and drop index
- Select, insert, update, and delete

After the database is bootstrapped and entity types and implementation-defined segments are configured, you can opt to restrict the user account if your organization requires this. A restricted user account has only select, insert, update, and delete permissions.

It is suggested that you configure a one-to-one relationship between the database user and the database so that users do not have access to multiple databases. This model provides a security layer that can prevent one database user from dropping the tables of another.

Record the database user account credentials; you need this information to complete the Master Data Engine installation worksheets and when you create the engine instance runtime environment.

Connections: The database connection count is the sum of connections used by the Master Data Engine and by any entity managers. Some Master Data Engine or IBM Initiate Workbench processes require additional database connections, which are closed when the process is completed. In addition, LDAP processes require connections intermittently. Allow additional connections for these processes in your configuration.

Microsoft SQL Server database: If you are using a Microsoft SQL Server database, you must configure the database with a specific collation name and enable TCP/IP.

- For Collation Name, select **SQL Latin1_General_CP 1_CS_AS** (case sensitive). If you are implementing a language other than U.S. English, see the related concepts listed at the end of this topic.
- Verify that TCP/IP is enabled. Use the SQL Server Configuration Manager to access **SQL Server 2005 Network Configuration** > **Protocols for MSSQLSERVER**. In the right pane, right-click **TCP/IP**, and then select **Properties**.

Oracle database: If you are using an Oracle database, use this command to create the database:

```
CREATE DATABASEdname...CHARACTER SET AL32UTF8
```

You must set the character length semantics for Unicode. Set the variable NLS_LANG_SEMANTICS to CHAR (the default setting is BYTE). Run this command:

```
ALTER SYSTEM SET NLS_LENGTH_SEMANTICS=CHAR SCOPE=BOTH
```

IBM DB2: If you are using an IBM DB2 database, use this command to create the database:

```
CREATE DATABASEdnameUSING CODESET UTF-8 TERRITORYterritory code
```

When the database is created from a command line the default table space page size is 4K. However, all of the tables in IBM Initiate Master Data Service database do not fit into a table space with 4K page size. If performance is not an issue in your implementation, you can use a 16K table space to hold all the tables. If performance is a consideration, use a 8K or 16K table space for those tables that do not fit into a 4K table space and use 4K table spaces for the rest. You should also use separate table spaces to store tables and their indexes.

Use this command to create your database:

```
CREATE DATABASE dname AUTOMATIC STORAGE YES ON dbase_location DBPATH ON
location_of_transaction_log USING CODESET UTF-8 TERRITORY territory code
COLLATE USING IDENTITY PAGESIZE 4K
```

Then, create table spaces and their corresponding buffer pools. For example, to create a 4K buffer pool use this command:

```
CREATE BUFFERPOOL bufferpool_name IMMEDIATE SIZE starting_page_size
AUTOMATIC PAGESIZE 4K
```

Next, use this command to create the 4K table space:

```
CREATE TABLESPACE tablespace_name PAGESIZE 4K MANANGED BY AUTOMATIC STORAGE
AUTOSIZE YES BUFFERPOOL bufferpool_name NO FILE SYSTEM CACHING
```

You can change the page size in the above commands to create 8K or 16K buffer pools and table spaces.

These are the tables that require an 8K or16K table space for the table or their index.

- MPI_STRFREQ - 4K data table space, 8K index table space
- MPI_STRANON - 4K data table space, 8K index table space
- MPI_DVDXSTD - 8K data table space, 8K index table space
- MPI_APPPROP - 16K data table space, 8K index table space
- MPI_STRCONFIG - 4K data table space, 8K index table space
- MPI_STRSET - 4K data table space, 8K index table space
- MPI_USRPROP - 16K data table space, 8K index table space
- MPI_AUDNOTE - 16K data table space, 8K index table space
- MPI_WFTYPE - 16K data table space, 8K index table space
- MPI_WFTYPESTEP - 16K data table space, 8K index table space
- MPI_WFXCRCONFIG - 16K data table space, 8K index table space
- MPI_WFINSTANCE - 16K data table space, 8K index table space
- MPI_CLUSTERCFG - 16K data table space, 8K index table space
- MPI_IDXIQUE - 16K data table space, 8K index table space

Additional suggestions for your IBM DB2 database include:

- Activate the database after creation.
- Set the parameter DB2_INLIST_TO_NLJN to `ON`.
- Set database parameter STMT_CONC LITERALS to `immediate`.

- Set the cardinality of queue tables MPI_MEMIQUE, MPI_MEMOQUE, MPI_ENTIQUE_ID, MPI_ENTOQUE_ID, MPI_ENTIQUE_ORG, MPI_ENTOQUE_ORG, and the MPI_RELLINK table to `VOLATILE`.
- Re-organize all tables as needed.
- Run statistics on all tables as needed.
- Rebind data direct bind packages with `RESOLVE ANY REOPT ONCE` option.

Related concepts

"Database prerequisites for using Unicode in the Master Data Engine" on page 195

Related reference

"Master Data Engine installation worksheet" on page 6

"Master Data Engine instance worksheet" on page 6

"Data source worksheet" on page 4

Master Data Engine and instances installed on different drives (Microsoft Windows)

Engine instances can be physically located on drives other than where the Master Data Engine is installed.

You have the option of installing the software and the instances on different physical drives. One reason some system administrators prefer this structure is because each instance has activity logs. These logs can grow quite large. The most common reason for using different physical drives is that in many non-production environments, you might have multiple copies of the software, and multiple instances. For example, you might have separate software installations and instances for developers, testers, and training. You might also have two versions of the software that you use during an upgrade or conversion project.

To locate an instance that is on a different physical drive on a Microsoft Windows operating system, both physical drives must have entries in the `java.policy` file. This file is located in the `MAD_ROOTDIR/conf` directory.

These entries are in the form of:

```
grant codebase "file:/C:/-" {
        permission java.security.AllPermission;
};

grant codebase "file:/D:/-" {
        permission java.security.AllPermission;
};
```

Replace the `C:`and `D:`drive letter entries in these examples with the physical drive letters used in your `MAD_ROOTDIR` (Master Data Engine installation directory) and `MAD_HOMEDIR` (instance installation directory) environment variables.

Chapter 5. Installing the Master Data Engine

You can run the installer on Microsoft Windows, IBM AIX, Linux, or Solaris operating systems.

About this task

This procedure creates the `MAD_ROOTDIR` for your Master Data Engine runtime environment. Use this procedure each time you want to install a runtime environment.

Procedure

1. To install the Master Data Engine, run the file appropriate for your operating system: `IBM_Initiate_MasterDataEngine_10.0.0.n_xxx.exe` for Microsoft Windows or `IBM_Initiate_MasterDataEngine_10.0.0.n_xxx.bin` for IBM AIX, Linux, or Solaris.
2. Follow the installer prompts to install the Master Data Engine.

What to do next

After you installing the engine, continue with configuring your environment.

Related reference

"Master Data Engine installation worksheet" on page 6

"Master Data Engine instance worksheet" on page 6

Related task

"Creating a Master Data Engine instance" on page 47

Chapter 6. Configuring the Master Data Engine environment

Configuring your Master Data Engine environment includes creating runtime environments and conducting any applicable post-installation tasks.

Use the installation worksheets as a guide when you create your instances. Complete the applicable procedures in the order presented.

1. Configure the initial Master Data Engine runtime environment:
   - If desired, encrypt the password for the database user account.
   - Create an engine data source.
   - If applicable, implement a stand-alone instance of the IBM Initiate LDAP directory server.
   - Create an engine instance. Instances are created using the madconfig utility create_instance target. In a multi-environment setting, you also have the option of using an automated madconfig utility script and running a recorded response file.
   - If applicable, implement a stand-alone entity manager.
2. Conduct the post installation tasks, including:
   - Start the engine instance.
   - Confirm that the engine instance is started.
   - Configure required environment variables.
   - Configure globalization or SSL security if necessary.

Related concept

Chapter 3, “Planning your Master Data Engine installation,” on page 25

Appendix A, “LDAP Directory Server for the Master Data Engine,” on page 197

Chapter 8, “Entity managers,” on page 73

Chapter 2, “Installation worksheets,” on page 3

Database user account password encryption

To avoid the use of a clear-text password in configuration files, you can encrypt the password by using either the madpwd2 or madpwd3 utilities.

Using either of these utilities encrypts the password string for the Master Data Engine database user account.

If you are planning to use AES encryption (Advanced Encryption Standard (AES) 128-, 192- or 256- bit), you must use the madpwd3 utility. The madpwd3 utility encrypts a password with the generated AES key and iv (initialization vector).

Related concept

“Master Data Engine database configuration” on page 36

Related reference

"madpwd2 utility" on page 128

"madpwd3 utility" on page 128

Appendix I, "AES encryption," on page 243

Encrypting the password for the database user account with the madpwd2 utility

Use the mapdpwd2 utility to encrypt database passwords.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `bin` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.0.x\bin`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.0.x/bin`
2. Run this command:

   `madpwd2.exe -e plaintextPwd`

   where *plaintextPwd* is the original password for the database user account in plain text format. You created the account as a part of configuring the database.

   This text is displayed in the output:

   ```
   PLAINTEXT = (plaintextPwd)
   ENCRYPTED = (2AEBE6CA01432400F4619011D8AC5B50)
   DECRYPTED = (plaintextPwd)
   ```

   where *2AEBE6CA01432400F4619011D8AC5B50* is the newly encrypted password string.
3. Save the encrypted password string in a safe place; then, apply the literal string when prompted for the corresponding value when you create you engine instance.

   Related concept

   "Master Data Engine database configuration" on page 36

   Related task

   "Creating a Master Data Engine instance" on page 47

Encrypting the password for the database user account with the madpwd3 utility

If your are planning to use AES encryption (Advanced Encryption Standard (AES) 128-, 192- or 256- bit) for your implementation, you must use the madpwd3 utility.

Before you begin

Before encrypting the AES password, review the "AES encryption" topic. Verify that these requirements have been met.

- You have set the these variables in the `com.initiate.server.system.cfg` file:
  - On Microsoft Windows, set `MAD_SSLLIB=ssleay32.dll` and `MAD_SSLCRYPTOLIB=libeay32.dll` on Microsoft Windows
  - IBM AIX, Linux, or Solaris, set `libssl.so` and `libcrypto.so`
- You have generated the AES key and initialization vector (iv).

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to thebin directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.0.x\bin`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.0.x/bin`

2. The madpwd3 utility allows for the key and iv to be entered either from a file or directly on the command line. Use the `-keyfile` and `-ivfile` options to specify as a file. Use the `-key` and `-iv` options to directly enter them. There is no length limit on the password input.

   a. Run this command to specify as a file:

      `madpwd3.exe -keyfile filename -ivfile filename -in text`

      Provide the full path and file names for the keyfile and iv file. Text is the plain text version of the password you created as a part of completing the engine database configuration.

   b. Run this command to specify directly:

      `madpwd3.exe -key key -iv iv -in text`

      where *key* and *iv* are the contents of the `.dat` files that you created during key generation, and *text* is the plain text version of the password that you created as a part of completing the engine database configuration.

      The output appears as:

      ```
      PLAINTEXT = (plaintextPwd)
      ENCRYPTED = (99CA56BDF62638567F456941650237AB)
      DECRYPTED = (plaintextPwd)
      ```

Related concepts

"Master Data Engine database configuration" on page 36

Appendix I, "AES encryption," on page 243

Related task

"Generating AES keys and password" on page 244

Creating a data source

Use this procedure to create a data source for your Master Data Engine.

Before you begin

Complete the engine data source worksheet.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`

2. Run the applicable command:

   Microsoft Windows: `madconfig create_datasource`

   IBM AIX, Linux, or Solaris: `madconfig.sh create_datasource`

3. For each prompt, review the information; type the corresponding value that you defined in the data source worksheet; and then press Enter. To apply the default value, press Enter without typing a value.

4. In the output, confirm that `BUILD SUCCESSFUL` displays

What to do next

If your implementation requires a stand-alone LDAP directory server, create the stand-alone LDAP instance. Otherwise, continue with creating engine instances by using the madconfig utility. You can review full madconfig utility usage by running the applicable command:

Microsoft Windows: `madconfig –projecthelp`

IBM AIX, Linux, or Solaris: `madconfig.sh –projecthelp`

Related reference

"IBM Initiate LDAP directory server stand-alone instance" on page 29

"Data source worksheet" on page 4

"Master Data Engine instance worksheet" on page 6

"madconfig utility" on page 102

Related tasks

"Creating a Master Data Engine instance" on page 47

Creating a stand-alone IBM Initiate LDAP directory server instance

Use this procedure to create a stand-alone LDAP directory server instance.

Before you begin

Complete the LDAP directory server worksheet.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the applicable command:

   Microsoft Windows: `madconfig create_ldap`

   IBM AIX, Linux, or Solaris: `madconfig.sh create_ldap`
3. For each prompt, review the information; type the corresponding value that you defined in LDAP directory server worksheet; and then press Enter. To apply the default value, press Enter without typing a value.
4. In the output, confirm that `BUILD SUCCESSFUL` displays.

   Related reference

   "LDAP directory server worksheet" on page 14

Creating stand-alone entity managers

If your implementation requires, use this procedure to create a stand-alone entity manager.

Before you begin

- Review the entity managers topic.
- Complete the stand-alone entity manager worksheet.
- Create a Master Data Engine instance.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the applicable command:

   Microsoft Windows: `madconfig create_entitymgr`

   IBM AIX, Linux, or Solaris: `madconfig.sh create_entitymgr`
3. For each prompt, review the information; type the corresponding value that you defined in stand-alone entity manager worksheet. Press Enter. To apply the default value, press Enter without typing a value.
4. In the output, confirm that `BUILD SUCCESSFUL` appears.

   To review additional madconfig utility options, run the applicable command to review full madconfig usage:

   Microsoft Windows: `madconfig –projecthelp`

   IBM AIX, Linux, or Solaris: `madconfig.sh –projecthelp`

   Related concepts

   Chapter 8, “Entity managers,” on page 73

   Related reference

   “Stand-alone entity manager worksheet” on page 16

   “madconfig utility” on page 102

   Related tasks

   “Creating a Master Data Engine instance”

Creating a Master Data Engine instance

Use this procedure to create a new Master Data Engine instance. Before creating your instance, you must have a data source created.

Before you begin

Complete the Master Data Engine installation worksheets.

You must have a data source for your engine instance to point to.

If your implementation requires that the Master Data Engine be FIPS-compliant, review the FIPS compliance topics before creating your instance (see related topics list at the end of this topic).

About this task

Attention: If you are upgrading an existing instance (for example, upgrading version 9.7 to 10.0) do not bootstrap the database. Bootstrapping overwrites the existing data without an automated method for recovery. (See the "Upgrading the Master Data Engine environment topic listed at the end of this topic.)

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the applicable command:

   Microsoft Windows: `madconfig create_instance`

   IBM AIX, Linux, or Solaris: `madconfig.sh create_instance`
3. For each prompt, review the information; type the corresponding value that you defined in the engine instance worksheet; and then press Enter. To apply the default value, press Enter without typing a value.
4. In the output, confirm that `BUILD SUCCESSFUL` displays.

   You can review full madconfig utility usage by running the applicable command:

   Microsoft Windows: `madconfig –projecthelp`

   IBM AIX, Linux, or Solaris: `madconfig.sh –projecthelp`

What to do next

If your implementation requires a stand-alone LDAP directory server or entity manager, you must create those instances.

Otherwise, continue with post-installation tasks.

Related concepts

Appendix K, “FIPS compliance,” on page 255

Chapter 7, “Upgrading the Master Data Engine environment,” on page 63

Related reference

“madconfig utility” on page 102

“Data source worksheet” on page 4

“Master Data Engine instance worksheet” on page 6

Related tasks

“Creating a data source” on page 45

“Enabling FIPS compliance in the Master Data Engine” on page 255

“Creating a stand-alone IBM Initiate LDAP directory server instance” on page 46

“Creating stand-alone entity managers” on page 46

Creating an automated madconfig utility script

You can use the madconfig utility `–recordfile` option to record a set of responses in a property file that provides input to any madconfig operation. A response property file is useful for running madconfig options on a scheduled basis, or for simplifying scripted installations in a test or multi-environment setting.

Procedure

- To record a response file, run the madconfig utility with the `–recordfile` option and specify a target property file in the command. For example, use the command

  ```
  madconfig -recordfile myfile.properties create_datasource
  ```

  where *myfile.properties* is the name of the property file that stores your responses. You can define a full path to the property file.
- When you record the response file, the madconfig operation you are recording is performed (that is, the madconfig utility both performs the operation and records your responses). For example, if you run this command:

  ```
  madconfig -recordfile myfile.properties create_datasource
  ```

  a data source is created when the responses are recorded.
- You can edit the property file after its creation to modify the responses, if needed. If you want to add comments to your property file, use number signs (#) at the beginning of each comment line. For example:

  ```
  # ------------------------------------------------------------
  # ---- Database properties ----
  # ------------------------------------------------------------
  mad.db.name=MyDatabase
  mad.db.host=localhost
  mad.db.type=mssqlu
  ```

  See the documentation on the individual madconfig utility operations for full details about the prompts and variables for each.

Running the madconfig utility by using a recorded response file

Use this procedure to run the madconfig utility by using a recorded property file.

Procedure

Run your madconfig utility operation by using the `–propertyfile` option and the name of the property file where your responses are recorded. For example:

```
madconfig -propertyfile myfile.properties create_datasource
```

where *myfile.properties* is the name of the property file where your responses are stored. Be sure to include any path information with the property file, if it is stored outside of the `\scripts` directory where the madconfig utility command is run.

Related reference

"madconfig utility" on page 102

Merging multiple response property files

You can run multiple madconfig utility operations at one execution of madconfig using a single property file. To do this, you must manually merge property files for multiple operations.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`

2. To run multiple madconfig operations at one time against a merged property file, use syntax that conforms to this example:

   ```
   madconfig –propertyfile myMergedFile.properties create_datasource
   create_instance
   ```

   where *myMergedFile.properties* is the name of the merged property file where your responses to `create_datasource` and `create_instance` are stored.

Installation error log

If you are unable to complete the engine installation successfully, see the engine installation error log (`install.log`).

This file is located by default at the root of the installation directory, for example:

Microsoft Windows: `C:\Program Files\IBM\Initiate\Engine10.0.x\install.log`

IBM AIX, Linux, or Solaris: `/opt/IBM/Initiate/Engine10.0.x/install.log`

If you have any problems or questions while installing the Master Data Engine, contact IBM Software Support.

Master Data Engine configuration files

The Master Data Engine employs a Java-based service platform that allows for remote management. Master Data Engine configuration variables are set within a set of `com.initiate.server.*.cfg` files maintained within the engine instance `\conf` directory.

Attention: If you are upgrading from version 9.0 or earlier, review the "Configuration file changes" topic (see "Related reference" section in this topic).

This list details the configuration files found in the `\conf` directory.

- `com.initiate.server.appsvcs.cfg` - Maintains the loginProvider and contextPool parameters that are required if, when creating the instance, you chose for the Engine to provide embedded Application Services. You typically do not have to edit these settings.
- `com.initiate.server.entity.cfg` - Contains a single parameter that governs whether the entity manager relationship linker is enabled for the instance.
- `com.initiate.server.event.cfg` - Contains the parameters that provide for event notification from the Master Data Engine to external systems.
- `com.initiate.server.queue.cfg` - Contains parameters that define the entity manager queue process.
- `com.initiate.server.features.cfg` - Contains a single parameter that lists which embedded features are enabled for the instance, for example LDAP, entity manager, or application services. You typically do not have to edit these settings.
- `com.initiate.server.handler.cfg` - Contains a single variable that indicates the context pool size for callout handlers. The default is 2.

- com.initiate.server.hibernate.cfg - Maintains parameters for an internal-only object and relational persistence and query service. You typically do not have to edit these settings.
- com.initiate.server.jdbc.cfg - Contains parameters for enabling database connectivity including database user name, password. database name, server name, and port.
- com.initiate.server.jmx.jmxmp.cfg - Defines the service name and URL for the JMX Message Protocol (JMXMP), which facilitates the managing and monitoring of services and applications. An addition parameter in the file determines whether to use SSL for JMXMP communications.
- com.initiate.server.jmx.rmi.cfg - Maintains values that enable remote connection for JMX-based managing and monitoring. (RMI stands for Remote Method Invocation.) An additional parameter in the file determines whether to use SSL for the JMX RMI communications.
- com.initiate.server.ldap.cfg - Contains connectivity and security parameters for internal and external LDAP servers.
- com.initiate.server.logic.cfg -
- com.initiate.server.net.cfg - Maintains connectivity parameters for the Master Data Engine.
- com.initiate.server.search.cfg - Contains parameters for IBM Initiate Flexible Search .
- com.initiate.server.smt.cfg - Contains a single parameter that defines the language locale for the Java layer. The default is en_US.
- com.initiate.server.system.cfg - Contains most of the configuration parameters for the instance including global settings for logging, the language locale setting (C layer), and the detail settings for SSL communication.
- com.initiate.server.tasks.cfg - Contains parameters for the embedded engine task (process) manager. The engine task manager polls certain events or processes and executes the associated logic on a scheduled basis. Parameters include polling intervals, wait time, task expiration settings, and frequency-based bucketing settings.
- com.initiate.server.web.cfg - Contains a single port parameter for the embedded web server.
- logj4.xml - Contains log4j appenders and parameters for configuring logging for the instance.
- wrapper.conf - Maintains a number of internal-only parameters, including the MAD_ROOTDIR and MAD_HOMEDIR parameters that indicate the file system paths to the engine and instance. You typically do not have to edit these settings.

Some configuration variables exist in configuration files only if you choose to enable non-default settings during instance creation. For example, MAD_ENCODING (which governs the system globalization behavior) is shown only if you opt for a setting other than the default, latin1. For any MAD_* variable not listed in this table, you can add the variable directly within the com.initiate.server.system.cfg file.

Related concept

"Master Data Engine environment variables" on page 87.

Configuration file changes

If you are upgrading from IBM Initiate Master Data Service version 9.0 or earlier, all of the variables that you previously saw in the `engine.properties` file have been distributed to the various `com.initiate.server.*.cfg` files. All of the configuration variables from the `ldap.properties` file have been moved to `com.initiate.server.ldap.cfg`.

Administrators upgrading the Master Data Engine from an earlier release must manually move any custom configuration settings from the `engine.properties` and `ldap.properties` files to the new `*.cfg` files. This process is described in post-upgrade task topic.

The installation process for stand-alone entity managers also no longer creates an `engine.properties` configuration file. As with the Master Data Engine, configuration parameters for a stand-alone entity manager previously set within engine.properties are now set within `.cfg` files in the entity manager `\conf` directory.

The Master Data Engine configuration files topic contains a listing of configuration files. .

In some cases, the variable names have changed. This list describes the variables and their new file locations, and names if applicable.

Variable
: **New location (and new name if applicable)**

MAD_AUDIT
: renamed to `AuditLog` within `log4j.xml`

MAD_CALLBACKLIB
: `com.initiate.server.system.cfg`

MAD_CONFNAME
: `com.initiate.server.system.cfg`

MAD_CONNSTR
: `com.initiate.server.system.cfg`

MAD_CTXLIB
: `com.initiate.server.system.cfg`

MAD_DBTYPE
: `com.initiate.server.system.cfg`

MAD_DBNAME
: `com.initiate.server.system.cfg`

MAD_DBPASS
: `com.initiate.server.system.cfg`

MAD_DBSERVERS
: `com.initiate.server.system.cfg`

MAD_DBSETUP
: `com.initiate.server.system.cfg`

MAD_DBSQL
: replaced by `SqlLog` in `log4j.xml`

MAD_DBUSER
: `com.initiate.server.system.cfg`

MAD_DBXTEST
com.initiate.server.system.cfg

MAD_DDLFILE
com.initiate.server.system.cfg

MAD_DEBUG
managed by NativeLog in log4j.xml

MAD_DICTIMEOUT
com.initiate.server.system.cfg

MAD_HOMEDIR
wrapper.conf

MAD_INSTDIR
wrapper.conf

MAD_IPVERSION
com.initiate.server.system.cfg

MAD_LOGPFX
controlled by ConversionPattern parameters within log4j.xml

MAD_LOGNAME
replaced by mad.log.name within com.initiate.server.system.cfg

MAD_OBJCODE
com.initiate.server.system.cfg

MAD_PERFLOG
renamed to PerformanceLog in log4j.xml

MAD_ROOTDIR
wrapper.conf

MAD_SECLIB
com.initiate.server.system.cfg

MAD_SMTLIST
com.initiate.server.system.cfg

MAD_SRVNO
com.initiate.server.system.cfg

MAD_SSLFIPSMODE
com.initiate.server.system.cfg

MAD_STOFILE
com.initiate.server.system.cfg

MAD_TABPFX
com.initiate.server.system.cfg

MAD_TABSFX
com.initiate.server.system.cfg

MAD_TIMER
replaced by TimerLog in log4j.xml

MAD_TRACE
managed by NativeLog in log4j.xml

MAD_UNLDIR
com.initiate.server.system.cfg

MAD_UNLFSR
com.initiate.server.system.cfg

AlgorithmLog
new variable, set in log4j.xml

com.initiatesystems.hub.jmx.objectname=com.initiatesystems:service=MPINETPROD900_1
mad.jmx.objectname within com.initiate.server.system.cfg (This variable is typically not changed.)

com.initiatesystems.hub.jmx.mgmtregport
serviceUrl within com.initiate.server.jmx.jmxmp.cfg

com.initiatesystems.hub.jmx.profiles
useSSL within com.initiate.server.jmx.jmxmp.cfg

com.initiatesystems.hub.handler.cbthreads
contextPoolSize within com.initiate.server.handler.cfg

com.initiatesystems.hub.ldap.config.cache.timeout.seconds
ldap.config.cache.timeout.seconds within com.initiate.server.ldap.cfg

com.initiatesystems.hub.job.workDir
This variable is no longer used.

javax.net.ssl.keyStore
com.initiate.server.system.cfg

javax.net.ssl.trustStore
com.initiate.server.system.cfg

javax.net.ssl.keyStorePassword
com.initiate.server.system.cfg

javax.net.ssl.trustStorePassword
com.initiate.server.system.cfg

javax.net.ssl.keyStoreType
com.initiate.server.system.cfg

javax.net.ssl.trustStoreType
com.initiate.server.system.cfg

com.initiatesystems.hub.executor.corePoolSize
This variable is no longer used.

com.initiatesystems.hub.executor.queueSize
This variable is no longer used.

com.initiatesystems.hub.mpientmgr.entlinker.enabled
This variable is no longer used.

com.initiatesystems.hub.mpientmgr.rellinker.enabled
relationshipLinkerEnabled in com.initiate.server.entity.cfg

com.initiatesystems.hub.mpinet.host
host within com.initiate.server.net.cfg

com.initiatesystems.hub.mpinet.port
port within com.initiate.server.net.cfg

com.initiatesystems.hub.mpinet.threads
contextPoolSize within com.initiate.server.net.cfg

com.initiatesystems.hub.mpinet.entmgrs
controlled by featuresBoot within com.initiate.server.features.cfg

com.initiatesystems.hub.mpinet.tagmanager
port within com.initiate.server.net.cfg

com.initiatesystems.hub.mpinet.seclib
useSSL within com.initiate.server.net.cfg

Other variable changes include:

- The MAD_NOCHG variable has been removed. This variable prevented log file rollover. While preventing log rollover is not suggested, if you do require a rollover, you can remove the %s from the mad.log.name variable in com.initiate.server.system.cfg.
- MAD_LOGDIR has also been removed. The equivalent behavior can be achieved by specifying the full path (including the file name) as the value for mad.log.name.

Related task

"Conducting the Master Data Engine post-upgrade tasks" on page 70

Post-installation tasks

After you have installed the Master Data Engine, there are some additional configuration steps and testing that you might need to perform.

- Test your instances by starting and stopping each instance
- Create additional engine instances or runtime environments to fully create the planned database architecture.
- Create additional stand-alone entity managers.
- Configure and deploy an event handler if you are using event notification. If you did not embed your event manager, you also need to create a stand-alone event manager.
- Implement SSL communication.
- Globalize one or more of the runtime environments.
- Configure one or more instances to communicate with an external corporate LDAP directory server.
- Per engine host, create applicable environment variables in the corresponding configuration files.

Related concepts

"Master Data Engine database configuration" on page 36

"Master Data Engine installation in a high-availability environment" on page 27

Appendix A, "LDAP Directory Server for the Master Data Engine," on page 197

Related task

Chapter 5, "Installing the Master Data Engine," on page 41

"Creating stand-alone entity managers" on page 46

"Enabling event notification" on page 83

Chapter 10, “Configuring Master Data Engine environment variables,” on page 87

Chapter 13, “Configuring SSL,” on page 191Chapter 14, “Configuring globalization of the Master Data Engine,” on page 195

Chapter 10, “Configuring Master Data Engine environment variables,” on page 87

Starting and stopping your instances

After you have created your initial runtime Master Data Engine environment and the associated instances, you should start and stop the instances to verify that they are running.

Each instance has a corresponding process on the host on which it is installed. Use the procedure specified for your operating system to start or stop an instance. For Microsoft Windows, you can use the Control Panel. For UNIX and Linux operating systems, you can use the madconfig utility. You can also use a batch or script file to start and stop an instance.

Starting an engine instance from the Microsoft Windows Control Panel

For Master Data Engine instances on Microsoft Windows , you can use the Control Panel to start and stop the associated services.

Procedure

1. From the Control Panel, open **Administrative Tools**, and then double-click **Services**.
2. From the list of services, select **IBM Initiate Master Data Engine 10.0.x** *name*, where *name* is the name of the engine instance that you want to start.
3. On left side of the list, click the **Start** link.

Stopping an engine instance from the Microsoft Windows Control Panel

Use this procedure to stop a Master Data Engine instance on Windows.

Procedure

1. From the Control Panel, open **Administrative Tools**, and then double-click **Services**.
2. From the list of services, select **IBM Initiate Master Data Engine 10.0.*x name***, where *name* is the name of the engine instance you want to stop.
3. On left side of the list, click the **Stop** link.

Starting an engine instance with the madconfig utility

Use this procedure to start a Master Data Engine instance with the madconfig utility.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the applicable command:

   Microsoft Windows: `madconfig start_instance`

   IBM AIX, Linux, or Solaris: `madconfig.sh start_instance`

3. Type the name of the instance you want to start, and then press Enter.
4. In the output, confirm that `BUILD SUCCESSFUL` appears.

Stopping an engine instance with the madconfig utility

Use this procedure to stop a Master Data Engine instance with the madconfig utility.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the applicable command:

   Windows: `madconfig stop_instance`

   IBM AIX, Linux, or Solaris: `madconfig.sh stop_instance`
3. Type the name of the instance you want to stop, and then press Enter.
4. In the output, confirm that `BUILD SUCCESSFUL` appears.

Starting an engine instance with its batch or script file

Use this procedure to start a Master Data Engine instance with a batch or script file.

Procedure

1. On the command line, go to the directory that contains the batch or script file for the instance. For example:

   Microsoft Windows: `cd MAD_HOMEDIR\inst\mpinet_name\conf`

   IBM AIX, Linux, or Solaris: `cd /MAD_HOMEDIR/inst/mpinet_name/conf`

   where *MAD_HOMEDIR* is the full path to the directory created for the associated runtime instances and *name* is the name of the runtime instance (for example, `prod` or `qa`).
2. Run the applicable command:

   Microsoft Windows: `mpinet_name.bat start`

   IBM AIX, Linux, or Solaris: `mpinet_name.sh start`

   where *name* is the name of the instance.
3. In the output, confirm that this statement appears:

   ```
   Starting the IBM Initiate Master Data Engine
   10.0.x name service...
   IBM Initiate Master Data Engine 10.0.x Name
   started.
   ```

   where *name* is the name of the engine instance.

Stopping an engine instance with its batch or script file

Use this procedure to stop a Master Data Engine instance with a batch or script file.

Procedure

1. On the command line, go to the directory that contains the `mpinet_instance.bat` file for the instance.
2. Run the applicable command:

   Windows: `mpinet_name.bat stop`

   IBM AIX, Linux, or Solaris: `mpinet_name.sh stop`

where *name* is the name of the instance.

3. In the output, confirm that this statement is displayed:

   ```
   Stopping the IBM Initiate Master Data Engine
   10.0.x Name service...
   IBM Initiate Master Data Engine 10.0.x Name
   stopped.
   ```

   where *Name* is the name of the engine instance.

Using the madconfig utility to confirm that an engine instance is running

You can manually verify whether a Master Data Engine instance is running by using the madconfig utility `ping_instance` command.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`

2. Run the applicable command:

   Microsoft Windows: `madconfig ping_instance`

   IBM AIX, Linux, or Solaris: `madconfig.sh ping_instance`

3. Type the name of the instance you want to ping, and then press Enter.
4. In the output, determine whether the instance is running by looking for the statement shown in the "sample output from pinging the engine instance" table.

Results

Table 14. Sample output from pinging the engine instance

Instance state	Sample output

Running	`madeng_mpinet_ping:`
	`OK`
	`ping_instance:`
	
	`BUILD SUCCESSFUL`

Not running	`Aug 11, 2010 3:20:07 PM org.apache.bsf.BSFManager exec`
	`SEVERE: Exception`
	`java.security.PrivilegedActionException:`
	`org.apache.bsf.BSFException: The application script`
	`threw an exception: java.net.ConnectException: Connection`
	`refused: connect BSF info: ANT at l`
	`...`
	`BUILD FAILED`

Using the MPINET protocol to confirm that an engine instance is running

You can use the MPINET protocol to ping a Master Data Engine instance or verify database connectivity. The MPINET option is useful for programmatic checks of instance status and is often used by load balancers.

About this task

The telnet client does not have to run from the same server that is hosting the engine instance or database.

Procedure

1. To ping the engine by using MPINET, type `ping` at the command line followed by a new line or line feed character to the engine MPINET port. (The default engine MPINET port number is 16000.)

   For example, you can simulate this action with any Telnet application: `telnet [host] [port]`, then type `PING` and press Enter. If the ping is successful, an `OK` message is displayed.
2. Use the `pingdb` command to verify database connectivity.

   For more information about restarting instances and managing the Master Data Engine environment, see the *IBM Initiate Master Data Service Software Operations Guide*.

Using ping requests to monitor Master Data Engine and database availability

If you are using a load balancer, the load balancer might need to check that the Master Data Engine and database are running.

About this task

Use this task to enter a socket ping request to the Master Data Engine MPINET port.

Procedure

1. To verify that the Master Data Engine is running:
   a. On the command line, type `ping` followed by a new line or line feed character to the engine MPINET port (for example, you can simulate this action with any telnet application: `telnet` *`host port`*. Then type `ping`. The default port is 16000. For example:

      ```
      telnet localhost 16000
      ping
      ```
   b. Press Enter.
2. To verify that the Master Data Engine has a connection to the database:
   a. On the command line, type `ping` followed by a new line or line feed character to the engine MPINET port (for example, you can simulate this action with any telnet application: `telnet` *`host port`*. Then type `pingdb`. The default port is 16000. For example:

      ```
      telnet localhost 16000
      pingdb
      ```
   b. Press Enter.

Results

If the ping is successful, an `OK` message is returned. If the ping is unsuccessful, a `NA` message is returned.

After the command completes, the telnet session ends.

The output of the command can vary based on the telnet client used. For example, after typing the ping command, you might see output similar to this:

```
telnet host port
Trying host IP address...
Connected to hostname(IP address).
```

```
Escape character is '^]'.
ping
OK
Connection closed by foreign host.
```

Starting an entity manager instance through the Microsoft Windows Control Panel

For Master Data Engine instances on Microsoft Windows, you can use the Control Panel to start the associated entity managers.

Procedure

1. From the Control Panel, open **Administrative Tools**, and then double-click **Services**.
2. From the list of services, select **IBM Initiate Entity Manager 10.0.x *Instance***, where *Instance* is the name of the entity manager you want to start.
3. On left of list, click **Start**.

   A progress indicator appears and then disappears when the service is started.

Stopping an entity manager instance from Microsoft Windows Control Panel

For Microsoft Windows engine instances, you can use the Control Panel to stop the associated entity managers.

Procedure

1. From the Control Panel, open **Administrative Tools**, and then double-click **Services**.
2. From the list of services, select **IBM Initiate Entity Manager 10.0.x *Instance***, where *Instance* is the name of the entity manager you want to stop.
3. On left of list, click **Stop**.

   A progress indicator displays and then disappears when the service is stopped.

Starting an entity manager instance with the madconfig utility

You can use the madconfig utility to start a stand-alone entity manager.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the applicable command:

   Microsoft Windows: `madconfig start_entitymgr`

   IBM AIX, Linux, or Solaris: `madconfig.sh start_entitymgr`
3. Type the name of the entity manager instance you want to start, and then press ENTER.
4. In the output, confirm that `BUILD SUCCESSFUL` appears.

Stopping an entity manager instance with the madconfig utility

You can stop a stand-alone entity manager by using the madconfig utility.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`

2. Run the applicable command:

   Microsoft Windows: `madconfig stop_entitymgr`

   IBM AIX, Linux, or Solaris: `madconfig.sh stop_entitymgr`

3. Type the name of the entity manager instance you want to stop, and then press ENTER.
4. In the output, confirm that `BUILD SUCCESSFUL` displays.

Starting an entity manager instance with its batch or script file

You can start a stand-alone entity manager instance by using a batch or script file.

Procedure

1. On the command line, and go to the directory that contains the entity manager batch or script file. For example:

   Microsoft Windows: `cd \`*`MAD_HOMEDIR`*`\inst\mpientmgr_`*`name`*`\conf`

   IBM AIX, Linux, or Solaris: `cd /`*`MAD_HOMEDIR`*`/inst/mpientmgr_`*`name`*`/conf`

   where *MAD_HOMEDIR* is the full path to the directory created for the associated runtime instances (for example, `prod` or `qa`), and *name* is the name of the entity manager instance.
2. Run the applicable command:

   Microsoft Windows: `mpientmgr_`*`name`*`.bat start`

   IBM AIX, Linux, or Solaris: `mpientmgr_`*`name`*`.sh start`

   where *name* is the name of the entity manager instance.
3. In the output, confirm that this statement is displayed, where *instance* is the name of the entity manager instance:

```
Starting the IBM Initiate Entity Manager 10.0.x
instance service...
IBM Initiate Entity Manager 10.0.0 instance started.
```

Stopping an entity manager instance with its batch or script file

You can stop a stand-alone entity manager by using a batch or script file.

Procedure

1. On the command line, go to the directory that contains the instance `mpinet_`*`instance`*`.bat` file.
2. Run the applicable command:

   Microsoft Windows: `mpientmgr_`*`name`*`.bat stop`

   IBM AIX, Linux, or Solaris: `mpientmgr_`*`name`*`.sh stop`

   where *name* is the name of the entity manager instance.
3. In the output, confirm that this statement displays, where *Instance* is the name of the entity manager instance:

```
Starting the IBM Initiate Entity Manager 10.0.x
Instance service...
IBM Initiate Entity Manager 10.0.0 Instance started.
```

Chapter 7. Upgrading the Master Data Engine environment

Upgrading your Master Data Engine environment from one major version to another major version (for example, 9.7 to 10.0) often requires assistance from both members from your organization and IBM Services or an IBM consulting partner.

Do not upgrade your implementation without first identifying the team and relying on guidance from that team.

If your IBM Initiate Master Data Service configuration includes multiple instances of the Master Data Engine, consider by using the Interceptor tool to reduce downtime during upgrade.

After carefully reviewing the getting started topics, you are almost ready to start the upgrade. Use this checklist to familiarize yourself with the process. The upgrade tasks must be completed in the order presented.

1. Conduct the pre-upgrade tasks, including:
   - Review upgrade considerations.
   - Shut down any current Master Data Engine runtime instances.
   - Create backups of all operational data, including:
     - Instance home directories on each engine host. This includes the entire `MAD_HOMEDIR` directory
     - Master Data Engine database
   - Complete the 10.0 installation worksheets.
     - Engine installation worksheet
     - Engine data source worksheet
     - Engine instance worksheet
2. Create the initial 10.0 Master Data Engine runtime environment, including:
   - Run the Master Data Engine installer.
   - If you want to encrypt the database user password, use the madpwd2 or the madpwd3 utilities. The madpwd2 utility is a proprietary IBM Initiate encryption algorithm that is meh quality, and does not require keys. The madpwd3 utility offers full FIPS-compliant encryption that requires a valid key file.
   - Create a Master Data Engine data source.
   - If applicable, create a stand-alone instance of the IBM Initiate LDAP directory server.
   - Create a Master Data Engine instance.
3. Upgrade the Master Data Engine database to version 10.0:
   - Understand the database upgrade process.
   - Complete the database upgrade worksheet.
   - Run the database upgrade.
4. Conduct the post-upgrade tasks, including:
   - Start the 10.0 engine instance.
   - Confirm that the 10.0 engine instance is started.
   - Evaluate and conduct the next post-upgrade steps as applicable:

  - Configure the initial 10.0 runtime environment to communicate with an external corporate LDAP directory server.
  - Copy any custom environment variables and other settings created before the upgrade into the initial 10.0 runtime environment.
  - Create additional 10.0 engine instances or runtime environments to mimic the architecture implemented before the upgrade.
- Work with IBM to verify the upgrade and to re-derive data (re-dvd), re-generate weights, and potentially adjust threshold.

Related concept

Appendix J, "Interceptor tool," on page 247

Chapter 3, "Planning your Master Data Engine installation," on page 25

Conducting the pre-upgrade tasks

Before you upgrade the Master Data Engine, there are a few tasks that you must complete.

About this task

Complete these tasks in the order presented.

Procedure

1. Review the 10.0 *IBM Initiate Master Data Service Release Notes*® to identify any new system requirements and data model changes.
2. Consult the individuals involved with your implementation to obtain guidelines and deployment-specific suggestions. These individuals can include members from your organization and IBM Services, or IBM consulting partners.
3. If upgrading from version 7.5 or earlier, you must store a backup of the existing `services.ini` file. While this file is no longer in use by the Master Data Engine, it is required for the Message Broker Suite components
4. Confirm that the Master Data Engine database you are upgrading is at the minimum supported version for upgrade. The minimum supported version to upgrade to 10.0 is version 5.2.
5. Shut down your runtime instances. This includes engine, entity manager, and any inbound or outbound Message Broker Suite instances. For information about the Message Broker Suite components, see the *IBM Initiate Master Data Service Software Operations Guide*. The Message Reader process can continue running and queueing up messages from the source systems until the brokers are upgraded.
6. Back up your operational data.
   - On each engine host, create a backup image of the instance home directories (*MAD_HOMEDIR*).
   - Create backup images of the data on each Master Data Engine database server.
7. Review the Master Data Engine installation worksheets (engine installation, data source, and engine instance) for the new version you are upgrading to and define your installation values.

   Related tasks

"Stopping an engine instance from the Microsoft Windows Control Panel" on page 56
"Stopping an engine instance with the madconfig utility" on page 57
"Stopping an engine instance with its batch or script file" on page 57
"Stopping an entity manager instance from Microsoft Windows Control Panel" on page 60
"Stopping an entity manager instance with the madconfig utility" on page 60
"Stopping an entity manager instance with its batch or script file" on page 61

Related reference

"Master Data Engine installation worksheet" on page 6
"Data source worksheet" on page 4
"Master Data Engine instance worksheet" on page 6

Creating the initial 10.0 runtime environment

After completing the pre-upgrade tasks, you are ready to create the initial 10.0 runtime environment.

About this task

Use this procedure to create your initial 10.0 runtime environment.

Procedure

1. Confirm that the pre-upgrade tasks are complete.
2. Install the 10.0 engine software by running the installer specific to your operating system.
3. If you want to encrypt the password for the database user account use the madpwd2 or madpwd3 utility.
4. Use the 10.0 madconfig utility to complete these substeps.
   a. Create the 10.0 engine data source.

      Attention: In creating the engine instance, do not bootstrap the database. Bootstrapping overwrites the existing data without an automated method for recovery.
   b. If applicable, create a stand-alone instance of the IBM Initiate LDAP directory server.
   c. Create the 10.0 engine instance.

Results

This environment is used to upgrade your existing database to 10.0. Continue with upgrading the engine database to version 10.0.

Related concepts

"Database user account password encryption" on page 43

"IBM Initiate LDAP directory server stand-alone instance" on page 29

"Entity manager stand-alone instance" on page 29

"Upgrade the Master Data Engine database to 10.0" on page 66

Related tasks

"Conducting the pre-upgrade tasks" on page 64

Chapter 5, "Installing the Master Data Engine," on page 41

"Creating a data source" on page 45

"Creating a Master Data Engine instance" on page 47

Upgrade the Master Data Engine database to 10.0

After creating the initial 10.0 runtime environment, you must upgrade the existing database. Before beginning, it is helpful to understand the upgrade process.

After reviewing this topic, use the database upgrade worksheet to define the values needed for the upgrade. You then use the madconfig utility to perform the upgrade.

Incremental upgrade steps. The incremental upgrade, or skip-level upgrade, processes the upgrade incrementally by version. The madconfig utility upgrade_instance target is used to perform this process. During the upgrade, a confirmation prompt is used to process through each applicable stage of the upgrade steps:

- From 5.2 to 6.0
- From 6.0 to 6.1
- From 6.1 to 7.0
- From 7.0 to 7.2
- From 7.2 to 7.5
- From 7.5 to 8.0
- From 8.0 to 8.1
- From 8.1 to 8.5
- From 8.5 to 8.7
- From 8.7 to 9.0
- From 9.0 to 9.2
- From 9.2 to 9.5
- From 9.5 to 9.7
- From 9.7 to 9.8
- From 9.8 to 10.0

Related to the upgrade steps, it is suggested that you confirm system version information and verify the database upgrade.

- Before running the madconfig utility, confirm that your system version information is set correctly to represent the version that Master Data Engine runtime environment is running. Version information is found in the mpi_syskey table. The `keyval` column for the row when `keyname` equals `"ALIGNDEX_VERSION"` must accurately reflect the major.minor version of the product.
- After each upgrade step (for example, from 5.2 to 6.0, from 6.0 to 6.1, and so on), verify that the database is upgraded to the applicable version. Conduct this verification before allowing the madconfig utility to continue with the next upgrade step.

Checks in upgrade process. The upgrade script runs three checks for these items:

- **Custom libraries.** If custom libraries are found, the script exits the process. You can bypass this check by deleting the custom library from the mpi_libhead table. One caution is that if the custom functionality has not been integrated into the mainline Master Data Engine in a subsequent release, the upgrade appears to run smoothly. However, the Master Data Engine might not function as expected, or it might fail to run. If the implementation requires the use of a custom library and that functionality is not part of the mainline product, perform the database upgrade manually.
- **Merge EID script.** For upgrades at version 6.1 and later, the script can be run to resolve enterprise IDs for merged member records. If the upgrade script determines that the Merge EID script is applicable, you are prompted to choose whether to discontinue the upgrade. Running the Merge EID script is not necessary for the upgrade to complete successfully. You can apply `n` (to not run the script) and continue with the upgrade process, or `y` to discontinue the upgrade process. After discontinuing, you can run the MrgEID script and then continue with the upgrade process.
- **Multiple stdcodes.** If multiple standardization codes (stdcodes) are detected between 5.2 and 6.0, the upgrade process stops. In this case, you must perform the upgrade from 5.2 to 6.0 manually.
- **Duplicate bucket roles.** During a 7.5 to 8.7 skip upgrade, the DVDCODE 'DVDPER' error identifies duplicate bucket role (nn) settings. This condition arises when a single bucket role is defined more than once within the mpi_dvdxbkt table and is accompanied with a maximum bucket frequency (maxbktfreq) setting > 0. To resolve this issue, either remove or reassign one of the duplicate bucket roles. You can then proceed with the upgrade. You must re-derive your data after the upgrade is complete.

  ```
  ERROR: DVDCODE 'DVDPER' contains duplicate bucket roles with max bucket
  frequencies - See BktRole (1).
  ERROR: DVDCODE 'DVDPER' contains duplicate bucket roles with max bucket
  frequencies - See BktRole (2).
  Please contact IBM Sofware Support for further assistance.
  The database upgrade from 7.5 to 8.7 is aborted...
  ```

- **Composite view record number definition violations.** During a 6.x to 8.7 skip upgrade, the CVW Recno/s error identifies cvwrecno (nn) definition violations that are not supported as of the 7.5 release. If you encounter this error, you must rebuild your composite views that contain wildcard settings for segrecno, srcrecno, or attrrecno. (Composite views are created inIBM Initiate Workbench.) You can then proceed with the upgrade.

  ```
  ERROR: CVW Recno/s nn are no longer allowed in version 7.5.
  CVW wildcards of '0' are not allowed for segrecno, srcrecno or attrecno.
  Please contact IBM Software Support for further assistance.
  The database upgrade from 6.X to 7.5 is aborted...
  ```

- **Obsolete Enterprise IDs.** During a 5.2 or 6.0 to 6.1 skip upgrade, you might encounter this warning. If encountered, you can select whether to continue or stop the upgrade process. Contact IBM Software Support for further assistance in cleaning up obsolete EIDs as part of this upgrade process.

  ```
  It appears that there are obsolete EIDs present.
  # Although not required, it is recommended that the database be cleaned
  up (EIDs be merged) first.
  # Please contact IBM Software Support if you need further assistance.
  # Shall we continue with the upgrade?
  ```

Upgrade limitations. Before you begin a skip-level upgrade by using `madconfig upgrade_instance`, become familiar with these limitations:

- Message Broker Suite components are not upgraded.

- Accuracy of the upgraded data is not verified.
- Weights are not regenerated automatically.
- Data is not re-derived automatically.
- Backup and recoveries are not performed automatically.

Before starting the upgrade, complete the database upgrade worksheet.

Database upgrade worksheet

Before beginning your upgrade, complete the database upgrade worksheet to define the required values.

Completing this worksheet in advance of the upgrade can help you quickly respond to upgrade prompts.

Table 15. Database upgrade worksheet

Configuration	Description	Your value
Instance name	Identify the name for the 10.0 engine instance you just created.	
Rebuild the database tables in place	Identify whether to rebuild the existing database tables by using the "in place" or "file system" method. The values determine how to execute the upgrade process: • `y` for the in place method. The in place method runs entirely within the database. It involves copying table data from existing tables; making the new release table changes; and then copying the data from the copied tables back to the updated tables. **Important:** If you select `y`, the transaction rollback log on the target database can grow large. As a result, make sure that you have sufficient disk space or disable transaction rollback during the upgrade process. – Pros: Work is managed on the database server rather than locally. – Cons: Requires enough free space on database server to accommodate the largest table that is being reloaded. • `n` for file system method. This method runs by dumping existing data to `.unl` files; making the new release table changes; and then reloading the `.unl` files to the existing database. The process requires adequate disk space to store the `.unl` files. – Pros: Offers the opportunity to use a faster method of table loading by using the native database loader. – Cons: Requires enough free space locally to accommodate the largest table that is being reloaded.	

Table 15. Database upgrade worksheet (continued)

Configuration	Description	Your value
Reload tables with ODBC	**Attention:** Skip defining this value unless opting to rebuild the database tables by using the file system method (as described in previous row). Identify whether to reload the existing database tables by using ODBC. The values are: • `y` to reload the existing tables by using ODBC. This option requires that the data source is wire protocol compatible. – Pros: Does not require native tools. – Cons: Tends to be slow. • `n` means that the tables are reloaded by using the native database tools. – Pros: Fast. – Cons: Requires database client software to be installed and configured.	
Confirmation	Identify whether to proceed with upgrading the database from its current version to the next level. The madconfig utility detects which version is currently installed in the specified database. Depending on the existing version, you might have to upgrade incrementally in several steps by applying `y` on the confirmation prompts for each upgrade level. Continue applying `y` until you reach the final step to upgrade to 10.0.	
Validate data dictionary	Identify whether to run the mpxdata utility to validate the data dictionary in the newly upgraded database. After each upgrade step (for example, from 8.5 to 9.0), the madconfig utility prompts you about validating the data dictionary.	
Continue	Identify whether to continue database upgrade process.	
Install database patch	Identify whether to install a patch release on the newly upgraded database. Depending on the existing version, you must upgrade incrementally in several steps by applying `y` on the confirmation prompts for each upgrade level until you reach the final step to upgrade to 10.0 patch version.	
Validate data dictionary	Identify whether to validate the data dictionary after the patch is installed.	

Related task

"Creating the initial 10.0 runtime environment" on page 65

Related concept

"Upgrade the Master Data Engine database to 10.0" on page 66

Running the database upgrade

Use the madconfig utility and your 10.0 engine instance to upgrade your database.

Before you begin

Confirm that you have reviewed and completed any tasks from these topics:

- Conducted the pre-upgrade tasks
- Created your initial engine runtime environment
- Reviewed the "Upgrade the Master Data Engine database to 10.0" topic
- Complete the database upgrade worksheet

Also verify that you have unset the `MAD_ODBCLIB` environment variable.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the applicable command:

   Microsoft Windows: `madconfig upgrade_instance`

   IBM AIX, Linux, or Solaris: `madconfig.sh upgrade_instance`
3. For each prompt, review the information and type the corresponding value that you defined in database upgrade worksheet. Press Enter. To apply the default value, press Enter without typing a value.
4. In the output, confirm that `BUILD SUCCESSFUL` appears.

Related reference

"Database upgrade worksheet" on page 68

Related tasks

"Conducting the pre-upgrade tasks" on page 64

"Creating the initial 10.0 runtime environment" on page 65

Conducting the Master Data Engine post-upgrade tasks

After you have completed your upgrade, there are certain tasks you must conduct.

Procedure

1. Start the 10.0 engine instance.
2. Confirm that the 10.0 engine instance is running.
3. Evaluate the next post-upgrade steps by determining whether any of these subtasks apply. Conduct each applicable task now:
   a. If the upgraded solution is to include integration with an external corporate LDAP directory server, you must configure the corresponding `com.initiate.server.ldap.cfg` file. The `com.initiate.server.ldap.cfg` file is located in the instance `conf` directory. For example:

      Microsoft Windows: *MAD_HOMEDIR*`\inst\mpinet_`*name*`\conf`

      IBM AIX, Linux, or Solaris: `/`*MAD_HOMEDIR*`/inst/mpinet_`*name*`/conf`

      where *MAD_HOMEDIR* is the full path to the directory created for the associated runtime instances (for example, `prod` or `qa`), and *name* is the engine instance name.

      Within the database, the upgrade process preserves any accounts for an existing external corporate LDAP directory server.

b. Copy any custom environment variables and other settings created before the upgrade in the former runtime environment into the initial 10.0 runtime environment. In particular, transfer the customized values of variables in `engine.properties` and `ldap.properties` files to the relevant `com.initiate.server.*.cfg` configuration file.

c. Create additional 10.0 engine instances or engine runtime environments to match the architecture implemented before the upgrade.

   In addition, customize the engine instances according to the previous sub-steps in this step (Step 3 on page 70 in this procedure) as applicable. Start these engine instances and then confirm that they are running.

4. Work with the IBM Services team to verify the accuracy of the conversion and perform the necessary re-derivation (redvd) of data, new weight generation, and possibly threshold adjustment.

   Important: If implementation-defined segments apply, they must be updated before re-deriving data or generating weights.

Related tasks

"Starting an engine instance from the Microsoft Windows Control Panel" on page 56

"Starting an engine instance with the madconfig utility" on page 56

"Starting an engine instance with its batch or script file" on page 57

"Using the madconfig utility to confirm that an engine instance is running" on page 58

"Using the madconfig utility to confirm that an engine instance is running" on page 58

"Using the MPINET protocol to confirm that an engine instance is running" on page 58

Related concepts

Appendix A, "LDAP Directory Server for the Master Data Engine," on page 197

"Starting and stopping your instances" on page 56

Related task

Chapter 5, "Installing the Master Data Engine," on page 41

Chapter 10, "Configuring Master Data Engine environment variables," on page 87

Chapter 8. Entity managers

The entity manager is the logic within the Master Data Engine that controls when comparison takes place after member data is derived. You can configure your engine to use an embedded entity manager or a stand-alone (external) entity manager. Before starting your installation, you might find it helpful to understand the entity manager process.

Entities and entity types

An entity represents the logical relationship among two or more member records. Each unique relationship has a corresponding entity type such as identity or household. For example, 10 records that describe the same person is an identity entity. Three records that describe the same household is a household entity. An entity type is the basis for any algorithm configuration. In creating an algorithm configuration through IBM Initiate Workbench, you create its associated entity type first. For related instructions, see the "Algorithms" and "Configuration editor" chapters in the *IBM Initiate Workbench User's Guide*.

Entity management

Entity management is the process by which record comparison, data linking, and task creation occur. The management settings affect how entity type comparisons are conducted after member data is derived. There are two primary methods:

- **Synchronous management.** The management for a synchronous entity type happens immediately after a member record is updated. The update is done through a put transaction. Before the transaction occurs, the target member record is cross-matched against multiple candidates. As a result, synchronous management occurs based on built-in logic and without any entity management processes. It is unaffected by external processes, input queues, and any polling interval and work unit settings for an entity type.
- **Asynchronous management.** In contrast, the management for an asynchronous entity type happens based on member records that appear in an entity manager input queue (that is, the `mpi_entique_xx` table, where *xx* is the applicable `entType`, or entity type, code). The put transaction creates records for the queue; an asynchronous process queries the input queue and completes the cross match for the target member record. The data is stored in the database, and comparison and linkage and task management are performed periodically. If you use asynchronous entity management, you also have the option of further controlling when entity management occurs by priority.
  - **Priority management**: There are three ways in which you can override your normal asynchronous entity management process. Priority entity management manages the process in one of three ways: by source priority, by set priority, and by specialty queue.
    - With entity management by source priority, you set a default entity priority for definitional sources that controls the order in which the sources are processed by the entity manager. The lower the source priority, the higher the entity management processing priority. For example, a source with a default entity priority of 1 always has entity management run before other sources. Source priority entity management is disabled by setting all of your sources with the same value (the default value is 100). See the *IBM Initiate Workbench User's Guide* for more details.

- For entity management by set priority, you set the entity management priority at the time of the member write (this option overrides any source entity management priority setting). For example, you can set a lower priority for batch-loaded members. The set priority is used with the member put interactions. See the *IBM Initiate Master Data Service SDK Reference for Java and Web Services* for details.
- The third option is by specialty queue. This option allocates stand-alone entity managers for specific queue processing. For example, you might want to create a stand-alone entity manager that processes only low priority records. This option is accomplished by adding a wrkOwner variable in the `com.initiate.server.entity.cfg` file. For example, `wrkOwner=special` where "special" is the name of the stand-alone entity manager. After adding the wrkOwner variable, you then run an SQL query to start the entity management process. For example:

```
UPDATE mpi_entique_id SET wrkOwner = 'special'
WHERE wrkOwner = 'system'
```

Within the IBM Initiate Workbench Configuration editor, you set the "Asynchronous", "Uses an input queue" and "Priority" properties to determine whether an entity type is processed synchronously or asynchronously. For related instructions, see the "Configuration editor" chapter in the *IBM Initiate Workbench User's Guide*.

- Asynchronous
  - `true` for asynchronous processing.
  - `false` for synchronous processing (default).
- Uses an input queue
  - `true` for asynchronous processing.
  - `false` for synchronous processing.
- Priority
  - by Source - an integer 1 - 32767 in the Default Entity Priority field.
  - by Set - defined at the API interaction level (an integer 1 - 32767); not in IBM Initiate Workbench.
  - by Specialty Queue - defined in the `com.initiate.server.entity.cfg` file and run by using an SQL query; not in IBM Initiate Workbench.

For entity management, it is possible to implement one of the scenarios described in this table.

Table 16. Entity management scenarios per given engine instance

Scenario	Entity type configuration	Description
Built-in entity management	Synchronous	Occurs automatically after a member record is updated. See the "Synchronous management" description. All entity types configured for synchronous processing*.
Embedded entity manager	Asynchronous	Runs as a thread within the instance with one thread per asynchronous entity type configured. The engine instance is configured to use the embedded entity manager. See the "Asynchronous management" description.

Table 16. Entity management scenarios per given engine instance (continued)

Scenario	Entity type configuration	Description
Stand-alone entity managers	Asynchronous	Runs as a separate instance to manage a single entity type. For example, if the instance identity and household entity types are configured for asynchronous processing, you might create corresponding stand-alone entity managers for each. One process for the identity entity and one for the household entity. See the "Asynchronous management" description.
Combination	Synchronous and asynchronous	The entity types are managed as follows: • Synchronous configuration uses the built-in logic. • Asynchronous configuration without any corresponding stand-alone entity manager running. Embedded manager. • Asynchronous configuration with a corresponding stand-alone entity manager running. Both the embedded and its stand-alone manager.
No management	Asynchronous	All entity types configured for asynchronous processing. Any configured stand-alone entity managers are not running. For example, it might be cost⌂prohibitive for large installations with hundreds of millions of records to manage entities. Instead, they depend on soft entities for data matching. With soft entities, every record that scores *n* or above for a given search represents the same person or thing. The system does not assign an Enterprise ID to represent a soft entity.
Priority entity management	Priority by source	Runs in the defined entity priority order. The lower the priority number, the higher the entity management priority.
Priority entity management	Priority by set	Runs when the associated member put interaction is executed for the specified member. If the source associated with the member has a Default Entity Priority definition, the member put interaction overrides the default.
Priority entity management	Priority by specialty queue	Requires a stand-alone entity manager. Runs when the SQL command is started.

*In this case, the entity management settings in the `com.initiate.server.system.cfg` configuration file are ignored.

The entity manager is a multi-threaded process and depends upon a "queue manager" to dictate the entity management process.

Related reference

"Stand-alone entity manager worksheet" on page 16

"Master Data Engine instance worksheet" on page 6

Related task

"Creating stand-alone entity managers" on page 46

Entity manager queue management

The entity manager is a multi-threaded process and depends upon a "queue manager" to dictate the entity management process.

A multi-threaded process means that on a 16-CPU-core box, you can have one entity manager with 16 threads, as opposed to 16 separate entity managers.

At the heart of entity and relationship management is what is called the entIque record. When a new member or a change to an existing member is processed by the Master Data Engine, an entIque record is created. The entIque record is consumed by both the entity linker and relationship linker to trigger potential work by the entity manager.

The entity manager retrieves entIque records from the database and sends them to an internal queue (or input queue). From this queue, the records are dispatched to multiple worker threads that simultaneously process each entity management event. A worker thread executes multiple pieces of logic by passing the data from one logic process (entity linking) to another (relationship linking). The number of worker threads used is configurable. If you have eight threads configured, eight entIque records are processed at one time.

Each entity type you have implemented has an entity process. For example, an implementation that uses identity and household entity types has two entity manager processes.

Queue management parameters are stored in the `com.initiate.server.queue.cfg` file.

Related reference

"Entity manager configuration parameters"

Entity manager configuration parameters

Configuration files are stored in different location depending on whether you use a stand-alone or an embedded entity manager.

Stand-alone entity manager configuration parameters are stored `.cfg` files located in the *entitymanager_instance_homeDir*`/inst/`*instance_name*`/conf` directory.

If you employ an embedded entity manager, the associated `.cfg` files are found in the *MAD_HOMEDIR*`/inst/`*instance_name*`/conf` directory.

The entity manager queue manager configuration file is `com.initiate.server.queue.cfg`. The parameters contained in this file are listed in this table.

Table 17. Entity manager queue parameters

Parameter	Description	Default and examples
entityTypes	A comma delimited list of the entity types that this queue manager processes. A wildcard character of "*" can be used to indicate all entity types. The character cannot be mixed with additional characters to configure subsets of entity types.	Default: * Valid examples: • id • id,hh,provider • * Invalid examples: • *,id • id*
pollSeconds	The number of seconds the queue manager waits after it polls the mpi_entique_xx table and finds no entries or the internal queue is full.	Default: 10 Valid examples: • 5 • 10 Invalid examples: • 0 • -5
workUnit	The row count that is pulled from the mpi_entique_xx table and inserted into the internal queue. This value should be less than the maxQueueDepth parameter.	Default: 500 Valid examples: • 500 • 5000 Invalid examples: • 0 • -500
workerThreads	The number of threads per entity type that are used to process work in parallel. This number can be based on the number of CPU cores available on the server. Increasing this number does not increase performance if CPU cores are not available and can decrease performance.	Default: 1 Valid examples: • 1 • 5 Invalid examples: • 0 • -5
maxQueueDepth	The maximum record size of the internal queue used by the worker threads. This number must always be larger than the workUnit and based on the number of queue managers, memory on the server, and database subsystem performance.	Default: 5000 Valid examples: • 5000 • 15000 Invalid examples: • 0 • -5000

Table 17. Entity manager queue parameters (continued)

Parameter	Description	Default and examples
inputWrkOwner	The name of the wrkOwner in the mpi_entique_xx table that this queue manager uses to pull records. This parameter must not be the same as the queueWrkOwner or errorWrkOwner.	Default: system Valid examples: system Invalid examples: error
queueWrkOwner	The name used for the wrkOwner in mpi_entique_xx table for the records this queue manager owns. This parameter should be unique for each queue manager that is running. This parameter must not be the same as the inputWrkOwner or errorWrkOwner.	Default: queue Valid examples: • queue • queue1 • queue2
errorWrkOwner	The name used for the wrkOwner in the mpi_entique_xx table if a record that is being processed has errored out and errorDequeueEnabled is set to true.	Default: error Valid examples: • error • error1 • error2
errorMaxRetryWaitSeconds	The number of seconds the queue manager waits before retrying a database operation that exceeds the errorRetryAttempts value. This value only comes into play on queue manger operations involving an mpi_entique_xx that cannot be de-queue or ignored. The minimum value for this parameter is 30 seconds.	Default: 60 Valid examples: • 30 • 90 Invalid examples: • 15
errorRetryWaitSeconds	The number of seconds the queue manager waits between an operation that returned an error and has not exceeded the errorRetryAttempts value.	Default: 5 Valid examples: • 5 • 10 • 0
errorRetryAttempts	The number of times the queue manager retries an operation before it takes one of these actions: • Sets the wrkOwner for the offending mpi_entique_xx record to the errorWrkOwner. • Deletes the offending mpi_entique_xx record. • Waits for the errorMaxRetryWaitSeconds value for a critical mpi_entique_xx database operation.	Default: 10 Valid examples: • 5 • 10 • 0

Table 17. Entity manager queue parameters (continued)

Parameter	Description	Default and examples
errorDequeueEnabled	This true or false setting determines the action that is taken when a recoverable error exceeds the errorRetryAttempts parameter. If this value is set to true, the mpi_entique_xx record has the wrkOwner set to the errorWrkOwner. If the parameters is false, the record is deleted from the mpi_entique_xx table.	Default: false Valid examples: • true • false
workPriorityEnabled	This true or false setting determines the order in which records from mpi_entique_xx are processed. If this value is true, the work priority is used to determined how the records are processed. If the value is false, the oldest records are processed first.	Default: true Valid examples: • true • false
maxWorkPriority	The maximum work priority record the queue manager processes. The higher the number, the lower the work priority. This setting is only valid if workPriorityEnabled is set to true.	Default: 0 Valid examples: • 0 • 50 Invalid examples: • -50
entityLinkerEnabled	This true or false setting determines whether entity linking is enabled for this queue manager.	Default: true Valid examples: • true • false
relationshipLinkerEnabled	This true or false setting determines whether relationship linking is enabled for this queue manager.	Default: false Valid examples: • true • false

Individual implementations can have their own configuration parameters that do not apply to the queue manager.

Chapter 9. Event notification

Event notification allows external sources to receive messages for a subset of internal events that are generated within the IBM Initiate Master Data Service. These messages are generated in an XML format and are published to a JMS queue.

The external source must implement a supported JMS queue provider and set up queues to which the IBM Initiate Master Data Service publishes and to which the consuming source subscribes. The event notification logic uses three components: event manager, event work manager, and event handler.

- Event manager - manages the internal event queue and passes events to the event work manager. An embedded event manager is configured during the creation of the Master Data Engine instance. You also have the option of creating a stand-alone instance by using the madconfig utility create_eventmgr command.
- Event work manager - creates external event messages based on the internal events and passes the message to the event handler. The event work manager is configured in IBM Initiate Workbench. The event work manager configuration consists of a list of events to be published (this list is stored in the mpi_evtlist table) and a list of destinations to be published to (stored in the mpi_evtdest table). The list of published events also has configuration parameters available to identify what attributes are included in the message (based on the selected composite view) and additional flags for message creation options.
- Event handler - publishes the event message to the consumer (JMS Queue is the supported message service). Use the madconfig utility configure_instance_eventhandler command to configure an event handler for an embedded event manager. Use the madconfig utility configure_eventmgr_eventhandler command to configure an event handler for a stand-alone event manager. The event handler is deployed to an embedded event manager using the madconfig utility deploy_instance_eventhandler command. The event handler is deployed to a stand-alone event manager using the madconfig utility deploy_eventmgr_eventhandler command.

Supported event types

The event types supported for event notification are preconfigured in your IBM Initiate Master Data Service data model. The supported event types table lists the preconfigured events.

Table 18. Supported event types for event notification

Event	Description
Member Created	The member created event occurs when either a create member interaction is executed or as part of a member put interaction which also includes a member update. A member created event occurs only once during the lifetime of a member.

Table 18. Supported event types for event notification (continued)

Event	Description
Member Updated	The member updated event occurs when member data is initially created. A member update then subsequently each time the member attribute data is updated, new attributes are added, existing attributes are deleted, or the member status is changed. The change might or might not result in any change to the persisted member data depending on engine configuration options.
Member Merged	The member merged event occurs when a member merge interaction is processed by the engine. A member might be merged multiple times because of the unmerge capability. The event represents a change to the member status and not the member attributes data.
Member UnMerged	The member unmerge event occurs when a previously merged member is the target of a member unmerge interaction. A member might be unmerged multiple times because of the merge capability. The event represents a change to the member status and not the member attribute data.
Member Deleted	The member deleted event occurs when a member delete interaction is processed. A member might be deleted multiple times due to the member undelete interaction. The event represents a change to the member status and not the member attribute data.
Member UnDeleted	The member undeleted event occurs when a member undelete interaction is processed. A member might be undeleted multiple times due to the member delete interaction. The event represents a change to the member status and not the member attribute data.
Member Linked	The member linked event occurs when entity management links a member to a new entity. A member might link multiple times during its life span. The event represents a change in the member entity participation but not in member status or attribute data.
Member Unlinked	The member unlinked event occurs when entity management unlinks a member from an existing entity. A member might unlink multiple times during its life span. The event represents a change in the member entity participation but not in the member status or attribute data.
Member Dropped	The member dropped event occurs when a member drop interaction is executed. There is a limited amount of data in a member dropped event because the member data has been permanently removed from the system. A member dropped does not represent a change in status or attribute data since the data is removed.
Entity Created	The entity created event occurs when a new entity is created. New entity creation can occur when a new member is added to the system or when a member unlinks from an existing entity to form a new entity. The event represents a change in the number of entities for a given entity type participating in the system.

Table 18. Supported event types for event notification (continued)

Event	Description
Entity Updated	The entity updated event occurs when a change is made to the composition of an existing entity or when the composite view for the entity changes. These composition changes manifest themselves as members joining or leaving an entity. The composite view changes are represented by changes to attribute values that participate in the view. It is possible for a single member update to trigger this event twice, once for the attribute changes and once for the entity composition changes.
Entity Deleted	The entity deleted event occurs when an entity ceases to exist in the system. These changes can occur when all members of an existing entity join another existing entity or when a member is dropped and that member was a singleton entity. The event represents a change in the number of entities for a given entity type participating in the system.

Composite views

All events can have either a member-based composite view or an entity-based composite view defined. The decision on which composite view to use is dependent on what data the external (downstream) system requires. In most cases, an entity-based composite view is the most appropriate. This view gives a robust picture of an entity from multiple sources. The event type (member or entity) does not play a factor into which composite view to use.

For example, one useful configuration is to subscribe to member update events and use an entity composite view. This configuration allows the external system to have visibility of all changes to attribute data within an entity and receive an event with the entity data. Additional options can be configured, such as "Enable change detection" (isChanged field in the mpi_evtlist table) that allows the message to be sent only if the data in the composite view has been changed since the last message was created.

Some events will never send composite view data in a message regardless of whether a composite view is configured. A member drop, for example, has no data to form a composite view and therefore cannot send a composite view. An entity delete, like a member drop, has no reference to create a composite view because the entity no longer exists.

Related reference

"Event notification worksheets" on page 19

Enabling event notification

Use this procedure to enable event notification.

Before you begin

Before beginning this task, complete the event notification worksheets.

Procedure

1. Determine whether to use an embedded or stand-alone event manager.
2. Create your event manager.
   a. On the command line, go to the Master Data Engine installation directory (MAD_ROOTDIR). Then go to the scripts directory. From this directory, you use the madconfig utility.
   b. If you are implementing an embedded event manager, respond y to the embedded event manager prompt during your Master Data Engine instance creation.
   c. For a stand-alone event manager, after you have created your engine instance, type this command: madconfig create_eventmgr
   d. Complete the prompts using the applicable event manager worksheet for guidance.
   e. In the output, confirm that BUILD SUCCESSFUL displays.
3. Configure the event work manager using IBM Initiate Workbench Hub Configuration > Events. Instructions can be found in the *IBM Initiate Workbench User's Guide*.
4. Configure the event handler.
   a. On the command line, go to the Master Data Engine installation directory (MAD_ROOTDIR). Then go to the scripts directory.
   b. For an embedded event manager, type this command: madconfig configure_instance_eventhander
   c. For a stand-alone event manager, type this command: madconfig configure_eventmgr_eventhandler
   d. Complete the prompts using the applicable event notification event handler worksheet.
   e. In the output, confirm that BUILD SUCCESSFUL displays.
   f. In the com.initiate.server.event.cfg file, edit the #publish.environment properties for your JNDI environment. Remove the hash mark (#) from each property that you edit. These properties are required:

      ```
      #publish.environment.initialcontextfactory.key=
      #publish.environment.initialcontextfactory.value=
      #publish.environment.jndiprovider.key=
      #publish.environment.jndiprovider.value=
      ```
5. Also in the com.initiate.server.event.cfg file, if passwords are required, you must modify the #publish.environment.password.3=mypassword property based on the password scheme. Passwords are optional depending on how your MQ Server is configured and what type of user authentication is required in order to connect to those destinations (queues).
   a. If the user password is plain text, use: publish.environment.password.*x=textpassword*.
   b. If the password is encrypted using the madpwd2 utility, use: publish.environment.password2.*x=encryptedpassword*.
   c. If the pass word is encrypted using the madpwd3 utility, use: publish.environment.password3.*x=encryptedpassword*. If you used a password encrypted by the madpwd3 utility, these properties are also required:

      ```
      publish.environment.aeskeyfile.x=mykeyfile.txt
      publish.environment.aesivfile.x=myivfile.txt
      publish.environment.aesprovider.x=
      ```
6. Deploy the event handler. For embedded, type this command: madconfig deploy_instance_eventhandler. For stand-alone, type this command: madconfig

deploy_eventmgr_eventhandler. The client must provide the required JMS libraries needed to connect and send messages to MQ Sever.

Results

Related reference

"madconfig utility" on page 102

"Event notification worksheets" on page 19

"Master Data Engine instance worksheet" on page 6

Sample com.initiate.server.event.cfg configuration file

The com.initiate.server.event.cfg file contains the parameters that provide for event notification from the Master Data Engine to external systems.

On an engine host, the location of the com.initiate.server.event.cfg file depends on whether you use an embedded or stand-alone instance. This file is stored in the engine instance conf directory for an embedded event manager. For example:

Microsoft Windows: *MAD_HOMEDIR*\inst*mpinet_name*\conf

IBM AIX, Linux, or Solaris: /*MAD_HOMEDIR*/inst/*mpievtmgr_name*/conf

For a stand-alone instance, the file is stored in the event manager instance conf directory. For example:

Microsoft Windows: *event_manager_HOMEDIR*\inst*mpievtmgr_name*\conf

IBM AIX, Linux, or Solaris: /*event_manager_HOMEDIR*/inst/*mpievtmgr_name*/conf

The first part of the configuration file contains the settings defined during event manager configuration. For example:

```
queueWrkOwner=76877787
pollSeconds=10
workUnit=500
maxQueueDepth=5000
workerThreads=1
ctxPoolSize=1
```

The next part of the configuration file is used for creating the JNDI environment for the InitialContext (lines that begin with #publish.environment). The properties in this section must be manually modified for your configuration. Some of these properties are required, while others (mainly the lines that start with publish.environment.key) are optional and vary for each MQ Server implementation. In this example, the required properties are identified with an (R) and optional properties are identified with an (O). They are not identified in the configuration file.

```
#publish.environment.initialcontextfactory.key=java.naming.factory.initial (R)
#publish.environment.initialcontextfactory.value=com.sun.jndi.ldap.LdapCtxFactory (R)
#publish.environment.jndiprovider.key=java.naming.provider.url (R)
#publish.environment.jndiprovider.value=ldap://localhost:389/o=IBM,c=US (R)
#publish.environment.key.1=java.naming.security.authentication (O)
#publish.environment.value.1=simple (O)
#publish.environment.key.2=java.naming.security.principal (O)
#publish.environment.value.2=cn=Manager,o=IBM,c=US (O)
```

```
#publish.environment.key.3=java.naming.security.credentials (0)
#publish.environment.password.3=mypassword (0)
#publish.environment.key.4= (0)
#publish.environment.value.4= (0)
```

All passwords are optional (for example, `publish.environment.password` and `publish.destination.password`) depending on how your MQ Server is configured and what type of user authentication is required in order to connect to those destinations (queues). If passwords are required, the `#publish.environment.password.3=mypassword` property must be modified based on the password scheme used. If the user password is plain text, use: `publish.environment.password.`*`x=textpassword`*.

If the password is encrypted using the madpwd2 utility, use: `publish.environment.password2.`*`x=encryptedpassword`*.

If the pass word is encrypted using the madpwd3 utility, use: `publish.environment.password3.`*`x=encryptedpassword`*. If you used a password encrypted by the madpwd3 utility, these properties are also required:

```
publish.environment.aeskeyfile.x=mykeyfile.txt
publish.environment.aesivfile.x=myivfile.txt
publish.environment.aesprovider.x=
```

The final part of the configuration file contains the settings defined during event handler configuration. For example:

```
publish.jms.connection.factory.name=cn=CF2
publish.destination.name.1=Test Queue 1 publish.destination.user.1=userq1
publish.destination.password.1=userq1pwd

publish.destination.name.2=Test Queue 2
publish.destination.user.2=userq2
publish.destination.password.2=userq2pwd
```

Related reference

"madpwd2 utility" on page 128

"madpwd3 utility" on page 128Appendix I, "AES encryption," on page 243

Chapter 10. Configuring Master Data Engine environment variables

After installation, you might find that you have additional environment variables to configure. Some variables apply for all installations while others are for IBM AIX, Linux, or Solaris installations only.

About this task

Use this procedure to add any additional environment variables needed for your implementation.

Procedure

1. From the runtime instance `conf` directory, open the relevant `com.initiate.server.*.cfg` file in a text editor. For example:

   Microsoft Windows: *MAD_HOMEDIR*`\inst\`*type_name*`\conf`

   IBM AIX, Linux, or Solaris: `/`*MAD_HOMEDIR*`/inst/`*type_name*`/conf`

   where:

   - *MAD_HOMEDIR* is the full path to the directory created for the associated runtime instances (for example, `prod` or `qa`).
   - *type_* is pre-determined prefix that is one of these options:
     - `mpinet_` for an engine instance.
     - `mpientmgr_` for an entity manager.
     - `mpildap_` for an IBM Initiate LDAP directory server.
   - *name* is the instance name, which is specified at instance creation time.
2. In the `com.initiate.server.*.cfg` file, add one or more environment variables that apply. Include each variable on its own line by using the *variable=value* syntax. For example: `MAD_SRVNO=10`
3. Save the changes to the file.
4. Restart the instance.

Related tasks

"Starting an engine instance from the Microsoft Windows Control Panel" on page 56

"Starting an engine instance with the madconfig utility" on page 56

"Starting an engine instance with its batch or script file" on page 57

"Starting an entity manager instance through the Microsoft Windows Control Panel" on page 60

"Starting an entity manager instance with the madconfig utility" on page 60

"Starting an entity manager instance with its batch or script file" on page 61

Master Data Engine environment variables

There are a number of environment variables and diagnostic logging options that can be set for the Master Data Engine. Some variables are set during installation and instance configuration. Others, however, are manually set.

Depending on the usage scenario, there are a few places where you might set applicable environment variables, including within:

- The applicable configuration files for the engine instance and other instance types (for example, stand-alone instances of entity managers and IBM Initiate LDAP directory servers). When an instance is started, its environment is set based on the properties defined in the `com.initiate.server.*.cfg` configuration files stored in the *instanceName*`\inst\mpinet_`*instanceName*`\conf` directory.

  You can set all Master Data Engine environment variables in an instance configuration file except for the three variables listed here. These three variables are set in the instance `wrapper.conf` file:

  – `MAD_ROOTDIR`
  – `MAD_HOMEDIR`
  – `MAD_INSTDIR`
- An *envar*`.bat` file. Running this file from a command line sets the appropriate environment in that shell. For example, a user might need to connect to ODBC mode and run engine utilities before the Master Data Engine runtime environment is up and running.
- The user profile. With the associated user logged in to a shell, that shell is set with the appropriate environment. The method for setting environment variables depends on the operating system.

Unless otherwise specified, the variable options are: `1` or `0` or `Y` or `N`. For additional variables not covered in this topic, see the related links listed in this topic.

Attention: Many of the variables in this table are not set by default. To create and set any of these variables, do so within the `com.initiate.server.system.cfg` file.

Table 19. Master Data Engine environment variables

Environment	Description	Defaults
`MAD_ROOTDIR`	Sets the IBM Initiate Master Data Service root directory. Set within `wrapper.conf`.	*installdir/product* (for example, `C:\Program Files\IBM\Initiate\Engine`)
	The following variables manage log information.	
`mad.log.name`	IBM Initiate® Master Data Service® application log name. Set within the `com.initiate.server.system.cfg` file. You can add a directory as a prefix to the log name to indicate that the log is to be written to a non-default directory. For example: `mad.log.name=C:\\myLogs\\mpinet_hub10081-%s.mlg`	*processname*-yyyymmdd-hhmmss.mlg (for example, `mpinet_prod10-20050824-162248.mlg`)
`NativeLog`	Runtime log level. Activate by setting `NativeLog` to `DEBUG`, `TRACE`, `WARN`, and so on, within `log4j.xml`.	Default = `INFO`
`ConversionPattern` parameters	Controls the parameters that are presented in log files.	Default = `%d{HH:mm:ss} [%5.5i] %-5.5p %32.32c: %m%n`

Table 19. Master Data Engine environment variables (continued)

Environment	Description	Defaults
AuditLog	Runtime log auditing. Set within log4j.xml.	Default = ALL
PerformanceLog	Enables performance logging for the engine. Set within log4j.xml. To enable performance logging, within log4j.xml set PerformanceLog from OFF to ALL.	Default = OFF
AlgorithmLog	Enables algorithm logging for the engine. Set within log4j.xml. To enable algorithm logging, within log4j.xml set AlgorithmLog from OFF to ALL.	Default = OFF
SqlLog	Runtime log dbsql statements. Set within log4j.xml.	Default = OFF
TimerLog	Runtime log timing. Set within log4j.xml.	Default = OFF
MAD_SMTLIST	Comma-separated list of language (SMT) codes. Set within com.initiate.server.system.cfg.	Optional; default = en_US for U.S. English
	The following variables manage database information.	
MAD_CONNSTr	ODBC connection string. The data source name is found in the ODBC.ini file on UNIX and Linux systems and in the System DSN on Microsoft Windows. Set within com.initiate.server.system.cfg.	(Invalid ODBC value characters = []{}(),;?*=!@\ per the spec) "DSN=data source name; UID=user name;PWD=user password;" PWD = a plain text password PWD2 = an encrypted password For encrypted passwords, you must use the following: PWD2=user password. ODBCCINI is used by the Data Direct drivers.

Table 19. Master Data Engine environment variables (continued)

Environment	Description	Defaults

MAD_CONNSTR	MPINET connection string if another process is using this string; for example, MBTS.
	
	Microsoft Windows Server requires the pipe delimiter to be escaped (by using the symbol ^) when setting the MAD_CONNSTR environment variable within a cmd.exe shell. Ex:
	`set MAD_CONNSTR=<hostname>^|<port>`
	
	UNIX and Linux system example:
	`export MAD_CONNSTR="hostname|port"`
	
	Set within com.initiate.server.system.cfg.
		hostname|portno [|protocol] (protocol = BINARY or STRING)

MAD_DBTYPE	Native database type. Set within com.initiate.server.system.cfg.	(oracle | db2 | mssqlu)
MAD_DBSERVER	Native database server. MAD_DBSERVER and MAD_DBNAME are only set under the following circumstances: **SQL Server**: MAD_DBSERVER is required if the SQL Server database is on a different server than the Master Data Engine. **Oracle**: MAD_DBNAME is required if the Oracle database is on a different server. If you must set MAD_DBSERVER and MAD_DBNAME, do so within com.initiate.server.system.cfg.	Optional.
MAD_DBNAME	Native database name. Passed to the database bulk-load utilities by the madhubload, madhubunload, madload, madunload, and maddbx utilities. MAD_DBSERVER and MAD_DBNAME are only set under the following circumstances: **SQL Server**: MAD_DBSERVER is required if the SQL Server database is on a different sever than the Master Data Engine. **Oracle**: MAD_DBNAME is required if the Oracle database is on a different server. If you must set MAD_DBSERVER and MAD_DBNAME, do so within com.initiate.server.system.cfg.	Database name (oracle sid).
	The following MAD_DB* variables are not set by default. These variables are typically set for stand-alone tools. You typically do not have to set these variables within the Master Data Engine configuration files.	

Table 19. Master Data Engine environment variables (continued)

Environment	Description	Defaults
MAD_DBUSER	Native database user ID. Passed to the database bulk-load utilities by the madhubload, madhubunload, madload, madunload, and maddbx utilities.	Database user login
MAD_DBPASS	Native database password. Passed to the database bulk-load utilities by the madhubload, madhubunload, madload, madunload, and maddbx utilities.	Database user password
MAD_DBSETUP	Setup command executed on each application server. Optional for supported databases	
MAD_DBXTEST	When turned on, this variable causes the engine to test long-held ODBC connections before use. 0 = False, 1 = True	Optional; default = 1 (true)
	The following variables support customer variability in IBM Initiate Master Data Service software functionality.	
MAD_HOMEDIR	Instance-specific home directory. Set within `wrapper.conf`.	Default: *instancename*/*installdir*/prod10
MAD_INSTDIR	Sets the Master Data Engine instance directory. Set within `wrapper.conf`.	
MAD_SRVNO	MPINET server ID. This setting is a unique eight-digit identifier based on the system clock at the time of instance creation. Set within `com.initiate.server.system.cfg`.	Optional, default = system time

Table 19. Master Data Engine environment variables (continued)

Environment	Description	Defaults
MAD_CTXLIB	Set within com.initiate.server.system.cfg. Specifies one of the following connection modes: **ODBC**. Communicate with the Master Data Engine database directly without going through another server layer. This mode is used for batch operations and upgrades. Only bulk cross match and server-based diagnostic utilities work in ODBC mode. **MPINET**. Wire protocol for communication between client applications and the Master Data Engine. Essentially, a client application initiates contact with the Master Data Engine. The engine in turn performs the business logic and communicates with the database on behalf of the client. This mode is used for programs that conduct individual transactions rather than batch operations. When the system was based fully on C or C++, clients were able to switch between the ODBC and MPINET connection modes. However, the Java and .Net APIs support only the MPINET connection mode.	(ODBC versus MPINET)
MAD_UNLFSR	unl field separator replacement. Set within com.initiate.server.system.cfg.	Default = '!'
MAD_DICTIMEOUT	In a multi-server environment, this setting is the maximum amount of time (in seconds) the Master Data Engine waits to check whether the dictionary segments in the database are different from the ones in its memory cache. If set to 0 (zero), the Master Data Engine checks every interaction. If set to -1, it never checks for change. Set within com.initiate.server.system.cfg.	Default = 300 seconds
MAD_OBJCODE	Default object code used by the Master Data Engine command-line utilities (multiple codes are supported). Options include disc, mem, and aud. Set within com.initiate.server.system.cfg.	NONE

Table 19. Master Data Engine environment variables (continued)

Environment	Description	Defaults
MAD_TABPFX	Table prefix used by the Master Data Engine command-line utilities The table creation, load, and unload utilities apply the variable when set; however, not all parts of the engine apply it. For example, JDBC-based access does not use this variable. To access a database without ANSI ownership prefixed to the table name, you must use RDBMS synonyms or some similar facility. It is possible to use the MAD_DBPFX environment variable to work with database backup images for which the object owner password is unknown. Set within com.initiate.server.system.cfg.	NONE
MAD_TABSFX	Default table suffix used by the Master Data Engine command-line utilities for the entity-specific tables. For example, _id, _pr, _og, and so on. Set within com.initiate.server.system.cfg.	NONE
MAD_UNLDIR	Default directory used by the Master Data Engine command-line utilities for the unload files. Set within com.initiate.server.system.cfg.	NONE
MAD_DDLFILE	Data Definition file used by the Master Data Engine command-line utilities. Set within com.initiate.server.system.cfg.	If not set, the madhub* and madent* utilities make assumptions as to the location of the ddlfile and stofile. Both the MAD_DDLFILEand MAD_STOFILE variables are still valid for use with the maddbx utility.
MAD_STOFILE	Storage definition file used by the Master Data Engine command-line utilities. These files are dependent on the database server being used. Set within com.initiate.server.system.cfg.	

Table 19. Master Data Engine environment variables (continued)

Environment	Description	Defaults
MAD_CALLBACKLIB	Enables the Master Data Engine to call custom handlers (callback and event notification functionality) The Master Data Engine instance must be restarted after modifying this variable. Set within com.initiate.server.system.cfg.	Microsoft Windows environment, set: MAD_CALLBACKLIB= mpicbjava.dll UNIX and Linux environment, set: libMPICBJAVA.so .NET (Microsoft Windows only), set: mpicbdotnet.dll
MAD_GNRCONFIG	If your hub algorithm configuration uses GNRMETA, set this variable before attempting to run any of the engine utilities (for example, mpxdata). This variable must point to the GNR nameworks.config file contained within the GNM INSTALL/data directory. For example, MAD_GNRCONFIG = C:\Program Files\IBM\GNM\data\ nameworks.config	NONE
net-listener	Presence of this value indicates that the Master Data Engine instance is providing MPINET TCP/IP communication behavior. This means that the engine listens for the standard MPINET protocol over port 16000. This value is initially set by answering 'y' to the "MPINet over TCP/IP" prompt during the madconfig instance creation process. Set within com.initiate.server.features.cfg. To unset this value, remove the entry.	NONE
net-servlet	Presence of this value indicates that the engine instance is providing MPINET over HTTP services. This value is initially set by answering 'y' to the "MPINet over HTTP" prompt during the madconfig instance creation process. Set within com.initiate.server.features.cfg. To unset this value, remove the entry. In some government environments, this method is required. See the *IBM Initiate Master Data Service Security Technical Implementation Guide (STIG)* for details. You can configure an instance to run over both TCP/IP and HTTP.	NONE

Table 19. Master Data Engine environment variables (continued)

Environment	Description	Defaults
MAD_IPVERSION	This variable is used to control the type of address family to use when constructing a client connection or server socket. This variable is honored by the Message Broker Suiteand command-line utilities that connect to an engine instance through MPINET. Values are: not set, 4, or 6. The madconfig utility adds the default setting of 4 to the MAD_CONFNAME file. Changes to this setting must be done manually to MAD_CONFNAME. For Message Broker Suite processes that act in both client and server mode (for example, Query Broker), MAD_IPVERSION is shared. This means that both the client and server portions of the instance are in the specified IP version (you cannot have an instance in IPv4 and one in IPv6). Engine utilities can have more flexibility. For more information about this variable setting for Brokers, see the *IBM Initiate Master Data Service Message Broker Suite Reference* .	Engine default = NONE. The IP version is detected by the engine, so no configuration is required by the user. Broker default = 4 (IPv4)
	The following variables are not used by the Master Data Engine, however they are used by the Message Broker Suite and IBM Initiate Web Reports.	
MAD_SECLIB	Security library; indicates that SSL is implemented for client communication.	SSL, optional
MAD_CONFNAME	Host level configuration file; setup through Microsoft Windows “My Computer > Properties”	
MAD_SSLFIPSMODE	This variable is used to enable FIPS mode within the Message Broker Suite and command-line utilities which can communicate over MPINET and SSL.	The default value is 0 (not enabled).
MAD_ENCODING	Sets the internationalization behavior. MAD_ENCODING does not appear in the configuration files by default. To set it to a value other than latin1, do so within the com.initiate.server.system.cfg configuration file. Log files created by the Master Data Engine are in ASCII encoding. Code points not encompassed by ASCII are in the standard Unicode form of U+XXXX.	Options are: latin1, utf8, utf16 Default: latin1

Related concepts

"SSL security" on page 191

Chapter 11, "Diagnostic logging," on page 97

"ConversionPattern format specification" on page 99

Appendix C, "Master Data Engine storage files (stofiles)," on page 217

Related reference

"Entity manager configuration parameters" on page 76

Chapter 11. Diagnostic logging

The Master Data Engine uses log4j to provide diagnostic information.

You can configure the contents of each diagnostic message to contain date and time stamps, program names, and other information that can be used to more easily identify the message. By putting this information about each line, you can use text searching utilities such as `grep` (IBM AIX, Linux, or Solaris), or `find` (Microsoft Windows) to help locate items in a large log.

The Master Data Engine processes use log4j exclusively, while the Message Broker Suite components use a combination of log4j and a native logging process. See the *IBM Initiate Master Data Service Message Broker Suite Reference* for details.

The contents of the logging topics are based on the assumptions that you are familiar with setting environment variables and understand valid file naming for the supported operating and file system in use.

Log file location and naming

By default, a program that produces log output writes the output file in the directory from which the program was invoked.

The location and name of the log file can be configured by setting these environment variables:

- `mad.log.name`

  Specifies the name of the log file.

  The name of the file can be any valid file name for the operating system on which the process is running. By convention, logs commonly include the file extension `.mlg`, but that is not a requirement. You can include any extension (for example, `.log` or `.txt`) or no extension at all.

  To write the logs to a location other than the default location, add a prefix to the relevant directory information to the setting for `mad.log.name`. For example:

  ```
  mad.log.name=C:\\myLogs\\mpinet_hub10081-%s.mlg
  ```

  When `mad.log.name` is set with a path and file name, the process writes to the file specified instead of the location where the program was invoked. The directory path must exist; the logging process does not create the directory.

  Also, include the format specifier `%s` so that the process adds a date stamp suffix to the file name in the format `YYYYMMDD-HHMMSS` based on date and time the file was first created. For example, with the `mad.log.name=MyLog-%s.mlg"` setting, a process that created its first log entry November 23, 2004 at 12:01:00 P.M. yields this log file:

  ```
  MyLog_20041123-120100.mlg
  ```

  Processes that run over into the next day, and daemon or service processes that run continuously, create new log files. The files are written upon the first log entry created on the new date.

Attention: To maintain pre-6.1 behavior with the date only and not the time stamp (for example, `/ibm/initiate/logs/MyLog_20041123.mlg`), specify a length setting in the `%s` format to include only the date portion. The length setting takes the form of a period (`.`) followed by a length limit (for example, 8). For example:

```
mad.log.name=MyLog-%.8s.mlg
```

Including `%.8s` yields the first 8 characters of the date-time stamp; `%.6s` yields only the year and month; and `%.4s` yields only the year. The valid values for the length are 1 through 15.

- `MAD_HOMEDIR`

Related concepts

"Master Data Engine environment variables" on page 87

"Master Data Engine directory structure - MAD_ROOTDIR and MAD_HOMEDIR" on page 32

Logging types

By setting values within `log4j.xml`, you can control the verbosity of log output. ERROR and INFO log messages are always written to the log.

The Master Data Engineuses a caching log writer, which is a mechanism for writing log events in to a circular memory buffer. The contents are written out to a log file when a triggering event occurs. Administrators can enable higher log levels for certain loggers without paying the performance and storage penalty of writing information to a file until the information is needed. The caching log writer is disabled by default. For details, see *IBM Initiate Master Data Service Software Operations Guide*.

You can set log settings on a temporary basis to run until the Master Data Engine is stopped from IBM Initiate Workbench. For example, you might need to increase verbosity to assist with troubleshooting. For instructions, see the *IBM Initiate Workbench User's Guide*.

This table lists the logging types that can be set.

Table 20. Logging types

Log Setting	Description
`NativeLog`, set to `DEBUG`	Produces low-level diagnostics used internally to identify what was happening in the system before an error condition occurred. This option generates a large amount of output per activity and should be used only for short time periods.
`NativeLog`, set to `TRACE`	Produces a trace of activity as interactions flow through the system. This option is verbose and should be used only for short time periods.
`NativeLog`, set to `WARN`	Produces alerts that are not errors, but might need to be tracked.
`AuditLog`	Produces activity information and non-critical warnings. This option is often used when a new system is first implemented to monitor activity.

Table 20. Logging types (continued)

Log Setting	Description
`SqlLog`	Outputs the SQL that is sent by the Master Data Engine database layer to the RDBMS. This logging can help in diagnosing database-related issues. This option can produce large amounts of output depending on the activity.
`TimerLog`	Produces timings on certain operations to help identify where significant processing time is elapsing.

ConversionPattern format specification

Within `log4j.xml`, the `ConversionPattern` parameter controls the contents of each line of the output inside of the log file.

Some of the most common parameters are listed in this table.

Table 21. Log ConversionPattern format codes

Conversion character	Effect
`%c`	Inserts the category of the log message.
`%d`	Inserts the date and time. Depends on format supplied.
`%i`	Inserts the thread ID. Unlike the thread name (indicated by %t), this is the numeric ID of the thread. This parameter is particular to IBM Initiate® Master Data Service®, while the other parameters listed here are standard with log4j.
`%m`	Inserts the log message itself.
`%n`	Inserts the platform-dependent line separator.
`%p`	Inserts the priority of the log message.
`%s`	Inserts a full date and time stamp into the log file name. Include a period and number within the code to truncate the length of the date and time stamp. For example, to include only the date string in the filename, specify mad.log.name=MyLog-%.8s.mlg. Valid values for the length are 1 - 15.
`%t`	Inserts the associated thread name.

Related concept

For more information about the format codes, see "Suggested logging settings."

Related information

For more detail about the syntax of ConversionPattern, see: http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html

Suggested logging settings

The log4j log system is set up with a flexible architecture to allow for different implementation scenarios.

This list provides some useful setting combinations for diagnosing problems.

- Enable the caching log writer and configure it to record the detail around triggering events. For details, see the *IBM Initiate Master Data Service Software Operations Guide*.
- Use the `%s` stamp in the `mad.log.name` settings for long⌂running processes. Doing so makes it easier to purge or back up old logs when a new log is created every day. This setting is best suited for MPINET and Message Broker Suite logs.
- The `ConversionPattern` settings depend on how the software is implemented and how you use the logs after archiving the system on which they are created. For instance:
  - If log files are going to be moved from the directory where they were created for archiving, it is a good idea to make the file name something unique. This naming can be important when moving multiple instances to the same archive directory. Also, it is helpful to make the log name descriptive such that the name identifies the process logged (for example, `prodEngine%s.mlg`).
  - Both the date and time code (`%d`) and the thread code (`%i`) are important for most logs. The time shows how long various operations take, and the thread ID helps separate multiple interactions on different threads logged at the same time.
- If disk space is a concern and the solution is running smoothly to date, consider turning off `AuditLog`. However, the `AuditLog` setting provides for a quick view into system activity and an easy way to monitor load and usage patterns. You can turn off `TimeLog` for the Master Data Engine and interfaces, but turn it on when running bulk cross match (BXM) utilities. Log settings can generate a large volume of data. Use them only when your are trying to pinpoint a problem.

Chapter 12. Using the Master Data Engine utilities

There are a number of command-line utilities that are included with the IBM Initiate Master Data Service software.

Before you begin

If your implementation is FIPS-compliant, command-line utilities that communicate over SSL must be configured to meet FIPS requirements.

About this task

These utilities are primarily used by the implementation and IBM Software Support teams. Some of these utilities are also available as jobs in IBM Initiate Workbench.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `bin` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\bin`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/bin`
2. To view command-line usage for any utility, run the command applicable for your operating system. Where *utility* is the utility name.
   - **Microsoft Windows**: *utility*`.exe` (for example, `mpxdata.exe`
   - **IBM AIX, Linux, or Solaris**: *utility*`.bat` (for example, `mpxdata.bat`)
3. Run the utility according to the usage output or the information provided in the utility topics. While the information in the topics covers greater detail than the usage output, it is a good idea to supplement the topic information with the usage output.

   Related concept

   Appendix K, “FIPS compliance,” on page 255

madcode utility

The madcode utility displays strings in Soundex, METAPHONE, and IDENTAPHONE form.

Any number of white space delimited arguments can be given on the command line. The madcode utility is a diagnostic utility that can be used to determine if two or more different strings can convert to the same phonetic encoded form.

Table 22. madcode options

Option	Type	Description	Default
`name`	Any string	Specifies the input to be converted by the various phonetic algorithms	NONE

madconfig utility

The madconfig utility is one of the most used Master Data Engine utilities. One of the more common uses of the utility is to configure instances for the Master Data Engine, components of the Message Broker Suite, stand-alone LDAP directory servers, and stand-alone entity managers.

This utility is run from the engine installation *MAD_ROOTDIR*\scripts directory. This utility configures the registry (Microsoft Windows) and creates configuration files for IBM AIX, Linux, or Solaris, and Microsoft Windows.

Attention:

* Although you can use the madconfig utility to generate and validates weights, it is suggested that you do these tasks from IBM Initiate Workbench. For guidance, see the *IBM Initiate Workbench User's Guide*.

** Registering and deploying callback handlers can be done by using madconfig or through IBM Initiate Workbench. The suggested method is to use IBM Initiate Workbench. If you register your handlers by using madconfig, the IBM Initiate Workbench configuration project does not know about the handler, and the dictionary becomes out of sync when the Master Data Engine configuration is deployed from IBM Initiate Workbench. For more information, see the *IBM Initiate Workbench User's Guide*. When using the "deploy" or "undeploy" targets, make sure to respond No to the prompts "Unregister all existing Handlers" and "Register new Handlers."

Restriction: *** Do not run the support target without consulting the IBM Software Support for guidance. Running this target can require significant time, depending upon your configuration and environment.

Table 23. madconfig options

Options and targets	Description	Default
`-projecthelp`	Lists the valid options.	NONE
`-propertyfile`	Loads properties from file.	NONE
`-recordfile`	Record response properties to file. For details on using the `-recordfile` option, see the Creating an automated madconfig utility scripts task.	NONE
`bootstrap_instance`	Bootstraps a Master Data Engine instance database	NONE
`check_entitymgr`	Check an entity manager instance (IBM AIX, Linux, or Solaris operating systems only).	NONE
`check_eventmgr`	Check an event manager instance (IBM AIX, Linux, or Solaris operating systems only).	NONE
`check_instance`	Check a Master Data Engine instance (IBM AIX, Linux, or Solaris operating systems only).	NONE
`check_ldap`	Checks for a response from an LDAP server (IBM AIX, Linux, or Solaris operating systems only).	NONE

Table 23. madconfig options (continued)

Options and targets	Description	Default
`configure_eventmgr_eventhandler`	Configures an event handler for a stand-alone event manager instance. This must be configured for event notification processing.	NONE
`configure_instance_eventhandler`	Configures an event handler for an embedded event manager instance. This must be configured for event notification processing.	NONE
`create_hubconnector`	Creates an IBM InfoSphere™ Master Data Management Hub Connector instance.	NONE
`create_datasource`	Creates a data source.	NONE
`create_entitymgr`	Creates an entity manager instance.	NONE
`create_eventmgr`	Creates an event manager instance	NONE
`create_instance`	Creates an engine instance. After executing, the utility walks you through the setup.	NONE
`create_ldap`	Creates a stand-alone instance of the IBM Initiate LDAP directory server. After executing, the utility walks you through the creation process.	NONE
`deploy_entitymgr_handlers`**	Deploys an entity manager handler. When prompted to "Unregister all existing Handlers" and "Register new Handlers," your response should always be "no."	NONE
`deploy_eventmgr_eventhandler`	Deploys an event handler for a stand-alone event manager instance.	NONE
`deploy_instance_eventhandler`	Deploys an event handler for an embedded event manager instance.	NONE
`deploy_instance_handlers`**	Deploys a handler for a Master Data Engine instance. When prompted to "Unregister all existing Handlers" and "Register new Handlers," your response should always be "no."	NONE
`enable_gnr`	This option is used to enable and configure the use of IBM InfoSphere Global Name Recognition (GNR) with the Master Data Engine. This option runs the `mpi_gnrconfig.sql` file and creates new entries in mpi_libhead, mpi_gnefunc, and mpi_bktxgen database tables. Use this target if you are configuring an algorithm to use the GNRMETA bucket generation function. You must restart your engine after running this option.	NONE

Table 23. madconfig options (continued)

Options and targets	Description	Default
`disable_gnr`	This option removes reference to GNR from the mpi_libhead, mpi_gnefunc, and mpi_bktxgen database tables. It does not remove any references to GNRMETA in your algorithm; you must edit your algorithm in IBM Initiate Workbench. You must restart your engine after running this option.	
`generate_perfrpt`	Takes the results from the Performance Logging Manager process and generates a Performance Log Report. See `start_perflogmgr` and `stop_perflogmgr` options in this topic. Additional detail is provided in the *IBM Initiate Master Data Service Software Operations Guide*.	NONE

Table 23. madconfig options (continued)

Options and targets	Description	Default
`generate_rocinp`	Generates the usamp, dsamp, and msamp files used by the Threshold Calculator for its calculations. This process can be run from the Generate Threshold Calculator Input Files job in IBM Initiate Workbench. Performance and optional parameters found in IBM Initiate Workbench can be modified from the command line by using the following ant properties listed here. The equivalent IBM Initiate Workbench property follows in parentheses (). Performance tuning parameters: `mad.rocinputs.threads` (Number of threads) `mad.rocinputs.num.parts.1` (Number of comparison bucket partitions) `mad.rocinputs.num.parts.2` (Number of random pairs bucket partitions) `mad.rocinputs.num.parts.max` (Maximum number of input and output partitions) `mad.rocinputs.upair.count` (Number of random pairs to generate) `mad.rocinputs.report.records` (Interval for reporting processed records) `mad.rocinputs.max.bucket.size` (Maximum bucket set size) `mad.rocinputs.min.weight` (Minimum weight for writing item records) `mad.rocinputs.num.parts.mem` (Number of member partitions) Optional parameters: `mad.rocinputs.cmpmode` (Comparison mode) `mad.rocinputs.use.all.attrs` (Skip last step because of too few attributes - true or false)	NONE

Table 23. madconfig options (continued)

Options and targets	Description	Default
`generate_weights`*	Runs the weight generation process (this process can be performed through IBM Initiate Workbench). If you do not want weight smoothing to take place during the generation process, make sure that you set the `mad.wgtgen.smooth` property to false. For example, `-Dmad.wgtgen.smooth=false`. The default setting for this property is true. If `mad.wgtgen.smooth` is set to true, then a `\smoothed` sub-directory is created. The directory contains smoothed usamp (unmatched pairs), msamp (matched pairs), and final binary and text report files. This structure maintains the original, unsmoothed, files that you can compare against the smoother files.	
`launch_etl`	Launches a CloverETL graph.	NONE
`list_datasources`	Lists existing database data sources.	NONE
`migrate_datasource`	Migrates a data source from one installation of the same version (X.X.X) to another. On Microsoft Windows operating systems, the data source is moved. On IBM AIX, Linux, or Solaris operating systems the data source is copied.	
`migrate_entitymgr`	Migrates a stand-alone entity manager (mpientmgr) instance from one installation of the same version (X.X.X) to another. **Important:** You must migrate your data source (by using `migrate_datasource`) before migrating your stand-alone entity manager.	
`migrate_eventmgr`	Migrates an event manager instance from one Master Data Engine installation to another installation of the same version.	
`migrate_instance`	Migrates an engine (mpinet) instance from one installation of the same version (X.X.X) to another. **Important:** You must migrate your data source (by using `migrate_datasource`) before migrating your engine instance.	
`migrate_ldap`	Migrates a stand-alone LDAP server (mpildap) instance from one installation of the same version (X.X.X) to another. **Important:** You must migrate your data source (by using `migrate_datasource`) before migrating your stand-alone LDAP server.	
`ping_instance`	Pings a Master Data Engine instance listener. You can also use the MPINET protocol to ping an instance; the MPINET option is useful for programmatic checks of instance status.	NONE

Table 23. madconfig options (continued)

Options and targets	Description	Default
`pingdb_instance`	Pings a Master Data Engine instance database via listener. You can also use the MPINET protocol to verify database availability; the MPINET option is useful for programmatic checks of database status.	NONE
`register_dotnet_assemblies`**	Registers IBM .NET assembly files for use by Master Data Engine instances that are using the .NET callbacks dll (`mpicbdotnet.dll`). This target is implicitly called during engine installation. However, if you are using the .NET callback dll, and have a version of .NET Framework earlier than 2.0, you must use this target to re-register your .NET assemblies post-engine-installation after upgrading to .NET Framework 2.0.	NONE
`register_handlers`**	Registers a handler. While this option remains, all handlers should be registered and unregistered through IBM Initiate Workbench.	NONE
`remove_hubconnector`	Removes an IBM InfoSphere Master Data Management Hub Connector instance.	NONE
`remove_datasource`	Removes an existing database data source	NONE
`remove_entitymgr`	Removes an entity manager instance.	NONE
`remove_eventmgr`	Removes an event manager instance.	
`remove_instance`	Removes a Master Data Engine instance.	NONE
`remove_ldap`	Remove a stand-alone instance of the IBM Initiate LDAP directory server.	NONE
`run_jobset`	Executes a job set that was initially created in IBM Initiate Workbench. If IBM Initiate Workbench is running on a different computer than the hub engine, you must copy the job set XML file from the IBM Initiate® Workbench project to the hub server. It is located at *`workbench_workspace`*`\`*`project_name`*`\jobTemplates`. If the job set includes the Deploy Hub Configuration job, the contents of the entire project must be copied to the server on which the Hub engine is running.	NONE
`run_mpitxm`	Executes the mpitxm utility. For process details, see the mpitxm utility documentation.	NONE
`run_rellinker`	Executes the relationship linker utility, which creates relationship linkages in bulk fashion.	NONE
`show_handlers`**	Shows all registered handlers for a Master Data Engine instance.	
`start_hubconnector`	Starts an IBM InfoSphere Master Data Management Hub Connector instance.	NONE

Table 23. madconfig options (continued)

Options and targets	Description	Default
`start_entitymgr`	Command to start a stand-alone instance for an entity manager; Microsoft Windows alternative is starting through Microsoft Windows Service.	NONE
`start_eventmgr`	Starts an event manager instance.	NONE
`start_instance`	Command to start a Master Data Engine instance; Microsoft Windows alternative is starting through a Microsoft Windows Service.	NONE
`start_ldap`	Command to start either an embedded or a stand-alone instance of the IBM Initiate LDAP directory server; Microsoft Windows alternative is starting through Microsoft Windows Service.	NONE
`start_perflogmgr`	Command to start the Performance Logging Manager process. Additional detail is provided in the *IBM Initiate Master Data Service Software Operations Guide*.	NONE
`stop_hubconnector`	Stops an IBM InfoSphere Master Data Management Hub Connector instance.	NONE
`stop_entitymgr`	Stops an entity manager instance.	NONE
`stop_eventmgr`	Stops an event manager instance.	NONE
`stop_instance`	Command to stop a Master Data Engine instance; Microsoft Windows alternative is stopping through Microsoft Windows Service.	NONE
`stop_ldap`	Command to stop either an embedded or a stand-alone instance of the IBM Initiate LDAP directory server; Microsoft Windows alternative is starting through Microsoft Windows Service.	NONE
`stop_perflogmgr`	Command to stop the Performance Logging Manager process. Additional detail is provided in the *IBM Initiate Master Data Service Software Operations Guide*.	NONE

Table 23. madconfig options (continued)

Options and targets	Description	Default
`support`***	Gathers hub metrics for IBM Software Support or for sizing. Queries are run against the database; the metrics are written to the instance-specified support directory in a single compressed file named after the run date. For example, if run on February 25 2010, the file is: `$MAD_HOMEDIR/support/201002025.zip`. Within the compressed file, madconfig writes a pipe-delimited `.unl` file with the result for each query. **Metrics gathered**. The `support` target gathers several metrics, including: definitional source count member count entity type count member type count attributes defined count member segment count row counts for all member segments bucket count comparison string count bucket role count comparison role count interactions audited count user count group count large bucket count (optional) large entity count (optional)	NONE
`test_datasource`	Tests a database data source.	NONE
`undeploy_entitymgr_handlers`**	Deactivates an entity manager handler. When prompted to "Unregister all existing Handlers" and "Register new Handlers," your response should always be "no."	
`undeploy_eventmgr_eventhandler`	Deactivates the event handler for an event manager instance.	
`undeploy_instance_eventhandler`	Deactivates an event handler for a Master Data Engine instance.	

Table 23. madconfig options (continued)

Options and targets	Description	Default
`undeploy_instance_handlers**`	Deactivates a Master Data Engine instance handler. When prompted to "Unregister all existing Handlers" and "Register new Handlers," your response should always be "no."	
`unregister_handlers**`	Unregisters a previously registered handler. While this option remains, all handlers should be registered and unregistered through IBM Initiate Workbench.	
`upgrade_entitymgr`	Upgrade an entity manager instance.	
`upgrade_eventmgr`	Upgrades an event manager instance.	
`upgrade_instance`	Upgrades a Master Data Engine instance.	NONE
`upgrade_ldap`	Upgrades an IBM Initiate LDAP Server instance.	
`validate_weights*`	Executes a validation process against the weights generated.	
`version`	Shows the version of the engine or broker.	

Related tasks

"Creating an automated madconfig utility script" on page 49

"Using ping requests to monitor Master Data Engine and database availability" on page 59

maddbx utility

The maddbx utility allows for database operations to retrieve information about a database, create or delete tables and indexes, along with several other database-related operations.

Table 24. maddbx options

Option	Type	Description	Default
`-numTabs`		Number of tables	NONE
`-tabList`		Table name (or names) for operation. Use `ALL` for all tables, or specify individual tables by name (for example, `mpi_memstat`).	NONE
`-colList`	tabName	Columns for table 'tabName'	NONE
`-idxList`	tabName	Indexes for table 'tabName'	NONE
`-numcols`	tabName	Number of columns for table 'tabName'	NONE
`-numidxs`	tabName	Number of indexes for table 'tabName'	NONE
`-numrows`	tabName	Number of rows of specified tables	NONE
`-crTable`	tabName	Creates table 'tabName'	NONE
`-drTable`	tabName	Deletes table 'tabName'	NONE

Table 24. maddbx options (continued)

Option	Type	Description	Default
-crIndex	idxName	Creates index 'idxName'	NONE
-drIndex	idxName	Deletes index 'idxName'	NONE
-crTabIdx	tabName	Creates all indexes for table 'tabName'	NONE
-drTabIdx	tabName	Deletes all indexes for table 'tabName'	NONE
-optimize	tabName	Optimize specified tables or ALL tables. This command is used as a starting point for table optimization. It is a generic command based on default optimizer settings. The effectiveness of the command could vary depending on the optimizer settings in place for that implementation. If table optimization is not achieved by running this command, contact the responsible database administrator to update statistics in a manner tailored to the optimizer settings in place for that implementation.	NONE
-loadunl	tabName	Loads table 'tabName' from .unl file*	NONE
-unload	tabName	Unloads table 'tabName to a .unl file	NONE
-rebuild	tabName	Rebuilds (creates the table and indexes) for table 'tabName	NONE
-truncate	tabName	Truncates (deletes all data) from table 'tabName'	NONE
-sqlexec	sqlFile	Executes commands from an SQL file.	NONE
These next arguments are additional arguments for –loadunl:			
-onepass		Implies –truncate, -loaddata, -index	
-truncate		Truncate tables only	
-loaddata		Load data information table (or tables) only	
-index		Index table (or tables) only	
-useint		Use internal loader instead of the database utility. For improved performance, use the internal loader.	
-remote		Database is remote	
-maxErrs		Maximum number of "per-table" errors allowed before ending	

Table 24. maddbx options (continued)

Option	Type	Description	Default
-commitSize		Number of records processed between commits	
-otherArgs		Additional arguments to database utility	
-rootDir		MAD_ROOTDIR; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-homeDir		MAD_HOMEDIR; location of a Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-connStr		MAD_CONNSTR; ODBC connection string	
-objCode		MAD_OBJCODE; restrict command to table or index belonging to this object. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbtype		MAD_DBTYPE; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbServer		MAD_DBSERVER; native database server. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbname		MAD_DBNAME; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbuser		MAD_DBUSER; native database user ID. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbpass		MAD_DBPASS; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native

Table 24. maddbx options (continued)

Option	Type	Description	Default
-ddlfile		MAD_DDLFILE; data definition file. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional based on command
-stofile		MAD_STOFILE; storage definition file. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional
-unldir		MAD_UNLDIR; load or unload .unl directory. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional based on command
-tabPfx		MAD_TABPFX; table and index name prefix. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional
-tabSfx		MAD_TABSFX; table and index name suffix. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional
-encoding		MAD_ENCODING; encoding of .unl files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1
-noexec		Show SQL statements only; no execution is performed	NONE
-help		List help information	NONE
-version		List version number	NONE

Related concept

"Master Data Engine environment variables" on page 87

madentcreate utility

The madentcreate utility creates database tables and indexes to support a new entity type (enttype).

Table 25. madentcreate options

Option	Type	Description	Default
-entType	entType	Specifies the name of the entity on which to perform the operation	NONE
-rptPfx		List the entity name prefixes	NONE
-noInit		Do not initialize after creation	NONE

Table 25. madentcreate options (continued)

Option	Type	Description	Default
-noExec		Show SQL statements only; no execution is performed	NONE
-verbose		Show progress information	NONE
-help		List help information	NONE
-version		List version number	NONE
-rootDir		MAD_ROOTDIR; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-homeDir		MAD_HOMEDIR; location of Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-connStr		MAD_CONNSTR; ODBC connection string. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbType		MAD_DBTYPE; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbServer		MAD_DBSERVER; native database server	Optional if not using native
-dbName		MAD_DBNAME; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbUser		MAD_DBUSER; native database user ID. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbPass		MAD_DBPASS; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-encoding		MAD_ENCODING; encoding of .unl files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1

Related concept

"Master Data Engine environment variables" on page 87

madentdrop utility

The madentdrop utility drops database tables and indexes of an entity type (enttype).

Table 26. madentdrop options

Option	Type	Description	Default
-entType	entType	Specifies the name of the entity on which to perform the operation	NONE
-rptPfx		List the entity name prefixes	NONE
-noExec		Show SQL statements only; no execution is performed	NONE
-verbose		Show progress information	NONE
-help		List help information	NONE
-version		List version number	NONE
-rootDir		MAD_ROOTDIR; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-homeDir		MAD_HOMEDIR; location of Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-connStr		MAD_CONNSTR; ODBC connection string. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbType		MAD_DBTYPE; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbServer		MAD_DBSERVER; native database server. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbName		MAD_DBNAME; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbUser		MAD_DBUSER; native database user ID.The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native

Table 26. madentdrop options (continued)

Option	Type	Description	Default
-dbPass		MAD_DBPASS; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-encoding		MAD_ENCODING; encoding of .unl files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1

Related concept

"Master Data Engine environment variables" on page 87

madentload utility

The madentload utility loads the entity tables from .unl files.

Table 27. madentload options

Option	Type	Description	Default
-entType	entName	Specifies the name of the entity on which to perform the operation	NONE
-rptPfx		List the entity name prefixes	NONE
-tabList	tabName	Table name (or names) to load. Use ALL for all tables, or specify individual tables by name (for example, mpi_memstat).	NONE
-unlDir	dirName	Location of the .unl files	NONE
-onepass		Implies –truncate, -loaddata, and –index options	NONE
-truncate		Truncates (deletes all data) from the table (or tables) before loading	NONE
-loaddata		Loads the data from the .unl file	NONE
-index		Indexes the table (or tables)	NONE
-useint		Use internal database loader instead of DBMS-specific utility. For improved performance, use the internal loader.	NONE
-remote		The DBMS is remote from this server	NONE
-maxErrs	N	Maximum errors before ending	NONE
-commitSize	N	Number of records to process before issuing a database commit operation	NONE
-otherArgs	args	Additional arguments for native DBMS utilities	NONE
-noExec		Show SQL statements only; no execution is performed	NONE

Table 27. madentload options (continued)

Option	Type	Description	Default
-verbose		Show progress information	NONE
-help		List help information	NONE
-version		List version number	NONE
-rootDir		MAD_ROOTDIR; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-homeDir		MAD_HOMEDIR; location of Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-connStr		MAD_CONNSTR; ODBC connection string. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbType		MAD_DBTYPE; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbServer		MAD_DBSERVER; native database server. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbName		MAD_DBNAME; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbUser		MAD_DBUSER; native database user ID. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbPass		MAD_DBPASS; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-encoding		MAD_ENCODING; encoding of .unl files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1

Related concept

"Master Data Engine environment variables" on page 87

madentreset utility

The madentreset utility truncates an entity.

Table 28. madentreset options

Option	Type	Description	Default
-entType	entType	Specifies the name of the entity on which to perform the operation	NONE
-rptPfx		List the entity name prefixes	NONE
-noExec		Show SQL statements only; no execution is performed	NONE
-verbose		Show progress information	NONE
-help		List help information	NONE
-version		List version number	NONE
-rootDir		MAD_ROOTDIR; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-homeDir		MAD_HOMEDIR; location of Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-connStr		MAD_CONNSTR; ODBC connection string. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbType		MAD_DBTYPE; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbServer		MAD_DBSERVER; native database server. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbName		MAD_DBNAME; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbUser		MAD_DBUSER; native database user ID. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native

Table 28. madentreset options (continued)

Option	Type	Description	Default
-dbPass		MAD_DBPASS; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-encoding		MAD_ENCODING; encoding of .unl files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1

Related concept

"Master Data Engine environment variables" on page 87

madentunload utility

The madentunload utility unloads one or more entity tables into .unl files.

Table 29. madentunload options

Option	Type	Description	Default
-entType	entType	Defines the suffix for entity table names	NONE
-rptPfx		Defines report prefix for entity table names	
-tabList		Defines the list of entity tables on which to operate. Use ALL for all tables, or specify individual tables by name (for example, mpi_memstat).	ALL | tabName(s)
-unlDir		Location of .unl files	
-commitSize		Defines the number of records processed between commits	
-noExec		Show SQL statements only, no execution is performed	
-verbose		Displays progress information	
-help		Displays usage information	
-version		Displays version information	
-rootDir		MAD_ROOTDIR; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	

Table 29. madentunload options (continued)

Option	Type	Description	Default
-homeDir		MAD_HOMEDIR; location of Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-connStr		MAD_CONNSTR; ODBC connection string. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbType		MAD_DBTYPE; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbServer		MAD_DBSERVER; native database server. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbName		MAD_DBNAME; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbUser		MAD_DBUSER; native database user ID. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbPass		MAD_DBPASS; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-encoding		MAD_ENCODING; encoding of .unl files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1

Related concept

"Master Data Engine environment variables" on page 87

madhubcreate utility

The madhubcreate utility creates database tables and indexes, and initializes base tables to support a new IBM Initiate Master Data Service software installation.

Table 30. madhubcreate options

Option	Type	Description	Default
`-noInit`		Do not initialize after creation	NONE
`-noExec`		Show SQL statements only; no execution is performed	NONE
`-verbose`		Show progress information	NONE
`-help`		List help information	NONE
`-version`		List version number	NONE
`-rootDir`		`MAD_ROOTDIR`; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
`-homeDir`		`MAD_HOMEDIR`; location of Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
`-connStr`		`MAD_CONNSTR`; ODBC connection string. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
`-dbType`		`MAD_DBTYPE`; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
`-dbServer`		`MAD_DBSERVER`; native database server. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
`-dbName`		`MAD_DBNAME`; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
`-dbUser`		`MAD_DBUSER`; native database user ID. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
`-dbPass`		`MAD_DBPASS`; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native

Table 30. madhubcreate options (continued)

Option	Type	Description	Default
`-encoding`		`MAD_ENCODING`; encoding of `.unl` files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1

Related concept

"Master Data Engine environment variables" on page 87

madhubdrop utility

The madhubdrop utility drops the database tables and indexes of an IBM Initiate Master Data Service software installation.

Table 31. madhubdrop options

Option	Type	Description	Default
`-confirm`		Required to confirm execution of drop command	NONE
`-noExec`		Show SQL statements only; no execution is performed	NONE
`-verbose`		Show progress information	NONE
`-help`		List help information	NONE
`-version`		List version number	NONE
`-rootDir`		`MAD_ROOTDIR`; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
`-homeDir`		`MAD_HOMEDIR`; location of Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
`-connStr`		`MAD_CONNSTR`; ODBC connection string. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
`-dbType`		`MAD_DBTYPE`; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
`-dbServer`		`MAD_DBSERVER`; native database server. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native

Table 31. madhubdrop options (continued)

Option	Type	Description	Default
-dbName		MAD_DBNAME; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbUser		MAD_DBUSER; native database user ID. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbPass		MAD_DBPASS; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-encoding		MAD_ENCODING; encoding of .unl files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1

Related concept

"Master Data Engine environment variables" on page 87

madhubload utility

The madhubload utility loads one or more IBM Initiate Master Data Service tables from .unl files.

Table 32. madhubload options

Option	Type	Description	Default	
-objCode	objCode	Object code (DIC, MEM, AUD, REL, TAG, HST, ANL) for the data to be loaded	NONE	
-tabList	tabList	List of tables to be loaded. Use ALL for all tables, or specify individual tables by name (for example, mpi_memstat).	NONE	
-unlDir	dirName	Location of the .unl files	NONE	
-onepass		Implies –truncate, -loaddata, and –index options	NONE	
-truncate		Truncates (deletes all data) from the table (tables) before loading	NONE	
-loaddata		Loads the data from the .unl file	NONE	
-index		Indexes the table (tables)	NONE	

Table 32. madhubload options (continued)

Option	Type	Description	Default	
-useint		Use internal database loader (ODBC) instead of DBMS-specific bulk load utility. This option can be used when the bulk load utilities are not installed on the system where the Master Data Engine is installed. For improved performance, use the internal loader.	NONE	
-remote		The DBMS is remote from this server	NONE	
-maxErrs	N	Maximum errors before stopping	NONE	
-commitSize	N	Number of records to process before issuing a database commit operation	NONE	
-otherArgs	args	Additional arguments for native DBMS utilities	NONE	
-noExec		Show SQL statements only; no execution is performed	NONE	
-verbose		Show progress information	NONE	
-help		List help information	NONE	
-version		List version number	NONE	
-rootDir		MAD_ROOTDIR; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.		
-homeDir		MAD_HOMEDIR; location of Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.		
-connStr		MAD_CONNSTR; ODBC connection string. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.		
-dbType		MAD_DBTYPE; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.		
-dbServer		MAD_DBSERVER; native database server. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native	
-dbName		MAD_DBNAME; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native	
-dbUser		MAD_DBUSER; native database user ID. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native	

Table 32. madhubload options (continued)

Option	Type	Description	Default	
-dbPass		MAD_DBPASS; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native	
-encoding		MAD_ENCODING; encoding of .unl files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1	

Related concept

"Master Data Engine environment variables" on page 87

madhubreset utility

The madhubreset utility truncates a Master Data Engine instance.

Table 33. madhubreset options

Option	Type	Description	Default
-noExec		Show SQL statements only; no execution is performed	NONE
-verbose		Show progress information	NONE
-help		List help information	NONE
-version		List version number	NONE
-rootDir		MAD_ROOTDIR; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-homeDir		MAD_HOMEDIR; location of Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-connStr		MAD_CONNSTR; ODBC connection string. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbType		MAD_DBTYPE; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbServer		MAD_DBSERVER; native database server. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native

Table 33. madhubreset options (continued)

Option	Type	Description	Default
-dbName		MAD_DBNAME; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbUser		MAD_DBUSER; native database user ID. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbPass		MAD_DBPASS; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-encoding		MAD_ENCODING; encoding of .unl files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1

Related concept

"Master Data Engine environment variables" on page 87

madhubunload utility

The madhubunload utility unloads one or more core (non-entity) tables into .unl files.

Table 34. madhubunload options

Option	Type	Description	Default
-objCode		Object code (DIC, MEM, AUD, REL, TAG, HST, ANL) for the data to be loaded	
-tabList		List of tables to be loaded. Use ALL for all tables, or specify individual tables by name (for example, mpi_memstat).	
-unlDir		Identifies the location of .unl files	
-commitSize		Defines the number of records processed between commits	
-orderBy		You can sort .unl files by the column or columns specified in the orderBy parameter. If a column is specified that does not belong to a table that you are trying to unload, then the madhubunload utility fails.	
-noExec		Show SQL statements only, no execution is performed	

Table 34. madhubunload options (continued)

Option	Type	Description	Default
-verbose		Shows progress information	
-help		Shows usage information	
-version		Shows version information	
-rootDir		MAD_ROOTDIR; location of IBM Initiate Master Data Service software. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-homeDir		MAD_HOMEDIR; location of Master Data Engine instance. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-connStr		MAD_CONNSTR; ODBC connection string. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbType		MAD_DBTYPE; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	
-dbServer		MAD_DBSERVER; native database server. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbName		MAD_DBNAME; native database name. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbUser		MAD_DBUSER; native database user ID. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-dbPass		MAD_DBPASS; native database password. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	Optional if not using native
-encoding		MAD_ENCODING; encoding of .unl files; options are: UTF8, UTF16, or LATIN1. The environment variable setting for this option (shown in uppercase) can be used in place of the command-line options.	LATIN1

Related concept

"Master Data Engine environment variables" on page 87

madload utility

The madload utility loads a database table from a `.unl` file.

This utility uses ODBC access to load the file one record at a time. (See the madhubload utility `–useInt` option.)

Table 35. madload options

Option	Type	Description	Default
`-con`		`CONNSTR`; ODBC connection string	NONE
`-tabName`	tabName	Name of table to be loaded	NONE
`-inpFile`	filename	Name of `.unl` file to be loaded	NONE
`-encoding`		Encoding of `.unl` files; options are: UTF8, UTF16, or LATIN1	LATIN1

madpass utility

The madpass utility generates the encrypted form of a plain text password.

This utility takes the plain text password as the only input parameter.

madpwd2 utility

The madpwd2 utility takes a plain text password and encrypts it, enabling administrators to type encrypted text in configuration and profile files.

Table 36. madpwd2 options

Option	Type	Description	Default
`-e`	string	The plain-text version of the password.	NONE

madpwd3 utility

The madpwd3 utility takes a plain text password and encrypts it by using the Advanced Encryption Standard (AES) 128, 192, or 256-bit encryption method.

Restriction: Before running this utility, you must have an AES key and initialization vector (iv) generated. You can enter the key and iv from a file or directly at the command line.

Table 37. madpwd3 options

Option	Type	Description	Default
`-key`	AES key	If you are entering the key at the command line, this option is the name of `.dat` file created during key generation. Usage example: `madpwd3 -key 99CA56BDF62638567F456941650237AB`	

Table 37. madpwd3 options (continued)

Option	Type	Description	Default
`-keyfile`	filename	If you are entering the key from a file, this option is the full path and key filename. Usage example: `madpwd3 -keyfile c:\Program Files\IBM\Initiate\ Engine10.0.0.x\myaeskey.dat`	
`-iv`	IV key	If you are directly entering the iv at the command line, this option is the name of `.dat` file created during iv generation. Usage example: `madpwd -iv 7E892875A52C59A3B588306B13C31FBD`	
`-ivfile`	filename	If you are entering the iv from a file, this option is the full path and iv filename. Usage example: `madpwd3 -ivfile c:\Program Files\IBM\Initiate\ Engine10.0.0.x\myiv.dat`	
-in	text	The plain text version of the password.	

Related concept

Appendix I, "AES encryption," on page 243

madsql utility

The madsql utility runs an SQL statement or file of SQL statements through the Master Data Engine interface.

Before loading `.unl` files, be aware of any data encoding before setting options.

Table 38. madsql options

Option	Type	Description	Default
`-conn`		MAD_CONNSTR; ODBC connection string	
`-dsn`		ODBC dsn name	
`-uid`		ODBC user ID	
`-pwd`		ODBC user password	
`-quiet`		Prevents display of results from command	NONE
`-unload`		Generates the output in `.unl` format and enables the display of Unicode characters	NONE
`-in INPUTENCODING`		Specifies the encoding type (UTF8, UTF16, or LATIN1) of the input data	LATIN1
`-out OUTPUTENCODING`		Specifies the encoding type (UTF8, UTF16, or LATIN1) of the output data	LATIN1

Table 38. madsql options (continued)

Option	Type	Description	Default
`-sqlfile`	sqlFile	Name of a file containing a list of SQL commands to execute	NONE
`-sqlstmt`	sqlString	A single SQL command to execute	NONE

madunload utility

The madunload utility unloads database tables into `.unl` files.

Table 39. madunload options

Option	Type	Description	Default
`-con`		CONNSTR; ODBC connection string	
`-tabName`	tabName	List of table names to unload	
`-encoding`		Encoding of `.unl` files; options are UTF8, UTF16, or LATIN1	LATIN1
`-outFile`	fileName	Defines the output file name. If `outfile` is not specified, the unloaded table contents are displayed.	

mpidelete utility

The mpidelete utility enables command-line access to the MEMDELETE (member delete) interaction.

Restriction: When running the mpidelete utility, you must use an MPINET connection.

Table 40. mpidelete options

Option	Type	Description	Default
`-conn`		ODBC connection string	
`-usrLogin`		User ID	
`-usrPass`		User password	
`-rec`	recString	The member record to delete, where `sourceCode:recordID` identifies the source and record.	NONE

mpidrop utility

The mpidrop utility enables command-line access to the MEMDROP (member drop) interaction.

Restriction: When running the mpidrop utility, you must use an MPINET connection.

Table 41. mpidrop options

Option	Type	Description	Default
`-conn`		MPINET connection string	

Table 41. mpidrop options (continued)

Option	Type	Description	Default
-usrLogin		User ID	
-usrPass		User password	
-rec	recString	The member record to drop, where sourceCode:recordID identifies the source and record.	NONE

mpiengget utility

The mpiengget utility outputs the logging levels that you have set in real time.

Restriction: When running the mpiengget utility, you must use an MPINET connection.

Table 42. mpiengget options

Option	Type	Description	Default
-unl or -html		Outputs the logging levels to a computer screen or HTML	
-usrLogin		User ID	
-usrPass		User password	
-bgcolor		Use only if you elect to output the levels to HTML	Pink

mpimcomp utility

The mpimcomp utility enables command-line access to the MEMCOMPARE (member compare) interaction.

This utility can be run from the command line or from IBM Initiate Workbench, **Initiate menu** > **New Job Set**. See the *IBM Initiate Workbench User's Guide*. This command takes "entType memRecno1 memRecno2" as input text at the command line. It outputs the comparison information as text.

Restriction: When running the mpimcomp utility, you must use an MPINET connection. This command operates only in ODBC mode.

Table 43. mpimcomp options

Option	Type	Description	Default
-entType	memRecno1 memRecno2	Specifies the name of the entity on which to perform the operation	
-encoding			
-usrLogin		User ID	
-usrPass		User password	

mpimerge utility

The mpimerge utility enables command-line access to the MEMMERGE (member merge) interaction.

Restriction: When running the mpimerge utility, you must use an MPINET connection.

Table 44. mpimerge options

Option	Type	Description	Default
`-rec1`	recString	Record 1 (Survivor) where `sourceCode:recordID` identifies the source and record.	NONE
`-rec2`	recString	Record 2 (Obsolete) where `sourceCode:recordID` identifies the source and record.	NONE
`-conn`		MPINET connection string	
`-usrLogin`		User ID	
`-usrPass`		User password	

mpimshow utility

The mpimshow utility is a diagnostic utility used to dump the in-memory contents of a member identified by a `memRecno`. The entire contents of the member are dumped to stdout, including associated derived data.

This command takes "`memRecno`" as input text at the command line. The command outputs the detailed information for a member as text.

Restriction: This command operates only in ODBC mode.

Table 45. mpimshow options

Option	Type	Description	Default
`-html` or `-xml`		Outputs information to an HTML or XML file	
`-memRecno`		Identifies one or more member record numbers	

mpinetget utility

The mpinetget utility captures engine performance statistics without needing to shut down the engine.

In mpinetget output, the time shown is in milliseconds (ms), and these values indicate interaction totals:

- **Totexecs**, total number of interactions received by this hub.
- **Totgood**, number of the total interactions (Totexecs) that did not return an error code to the sender.

Before running mpinetget from the command line, make sure that your environment has MAD_CTXLIB set to MPINET (not ODBC) and points to the applicable engine with MAD_CONNSTR. Running mpinetget in ODBC mode yields inapplicable results based on an internal instance of the engine. In this case, the command creates the temporary instance, and all counts are `0`. In a multi-hub environment, run mpinetget against each hub, and then calculate the totals.

Table 46. mpinetget options

Option	Type	Description	Default
`-unl`		Outputs the information to the screen	
`-html`		Dump the output to HTML format	
`-report`		Dumps the output to text format	
`-usrLogin`		User ID	
`-usrPass`		User password	
`-bgcolor`		Use with HTML only	Pink

mpitxm utility

The mpitxm utility can be used to update members without having to shut down the Master Data Engine.

During normal operation, the IBM Initiate Master Data Service software performs various operations each time a member is updated. These operations include the update of derived data used in cross matching, and the cross match of the updated member against other candidate members. The Master Data Engine uses bulk utilities that can perform these updates for the entire member population at load time or during system conversions. The bulk utilities work quickly, but they require that the engine be quiescent. There are times when you might want the engine to perform the update of the derived data, or to re-cross match many members. This utility provides a way to update the members without having to shut down the engine.

This utility works by generating a pseudo-update on each member specified in the input to the program. No data is changed or inserted, but the engine is "tricked" into performing the described maintenance activities. Since this utility looks like any other IBM Initiate Master Data Service client application, you can run more than one instance of the utility at the same time, to increase throughput. See the mpitxm options tables on how running multiple instances might affect the load on the running system.

The mpitxm utility differs from other engine utilities in that it is launched by using the madconfig utility. Navigate to the Master Data Engine `MAD_ROOTDIR\scripts` directory. Use this command to start the process:

```
madconfig run_mpitxm
```

Restriction: The engine instance must be up and running before starting the mpitxm utility. If the engine is taken down during the run, mpitxm reports an error and stops processing.

Attention: This utility can generate a large amount of traffic on a production server. Be careful not to impact production users. Information about settings, such as bulk processing and wait time between updates, is provided in the mpitxm options table.

Because mpitxm operates quickly, it can monopolize a server (context pool) thread. If you have a low number of threads, your user interactions might queue up and have to wait to get in between the mpitxm bulk puts. Be sure to check this setting before running this utility in production. Use a low priority entity manager queue setting for mpitxm so that your real-time traffic is processed first.

Table 47. mpitxm options

madconfig prompt	**Description**	**Default**
Engine host name	The name of the host where the engine is installed. This name can be a symbolic name or IP address.	Local host name
Engine host port number	The TCP/IP port number on which the engine is listening for client connections.	Local host port number
Engine user name	Then engine user name.	
Password scheme	The password scheme used.	
User password	The user password.	
Log path	Specify the file name where the log output is stored. This defaults to the name `mpitxm.log` in the `MAD_ROOTDIR\scripts` directory.	`MAD_ROOTDIR\ scripts\ mpitxm.log`
Memrecno range	A "y" or "n" response indicates whether the utility processes memrecnos within a specified range.	
Minimum memrecno	If you entered "y" to using a memrecno range, you are prompted to specify the minimum memrecno for the utility to process.	
Maximum memrecno	If you entered "y" to using a memrecno range, you are prompted to specify the maximum memrecno for the utility to process. If a member is not found in the memrecno range, the utility produces a non-unrecoverable error in the log and continues processing.	
Memrecno file	If you respond "n" to using a memrecno range, the utility asks if you want to use a memrecno file. Specify the file name of a text file containing one memrecno per line. The utility generates errors for any memrecno not found and continues on with the next member in the list. See the file formats section later in this topic for more information.	
Memrecno file path	Enter the location of the memrecno file.	

Table 47. mpitxm options (continued)

madconfig prompt	Description	Default
srccode|memidnum file	If you respond "n" to using a memrecno file, the utility asks if you want to use a srccode|memidnum file. Specify the name of a text file containing a list of source code and memidnum combinations that identify the members to be updated. This option is the fastest because the utility does not read in the member first. Rather the utility generates an update based on the identifying information provided. The format of the file is shown in the file format section later in this topic, along with sample SQL for dumping this output.	
srccode|memidnum file path	Enter the location of the srccode|memidnum file.	
Derive the data only	A "y" or "n" response indicates whether data will be derived. If you do not use the derive data option, each updated member is re-cross matched. If you use asynchronous entity management (most systems use asynchronous), then the engine creates a queue record that triggers the entity manager (or multiple entity managers, ir applicable) to start the re-cross match process. The mpitxm process can produce these queue records much faster than the entity managers can consume them. The mpi_entique table (or tables if you have multiple entities), grow very quickly. Make sure that you have enough database space to contain the queue records. On systems that use optimized put mode, the mpitxm utility is able to support only the re-derivation of the member data. Since mpitxm does not change any attribute data, the optimized put setting prevents it from creating entity manager queue records.	

Table 47. mpitxm options (continued)

madconfig prompt	Description	Default
Bulk size	Specify the size of the batch of members to be updated in a single interaction. The mpitxm utility creates an audHead record for each Member Put interaction. It might also generate an audXmem row. On systems with a large number of members, consider how much space might be consumed by the audit data and whether you have enough room in the database before running the utility. By specifying a bulk size, you can lower the amount of growth in the audit tables. If you have a one-million-member table and set your bulk size to 10, you create only 100,000 audHead records. The tendency is to set the bulk size to a high number, but remember that each bulk put is a single database transaction. The members in each batch remain locked while the put is processed. If you set the bulk size too high, you can cause resource contention and timeouts for production users. Set the bulk size to a number of 30 or less. On slower systems, or systems with a high amount of user traffic, set it to 10. The final downside to a large batch setting is that the bulk interaction is an all-or-nothing transaction. If one member in the batch fails, the entire batch is rolled back and none are updated. See the logging section in this topic for information.	1
Progress report	This entry is used to specify how often the utility reports its progress. A setting of 100 reports after 100 members are read. This information can be used to restart the utility should it be interrupted for any reason. It is a good idea to make this setting a multiple of your bulk size, as you might report on a member that has been read from the input but not yet updated. If you set the reporting frequency, try to not set too low of a number for the progress report. The utility generates two lines of text in the log every time the reporting frequency count is reached. If you have a two-million-member database and you set the reporting frequency to 20, you have two million text rows in the log just for status. Try to set the number to a multiple of -bulk. If the bulk size is 10, set the reporting frequency to 10000. By using a multiple of -bulk, you can eliminate the possibility of reading a row and reporting it to the log before the put batch is full and written to the engine.	1000 members

Table 47. mpitxm options (continued)

madconfig prompt	Description	Default
Wait between updates	Specify the number of seconds to wait between each update. If a time is not specified, the updates occur without pause. This option can be used to spread the load on the entity manager to allow real incoming updates to be processed in between the pseudo-updates. This option can be used to slow down the queue generation. The mpitxm utility pauses for (x) seconds between each batch of records. If you have this set to 5 and bulk size set to 20, then the utility waits for 5 seconds after each batch of 20 members has been processed.	0
Entity manager work queue priority	Specify the entity manager work queue priority. The lower the number, the higher the priority of the item in the queue. Values are 1 - 32767. The default value is the lowest priority. If you have a low number of context pool threads, you can use a low priority setting to process your real-time traffic processed first.	

File formats:

The file specified by the memrecno file option should look like:

```
398392
309843
398293
.
.
.
```

Each field in the file needs to have a trailing pipe "|" delimiter. The utility complains about improperly formatted lines, or missing members, but continues processing. This is an example of the SQL used to generate the output into file form:

```
select s.srccode, h.memidnum from mpi_memhead h, mpi_srchead s
where s.srcrecno = h.srcrecno and h.memstat = 'A';
```

If you save this SQL into a file called `buildList.sql`, you can use madsql to generate a pipe delimited file for you.

```
madsql -unload -sqlfile buildList.sql > idNum.txt
```

You can add on to the "where" clause to pick a subset of the members needed for processing. Run this utility only against active members (memStat = 'A'). You might want to split large result sets into multiple input files for parallel processing.

Logging:

The application logs its activity and errors to a text file. The mpitxm utility uses the internal Java logging facilities which generate at least two lines of text for each item reported. If you use an existing log file for multiple runs of the utility, it appends new information to the end of the file. This is an example of what the output can look like:

```
Jun 7, 2010 3:15:11 PM com.initiatesystems.mpitxm.MpiTxm logSetup
Jun 7, 2010 3:15:12 PM com.initiatesystems.mpitxm.MpiTxm procRange
INFO: 50 completed.  Last memRecno processed: 1044
Jun 7, 2010 3:15:13 PM com.initiatesystems.mpitxm.MpiTxm procRange
INFO: 100 completed.  Last memRecno processed: 1094
Jun 7, 2010 3:15:14 PM com.initiatesystems.mpitxm.MpiTxm cleanup
INFO: Program finished.
```

After each 50 rows, an INFO line is printed to show the last member record read (or calculated in the example of a minimum and maximum range). You might also see WARNING lines that list errors or problems that were not fatal. Non-fatal errors can include missing memrecno values, or improper or blank lines in an input file. If a batch failed because one of the records could not be processed, then the log shows the starting and ending members of the batch to let you know that none of them were updated.

If you see SEVERE messages in the file that means that an error occurred where processing stops.

mpiunmrg utility

The mpiunmrg utility enables command-line access to the MEMUNMERGE (member unmerge) interaction.

Restriction: When running the mpiunmrg utility, you must use an MPINET connection.

Table 48. mpiunmrg options

Option	Type	Description	Default
`-rec`	recString	Obsolete record where `sourceCode:recordID` identifies the source and record	NONE
`-conn`		MPINET connection string	
`-usrLogin`		User ID	
`-usrPass`		User password	

mpxbchk utility

The mpxbchk utility is an optional diagnostic utility used to check bucket sizes during bulk cross match operations.

All options and flags are case independent; option values are not.

Table 49. mpxbchk options

Option	Type	Description	Default
`-entType`	Name	Entity type name	NONE
`-bxmInpDir`	dirName	`.bin` file input directory	NONE
`-bxmOutDir`	dirName	`.bin` file output directory	NONE
`-nthreads`	N	Number of threads	1
`-nBktParts`	N	Number of bucket partitions	1
`-maxbktsize`	N	Maximum bucket-set size	500

Table 49. mpxbchk options (continued)

Option	Type	Description	Default
-minBktTag	N	Minimum bucket tag to use (0 = any)	0
-maxBktTag	N	Maximum bucket tag to use (0 = any)	0

mpxcomp utility

The mpxcomp utility enables the comparison of records and is one of the processes used during bulk (BXM) and incremental cross matches (IXM).

There are four different situations in which you run the mpxcomp utility. You can run mpxcomp:

- During the initial stage of implementation to generate baseline comparison scores.
- During the "reiterate" step of the implementation process. After going through the entire set of implementation steps and analyzing your data results, you might determine that modifications to your algorithm and data dictionary are necessary. If so, you typically re-derive your data (by using the mpxdata, mpxfsdvd, or mpxredvd utilities) and run another BXM.
- After implementation if you modify the attributes that are used by your comparison functions (for example, adding an alias to a name comparison) or you make changes to your bucketing configuration. Comparison function and bucket changes require new weights, a re-derivation of data, and a new BXM.
- When running an IXM.

The utility can be run from a command line or, preferably, from the IBM Initiate Workbench jobset wizard.

When run, this utility selects candidates, compares member records, and assigns comparison scores. The mpxcomp utility must be run once for each type of entity (for example, identity and household) implemented, because the comparison algorithm is specific to each entity type.

Regarding system performance, the mpxcomp utility loads the entire input data set into memory for processing. Working with large can cause memory issues. Your server must have sufficient continuous memory to accommodate the data files. For large data sets, you can elect to use the *Part options to conserve system memory and optimize performance. Use of these options (-nMemParts, ⌂nBktParts, -minBktPart, -maxBktPart, and -maxParts) partitions the data to avoid pulling the entire set into memory at one time. To accommodate available memory, start by adjusting the ⌂nBktParts option.

If you plan to partition data, devise a partitioning strategy before beginning data derivation. Data must be partitioned consistently between the derivation step (mpxdata, mpxfsdvd, mpxprep, or mpxredvd utility), the comparison step (mpxcomp utility), and the linkage step (mpxlink utility).

mpxcomp utility input and output dependencies

The input to mpxcomp is the binary files of derived data (bucket and comparison string binary files) from the mpxdata, mpxfsdvd, mpxprep, or mpxredvd utilities. The binary files are read into memory to speed up all the comparison calculations.

Attention: The mpxdata utility is used to created derived data binary files from raw extracts. If the member data exists in the database, either the mpxprep or mpxredvd utility can be used. If the member data exists in `.unl` files, the mpxfsdvd utility is often used to produce the binary files.

In addition to the derived data binary files, before running the mpxcomp utility you must have:

- A Master Data Engine instance created if you are running the utility from IBM Initiate Workbench; if running from a command line, the engine instance is not required
- A hub configured with your algorithm and data dictionary (includes threshold settings)
- Generated weights

The bulk cross match (BXM) process uses the weights to create an aggregate comparison score, which is then compared to the threshold settings to determine auto-links and tasks.

The output is additional binary files that represent the entity link and task groupings and comparison scores. This output is the input to the next phase of a BXM, which is the mpxlink utility.

mpxcomp utility options

There a number of options you can use when running the mpxcomp utility.

Command-line example: `mpxcomp -entType id -bxmInpDir /u01/isi/bxm -bxmOutDir /u01/isi/bxm.`

About performance tuning in mpxcomp:

The term "partitions" as used in these options refers to breaking the member, bucket, or query data files into pieces. The derivation utilities (mpxdata, mpxprep, mpxfsdvd, and mpxredvd) produce a set of initial bulk cross match (BXM) files that are used by other utilities down-stream to do a cross match or generate weights. If the data set is large, the BXM files are also large. The utilities that read these files (for example, mpxcomp) must be able to fit this data into the available memory (RAM). If the memory requirement is larger than the available memory, the processes might swap or even run out of memory and fail. By breaking the data into pieces (partitions), the utilities can read pieces of the BXM data at a time and run within the available memory.

Keep in mind these points when preparing to run mpxcomp.

1. Both `BktParts` and `MemParts` options are specified when doing data derivation. The output of the data derivation utilities becomes the input for the mpxcomp utility. The value specified for mpxcomp must match the value set for the data derivation processes.
2. `minBktPart` and `maxBktPart` settings override any value set for `nBktParts`.
3. To expedite the BXM process, set `minBktPart` and `maxBktPart` to run multiple processes against the same pool of buckets part files. For example, with a pool of 10 files (mpx_bxmbktd.001 through mpx_bxmbktd.010), running the mpxcomp utility set with `-minBktPart 5` and `-maxBktPart 10` instructs mpxcomp to consume only mpx_bxmbktd.005 through mpx_bxmbktd.010.

Important:

If you want to preserve Enterprise IDs (entrecnos) when running an incremental cross match (IXM), you must specify `-ixmMode` to specify incremental mode when running mpxcomp. If `-ixmMode` is not specified, the downstream mpxlink utility starts with the current entity set (the IXM process also includes running the mpxlink utility). Setting `-ixmMode` causes the mpxlink utility to re-evaluate all entity sets, preserving the previous Enterprise ID when all previous members are present in the new entity set.

When running from the command line, all options are case independent meaning that you can type, for example, `-entType`, `-enttype` or `-ENTTYPE` as your command-line option. However, option values are not case independent. For example, if your entity type is defined as 'id' in the database, you must enter 'id' on the command line as opposed to 'ID' or 'Id'.

Default option values are used if a value is not explicitly set.

Table 50. mpxcomp options

Command line option	Workbench option	Description
`-entType`	Entity Type	This option identifies the type of entity being computed. If you are implementing multiple entity types (for example, id for Identity and hh for Household), you muse run mpxcomp for each type. Required: Yes Default value: none
`-bxmInpDir`	Input directory	This option is the directory where the input binary (`.bin`) files from the mpxdata utility are stored. This directory is typically the work directory created under `MAD_HOMEDIR/inst/mpinet_$instname/work`. Multiple hubs might share the same `MAD_HOMEDIR`, but have different instance names. The hub instance that you connect to dictates where the work directory lives. Required: Yes Default value: none
`-bxmOutDir`	Output directory	This entry is the directory in which you want the mpxcomp output binary files to be placed. This location is typically relative to the work directory on the server hosting the hub configuration. Required: Yes Default value: none
`-nthreads`	Number of threads	The number of threads to used for the mpxcomp process. The number of threads set can have an affect on system performance. The value should correspond to the number of CPUs available on the server (for example, if the server has 4 CPUs, set the number of threads for mpxcomp to 4 to optimize the time it takes to process). Required: Yes Default value: 1 Maximum value: 64 Suggested value: 1 thread per processor

Table 50. mpxcomp options (continued)

Command line option	Workbench option	Description
`-nBktParts`	Number of bucket partitions	This option breaks up the member bucket data (membktd) into smaller, more manageable chunks to optimize memory use. This option can assist sort performance on large data sets. All members that share a given bucket value end up in the same part and are compared to one another. The number of bucket parts you set when running mpxcomp should match the number you specified for the derivation utility process (mpxdata, mpxfsdvd, or mpxredvd). If running an IXM, set this option to the number used in the original data load. To accommodate for available memory, BktParts is the first option you adjust. If `-minBktPart`hand `-maxBktPart` options in mpxcomp are used, they override any settings for `-nBktParts`. When attempting to reduce your memory footprint, increase BktParts before adjusting MemParts. Running a utility with the `-noexec` option outputs memory usage requirements that can help you determine how to adjust the -n*Part settings. Required: Optional Default value: 1 Maximum value: 100 Suggested value: Match the number of bktParts created in the bxmInpDir.

Table 50. mpxcomp options (continued)

Command line option	Workbench option	Description
`-nMemParts`	Number of member partitions	This option identifies the data partitions consumed by the mpxcomp utility. When this option is defined for the mpxdata, mpxprep, mpxredvd, or mpxfsdvd utilities, the processes breaks up the data from memHead and memCmpd (comparison data). While using this option can cut your memory usage significantly, your setting can affect performance. The higher the memParts setting, the slower your comparison process because the Master Data Engine is forced to do more duplicate comparisons. In order to compare every member that shares at least one bucket, the engine compares each memPart against itself and then against all other memParts. For example if you had memParts set to 3, you would have parts A, B, and C. For each BktPart, you compare: A > A , A > B, A > C B > B, B > C C > C If it is necessary to use this option, specify a minimum setting of 3. Since the comparison has to bring in two parts to compare against each other, only splitting the data in half does not save memory. Use enough parts to get all of the comparison data into physical memory. Required: Optional Default value: 1 Maximum value: 100 Suggested value: 1. The value used for the data derivation process should be the same values used for mpxcomp and mpxlink.
`-minBktPart`	Minimum bucket partitions to process	Determines the minimum number of bucket parts to process. Both `-minBktPart` and `-maxBktPart` are performance options that allow mpxcomp to process a range of bucket parts. This option is often used when running mpxcomp across multiple servers. For example, you can run bucket parts 1 through 5 on server 1 and parts 6 through 10 on server 2. Required: Optional Default Value: 0 Maximum value: Less than or equal to the value of the `maxBktPart` option

Table 50. mpxcomp options (continued)

Command line option	Workbench option	Description
`-maxBktPart`	Maximum bucket partitions to process	Determines the maximum number of bucket parts to process. Required: Optional Default Value: 0 Maximum value: 100 **Important:** The mpxcomp utility fails when: • Either `minBktPart` or `maxBktPart` option is set to 0 (which means `bktPart` is not used). The minimum and maximum bucket parts must be set to a valid range (for example, `-minBktPart 1 - maxBktPart 5`). • The value of `minBktPart` is greater than or equal to the value of `maxBktPart`.
`-nMxmParts`	Maximum number of output partitions	This option partitions the output of mpxcomp into smaller chunks for use by the mpxlink utility. The value set for mpxcomp determines how many partitioned file segments are passed to mpxlink, thus the MxmParts value for both must be the same. Required: Optional Default Value: 1 Maximum value: 100
`-maxbktsize`	Maximum bucket set size	Maximum bucket size determines the maximum number of members that can have the same bucket value for candidate selection. For example, with a the default of 500, if more than 500 members have the same bucket value, mpxcomp ignores those members for comparison. The value set depends on various factors, including the number of members in the database and the bucketing strategy. The log reports bucket hash values exceeding this parameter, as well as 5 members for the user to examine. If set appropriately, the values reported in the log might indicate a bucket value that can be defined as an anonymous value. Required: Optional Default Value: 500 Maximum value: 1048576
`-minBktTag`	Minimum bucket role	Minimum bucket tag to use. Bucket tags are used for speed optimization. Tags allow bucket data to be created on bucket roles greater than or equal to the minimum bucket tag, and less than or equal to the maximum bucket tag. By using the bucket tag option, you can eliminate roles that do not have any impact on the linking outcome. Required: No Default Value: 0 Maximum value: 15

Table 50. mpxcomp options (continued)

Command line option	Workbench option	Description
-maxBktTag	Maximum bucket role	Maximum bucket tag to use. Required: No Default Value: 0 Maximum value: 15
-cmpMode	Comparison mode	This option controls the mpxcomp utility comparison behavior and is intended to improve performance by excluding comparisons configured only for searching. Comparison modes can be set in your algorithm as such: cmpmode 1 = match and link members, cmpmode 2 = search members, cmpmode 3 = search, match and link members. The mode set in your algorithm does not have to match the option specified for mpxcomp. The cmpMode option acts as a filter for selecting which comparison functions are used for comparison. For example, if you specify option 1 (match and link), mpxcomp uses only the comparison functions that are set to match and link. Typically you use mpxcomp with comparison mode 3. If this option is not set, all comparison modes configured in your algorithm are compared. If set to 1, comparison modes 1 and 3 are compared. If set to 2, comparison modes 2 and 3 are compared. Required: No Default Value: 3
-{no}bxmLink	Write linkage item records	Determines whether to write the output for linkage records. The output is written to a file Use -nobxmLink if you do not want to write records. Required: No Default Value: -bxmLink (write linkage records)
-{no}bxmTask	Write task item records	Determines whether to write the output for task records. Use -nobxmTask if you do not want to write records. Required: No Default Value: -bxmTask (write task records)
-{no}bxmRvid	Write review identifier item records	Determines whether to write the output for Review Identifier task records. Use -nobxmRvid if you do not want to write records. Required: No Default Value: -bxmRvid (write Review Identifier tasks)
-{no}DiffSrcOnly	Do only different source comparisons	Use this option if you want to compare members from one source to only records in a different source. For example, records from Source A are compared to records in Source B and C, but not against records in A. Required: No Default Value: -noDiffSrcOnly (compares records across all sources)

Table 50. mpxcomp options (continued)

Command line option	Workbench option	Description
-{no}SameSrcOnly	Do only same source comparisons	Use this option to compare records against only those from the same source. For example, records from Source A are compared against A, but not B and C. Required: No Default Value: -noSameSrcOnly (compares records across all sources)
-allAtts	Not applicable for mpxcomp job. This option is found in the Weight Generation job.	This true or false option applies to weight generation and generates statistics on all attributes used in the comparison. When used with the -Msamp option, matched set statistics are generated by using all the attributes. Without use of the -allAtts option (set to false) during stat generation for an attribute, the attribute weight is omitted when computing. Required: No Default Value: True
-wgtInpDir	Not applicable.	Use this setting to specify the directory containing weight (wgt) tables as .unl files that are used in the comparison. This setting is optional and is used during weight generation when weights from another step in the process need to be considered (for example, determining when weights have converged, meaning the previous weights are compared to the new weights). Required: No Default Value: None
-strInpDir	Not applicable.	Use this setting to specify the directory containing string (str) tables as .unl files that are used in the comparison. Required: No Default Value: None
-encoding	Encoding	Determines the encoding of the .unl files. Options are latin1, utf8, and utf16. Required: No Default Value: latin1
-Usamp	Not applicable for mpxcomp job. This option is found in the Weight Generation job.	This option directs the process to conduct random (unmatched) pair sampling during weight generation. This option compares unmatched pairs of members. Unmatched means that the members are not in the same buckets, while matched pairs are in common buckets. Required: No Default Value: -noUsamp
-Msamp	Not applicable for mpxcomp job. This option is found in the Weight Generation job.	This option directs the mpxcomp process to conduct matched pair sampling during weight generation. Required: No Default Value: -noMsamp

Table 50. mpxcomp options (continued)

Command line option	Workbench option	Description
`-Mpair`	Not applicable for mpxcomp job. This option is found in the Weight Generation job.	This option directs mpxcomp to create candidate matched pairs during weight generation. `-Mpair` is shorthand for `-bxmItem -tMinWgt 80% comphead.wgtmat`, meaning that you can specify `-bxmItem`, `-tMinWgt`, and `-tMaxWgt` explicitly, and possibly with a different value for `-tMinWgt`. Or you can use `-Mpair`, which assumes the options: `-bxmItem`, `-tMinWgt` is 80 percent of the value set in `mpi_cmphead.wgtmat`, and `tMaxWgt` = 0. Required: No Default Value: `-noMpair`
`-bootWgts`	Not applicable for mpxcomp job. This option is found in the Weight Generation job.	This option can be used during the unmatched (random) pair sampling stage of the weight generation process to generate the initial weight tables. This setting overrides the requirements that explicit weight tables exist in the dictionary. Required: No Default Value: `-noBootWgts`
`-{no}bxmItem`	Not applicable.	Write item records. Used for weight generation only. Required: No Default Value: `-noBxmItem`
`-tMinWgt`	Not applicable for mpxcomp job. This option is found in the Weight Generation job.	The threshold for the minimum weight to include in the output. Used for weight generation only. Required: No Default Value: 0 (all)
`-tMaxWgt`	Not applicable for mpxcomp job. This option is found in the Weight Generation job.	The threshold for the maximum weight to include in the output. Used for weight generation only. Required: No Default Value: 0 (all)
`-noExec`	Generate memory usage information only	This true or false option generates memory usage information. This information is viewable in IBM Initiate Workbench by running the **Get job results** action on the mpxcomp job entry returned by the hub. Required: No Default Value: false
`-ixmMode`	Not applicable.	This true or false option sets the IXM mode. In IXM mode, a subset of members are compared rather than the entire member set. If running a BXM, use the default of false. If running an IXM, set this option to true. Required: No Default Value: false

Table 50. mpxcomp options (continued)

Command line option	Workbench option	Description
-{no}dense	Not applicable.	When -dense is specified, mpxcomp creates a memhead lookup table that is used during the comparison operation. The lookup table replaces runtime computation with a simple array indexing operation. The -dense option uses more memory, but is faster than -nodense. Specify -dense when you have sufficient memory for the data set, and if you have large gaps in your memRecno ranges. The default is -nodense.

mpxconv utility

The mpxconv utility is a weight generation program to check for weight table convergence.

All options and flags are case independent; option values are not.

Table 51. mpxconv options

Option	Type	Description	Default
-entType	Name	Entity type name	NONE
-wgtInpDir	dirName	Input weight directory	NONE
-encoding		Determines the encoding of the .unl files. Options are LATIN1, UTF8, or UTF16.	LATIN1

mpxdata utility

The mpxdata utility uses raw data to build member unload files (.unl), generate comparison strings, assign bucket hashes, and create binary files.

The utility allows you to specify memput (member put) or memcompute (member compute) interactions based on file input. You can run mpxdata from IBM Initiate Workbench **Initiate menu** > **New Job Set**. See the *IBM Initiate Workbench User's Guide* or from a command line.

The mpxdata utility performs several steps while running, from parsing data into .unl files to deriving data and organizing member records into buckets. The mpxdata utility also creates binary files, which are used to compare data faster than scanning through strings. The mpxdata utility parses raw data extracts into attribute-specific sets of data. For example, it can take a single record for a person and create one record for the address elements, another for the name elements, and a third for the telephone numbers. Parsing allows the hub to store multiple iterations of active and inactive data (such as a former address or phone number) and increases responsiveness when searching and comparing.

The mpxdata utility logic can process multiple attributes for a single member from the input data file. The attribute rows in the data file are grouped together by the record identifier (source code and memidnum pair), which means all attribute rows for the same member are continuous. Duplicate values are treated as a single

value, and empty values are skipped. Active (nsactive) and maximum (nsexist) settings are enforced before the attribute values and derived data are written into the output file.

All options and flags are case independent; option values are not.

Restriction: When running the mpxdata utility, you must use an ODBC connection.

Table 52. mpxdata options

Option	Type	Description	Default
`-ixnCode`	ixnCode	Interaction code, either MEMCOMPUTE or MEMPUT. MEMPUT inserts or updates members in an existing hub database for each record in the input file. MEMCOMPUTE generates `.unl` files which can then be loaded by using the madunlload utility, or another load utility. When processing an extract, MEMCOMPUTE is most often used because loading `.unl` files is faster than inserting each member through MEMPUT.	NONE
`-putType`	enumVal	Put type (MEMPUT only). Choices are insert_update, insert_only, and update_only. This option works at a member level, not an attribute level. • **insert_only** restricts the Master Data Engine to creating a member. If a member exists for this srcCode and memIdnum combination, the interaction fails with an error code of EXISTS. • **update_only** restricts the Master Data Engine to updating existing members only. If an attempt is made to update a member that does not exist, the interaction fails with an error code of ENOREC. • **insert_update** adds a member if one does not exist. If the member does exist, an update is made.	MPI_PUTTYPE_ INSERT_UPDATE

Table 52. mpxdata options (continued)

Option	Type	Description	Default
`-memMode`	enumVal	Member mode (MEMPUT only). Choices are complete, partial, attrcomp, and explicit. More detail on these modes can be found in the "Member put interaction" appendix of the *IBM Initiate Master Data Service SDK Reference for Java and Web Services*. • **Partial** is used when a source system sends an update to a member, but you do not know if the input is a complete picture of the member, or if you have the complete range of values for a given attribute. • **Attrcomp** stands for attribute complete. Like the partial mode, the attrcomp mode tells the Master Data Engine that it might not have a complete picture of all the attributes that make a complete member. However, for the attributes that are present, all known values for the member are included in the member put interaction. • **Complete** tells the engine that the input to the member put interaction contains all of the values for all of the attributes defined for this member type. • **Explicit** is used in situations where you want to control exactly what is stored for the member and the record status of the attributes being stored.	MPI_ MEMMODE_ COMPLETE
`-entPrior`	N	Sets the entity management priority. Use this option when you want to set the priority at member write and override any default entity priority previously set for the associated source.	source priority (Default Entity Priority setting in IBM Initiate Workbench)
`-config`	fileName	Name of configuration file. This is a specially formatted file that defines the fields for the data input file. For more information about the structure of the configuration file, see *IBM Initiate Workbench User's Guide*.	NONE
`-recSize`	N	Fixed-length record size (for fixed-length input files). Add the appropriate end-of-line characters to this value.	NONE
`-fldDelim`	delimChar	Field delimiter character for variable length record fields	|

Table 52. mpxdata options (continued)

Option	Type	Description	Default
`-inpFile`	fileName	Input file name defined by the configuration filename and either fixed length or delimited by the field delimiter (fldDelim) character. The location of this file is in the hub instance directory project workspace: (`MAD_HOMEDIR\inst\mpinet_`*`instance_name`*`\work\`*`project_name`*`\work`).	`input.dat`
`-rejFile`	filename	Rejected record file. If mpxdata is unable to parse data as it reads each row in the input file, it writes that data to the Rejects file. The mpxdata utility continues to parse remaining data, adding any additional "rejected" data to the rejects file. The default filename is `rejects.txt`.	`reject.dat`
`-maxRecs`	N	Maximum number of records to process before ending the mpxdata process. This option is disabled when the "Process all records" option is selected.	Unlimited
`-maxErrs`	N	Maximum number of errors allowed before stopping the process. This option sets a threshold for errors in the data. Once the threshold is reached, the mpxdata utility stops (that is not considered an mpxdata error). This option allows you to process an extract with tolerance for known data issues. For example, if the delimited extract file has too few or too many delimiters in a few records, you can set this option to an expected value. If the value is exceeded, mpxdata stops and gives you an opportunity to resolve the problem in the extract data or configuration file. The mpxdata utility writes records it cannot parse to the rejects file. The total error count (totErrs) is reported in the mpxdata log as an INFO message: `06:59:53 mpxdata INFO MPX_BxmData: totRecs=6, totErrs=3, elapsed=1 secs., recs/sec=6, minbkttag=0, maxbkttag=0, nMemParts=1, nBktParts=1, buffsize=65536`	100
`-skipRecs`	N	Number of records to skip before beginning processing. If there are any rows of text in the input file before the data rows begin, indicate how many rows to ignore. The number of skipped rows does not include lines that are commented out with the hash (#) character.	0

Table 52. mpxdata options (continued)

Option	Type	Description	Default
`-rptRecs`	N	Report records processed interval. The mpxdata log reports a status every *n* records. You might want to decrease the frequency to reduce the log output for large data sets, or increase it to get more granularity.	100000
`-buffSize`	N	Size for each file input and output buffer	65536
`-verbose`		Show progress information	FALSE
`-noexec`		Show SQL statements only; no execution is performed	FALSE
`-encoding`		Encoding of `.unl` files; options are LATIN1, UTF8, or UTF16	LATIN1
`-methods`		Output the method data, but do not process data. For details about using methods to trim blanks and zeros from data, see *IBM Initiate Workbench User's Guide*.	NONE
`-version`		Output the version information	NONE
`-memRecno`	N	MEMCOMPUTE ONLY. Starting memrecno. The value supplied is used as the first memrecno in the `.unl` files and is incremented by one for each additional record.	1
`-audRecno`	N	MEMCOMPUTE ONLY Common audrecno. The value supplied is used as the first audrecno in the `.unl` files and is incremented by one for each additional record.	2
`-unlOutDir`	dirName	MEMCOMPUTE ONLY. This option identifies the `.unl` file output directory. Used with `-unlOutSegs`, this option instructs mpxdata to create `.unl` files after reading and parsing the extract file.	NONE
`-unlOutSegs`	segList	MEMCOMPUTE ONLY. The Attribute segments to include in the output. Used with `-unlOutDir`, this option instructs mpxdata to create `.unl` files after reading and parsing the extract file. This option enables you to select the attributes (segments) to be included in the `.unl` output files.	NONE

Table 52. mpxdata options (continued)

Option	Type	Description	Default
`-unlAudSegs`	segList	Use this option to specify the audit segments you want to create during the mpxdata process and include in the `.unl` files. This allows you to preserve the record creation or last modified time and map it to the evtctime field, as well as mapping additional source system information to the evtType, evtInitiator and evtLocation audit fields. The mapping of these fields is available in the `.cfg` file specified by the `-config` option . You can choose audhead or audxmem. When using the `-unlAudSegs` option, the `-audhead` option is unavailable and the `-audRecno` indicates the starting audrecno for the records being created.	NONE
`-bxmOutDir`	dirName	MEMCOMPUTE ONLY . `.bin` output directory. Specifies the directory where the bulk cross match (BXM) files are saved.	NONE
`-{no}bxmBktd`		Generate member bucket (MEMBKTD) output	`-bxmBktd`
`-{no}bxmCmpd`		Generate member comparison data (MEMCMPD) output	`-bxmCmpd`
`-{no}bxmQryd`		Generate member query data (MEMQRYD) output. This option is used with the relationship linker and instructs mpxdata to create BXM files containing query data. The relationship types, attributes, and rules must already be defined, so that mpxdata knows what data to include in the BXM file.	`-bxmQryd`
`-nMemParts`	N	MEMCOMPUTE ONLY . Number of member partitions. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have on the hub server. The utility that consumes the mpxdata output (such as mpxfreq) must use a matching “memparts” value. Leave this option at the default unless you need more memory. The higher the member partition the slower your mpxcomp process, as the hub must do more duplicate comparisons.	1
`-nBktParts`	N	MEMCOMPUTE ONLY. Number of bucket partitions. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have on the hub server. Leave this option at the default unless you need more memory.	1

Table 52. mpxdata options (continued)

Option	Type	Description	Default
`-minBktTag`	N	MEMCOMPUTE ONLY. Minimum bucket tag to use (0=any). Specifies the lowest bucketing role to be included in the operation.	0
`-maxBktTag`	N	MEMCOMPUTE ONLY. Maximum bucket tag to use (0=any). Specifies the highest bucketing role to be included in the operation.	0
`-nQryParts`	N	MEMCOMPUTE ONLY. Number of query partitions. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have on the hub server. Leave this option at the default unless you need more memory. This option is enabled only when the option to Generate query BXM is also enabled.	1
`-minQryRole`	N	MEMCOMPUTE ONLY. Minimum query role to use (0=any) when the 'Generate query BXM' option is enabled. This option specifies the lowest query role to be included in the operation.	0
`-audhead`		MEMCOMPUTE ONLY. Write audhead records. When enabled, audhead records are written to `.unl` files (to be uploaded to the database later).	FALSE
`-append`		MEMCOMPUTE ONLY. Append to `.unl` files. This option applies only to the `.unl` files. If you are processing multiple extract data files (for example, from different sources), mpxdata writes new (or overwrites existing) `.unl` files when this option is not used. If used, the new `.unl` data written by mpxdata is added to the end of the existing `.unl` file.	FALSE
`-memType`	memName	MEMCOMPUTE ONLY. Member type name. This option sets a filter on the output of mpxdata for the specified member type. Setting this field to ALL processes all member types for the hub.	NONE
`-entType`	entName	MEMCOMPUTE ONLY. Entity type name.	NONE
`-strInpDir`	dirName	Allows specification of a directory containing string (str) tables as `.unl` files to update or append the contents in the dictionary.	NONE

mpxdist utility

The mpxdist utility is a weight generation program used to compute weight distributions, as well as clerical review and auto-link scores.

All options and flags are case independent; option values are not.

Table 53. mpxdist options

Option	Type	Description	Default
`-entType`	Name	Entity type name	NONE
`-bxmInpDir`	dirName	`.bin` input directory	NONE
`-bxmOutDir`	dirName	`.bin` output directory	NONE
`-nMemParts`	N	Number of member partitions	1
`-nthreads`	N	Number of threads	1
`-wgtInpDir`	N	`.unl` weight input directory override; allows specification of a directory containing weight (wgt) tables as `.unl` files to update or append the contents in the dictionary	NONE
`-strInpDir`	N	`.unl` string input directory override; allows specification of a directory containing string (str) tables as `.unl` files to update or append the contents in the dictionary	NONE
`-encoding`		Encoding of `.unl` files; options are LATIN1, UTF8, or UTF16	LATIN1
`-cmpMode`	N	Comparison mode: • `1` for match and link • `2` for search • `3` for match, link, and search	3

mpxdump utility

The mpxdump utility dumps binary files to text format.

Table 54. mpxdump options

Option	Type	Description	Default
`-binFile`	filename	Binary file name	NONE
`-mapFile`	Filename	Binary file name	NONE
`-tFPR`	V	False positive rate (distribution sampling file [dsamp] only)	NONE
`-tFNR`	V	False negative rate (dsamp only)	NONE

mpxfprof utility

The mpxfprof utility produces diagnostic information about the size and the number of fields and records in a data file.

Table 55. mpxfprof options

Option	Type	Description	Default
`-inpFile`	fileName	Input file name	
`-encoding`		Encoding of `.unl` files; options are LATIN1, UTF8, or UTF16	LATIN1
`-rptFile`			
`-fldDelim`		Delimiter used for field separation	
`-header`		File header	
`-skipRecs`		Skip specified records	
`-maxRecs`		Maximum number of records to process	

mpxfreq utility

The mpxfreq utility creates frequency files to support weight generation.

This utility can be run from the command line or from IBM Initiate® Workbench through the **Initiate menu** > **New Job Set** option. See the *IBM Initiate Workbench User's Guide* for job set details.

The mpxfreq utility generates data from which attribute frequencies can be derived. It requires derived data as input, such as that generated by the mpxdata, mpxprep, mpxfsdvd, or mpxredvd utilities.

The mpxfreq utility has three main options that determine the purpose of running the utility:

- The `-rawMode` option generates raw frequency table output for counting the values that are checked to determine whether they are anonymous.
- The `-fbbMode` option is for generating frequency tables to enable frequency-based bucketing (FBB). Frequency-based bucketing counts the number of occurrences of a particular string in the database, and then determines the buckets that exceed the maximum bucket frequency number (set in the bucket group properties in the algorithm).
- The `-wgtMode` option is used for generating frequency tables for weight generation.

Keep in mind these items when preparing to run mpxfreq:

- All options and flags are case independent; option values are not.
- Either `–memType` or `–entType` can be specified, but not both.
- Only one of `-rawMode`, `–wgtMode` or `–fbbMode` can be specified.
- In `–fbbMode`, the output is `mpx_bkrfreq_MEMTYPE.sql` and `mpi_strhead/freq.unl`.
- In`-wgtMode`, the output is `mpx_wgtfreq_MEMTYPE.txt` and is used by the mpxwgts utility.
- In `-rawMode`, the output is `mpi_strhead.unl` and `mpi_strfreq.unl` and is used by the Anonymous Values Utility.

- In –wgtMode, mpx_cmpfreq_MEMTYPE.sql is the output for experimental use only.
- When using -wgtMode, -entType is required.
- –refSrcCode is used to sub-filter the memType and entType.
- If using the -MI option, -minFreqCnt2 defaults to 1 if it is not explicitly set.
- If you do not specify -MI, then -minMemPart and -maxMemPart options cannot be specified.
- You can specify -minMemPart, -maxMemPart, and -nMemparts. However, be aware that the-maxMemPart setting overrides the -nMemParts setting.

Table 56. mpxfreq options

Option	Type	Description	Default
-memType	Name	Member type name. If you have multiple member types in the hub database and want to compute frequencies for one of those member types, the member type filter can be used. All entity types for that member are processed.	NONE
-entType	Name	Entity type name. If you have multiple entity types in the hub database and want to compute frequencies for one of those entity types, the entity type filter can be used. All member types within the specified entity type are processed.	NONE
-rawMode		Instructs mpxfreq to count the values that are checked to determine whether they are anonymous. (This option was called anonMode in earlier versions of the software.) The option provides the numeric input for the Anonymous Values Utility. You must have anonymous string codes configured, and they must be referenced in the standardization functions in your algorithm before you can run the mpxfreq utility to generate the output to feed to the Anonymous Values Utility. For more detail, see the *IBM Initiate Workbench User's Guide*. **Restriction:** This option does not generate the files needed for weight generation. When running the Generate Weights job, start from the "Delete artifacts from previous run" option. See the *IBM Initiate Workbench User's Guide*.	NONE

Table 56. mpxfreq options (continued)

Option	Type	Description	Default
-{no}cmpFreqSql		This -rawMode flag instructs mpxfreq to generate SQL files in the \frq subdirectory. You typically do not need both SQL output and UNL output.	-cmpFreqSql
-{no}cmpFreqUnl		This -rawMode flag instructs mpxfreq to generate .unl files in the \frq subdirectory. You typically do not need both SQL output and UNL output.	-cmpFreqUnl
-rawOutFile	Name	When using -rawMode with -MI and -merge options, this flag specifies the output file for the current patterns.	NONE
-fbbMode		Generates mpi_strfreq tables for all buckets that have "minway < maxway" settings. Uses the derived bucket data to generate a list of string frequencies. If a given string is over the maximum limit, it is added to a list in the dictionary and is not used for candidate selection. • Limits are specified in the algorithm. • Each bucket role can have a different limit. • Static process – you must run this step again as your data set changes and grows. If any strings are added to strFreq, then you can re-derive the bucket data in order to reduce the number of comparisons required by mpxcomp.	NONE
-{no}bktFreqSql		This -fbbMode flag generates SQL output. Instructs mpxfreq to generate SQL files in the FRQ output directory. You typically do not need both SQL output and UNL output.	-nobktFreqSql
-{no}bktFreqUnl		This -fbbMode flag generates UNL output. Instructs mpxfreq to generate UNL files in the FRQ output directory. You typically do not need both SQL output and UNL output.	-bktFreqUnl
-fbbNway	Name	This -fbbMode flag used only in combination with the -MI option, instructs mpxfreq to do a single pass of the Nwayfbb patterns.	NONE

Table 56. mpxfreq options (continued)

Option	Type	Description	Default
`-fbbInpList`	Name	This `-fbbMode` flag used only in combination with the `-MI` option, specifies the input file or files for the previous patterns.	NONE
`-fbbOutFile`	Name	This `-fbbMode` flag used only in combination with the `-MI` and `-merge` options, specifies the output file for the current patterns.	NONE
`-wgtMode`		Generates frequency tables for weight generation. For more detail, see the *IBM Initiate Workbench User's Guide*. This option is identical to running the "Generate counts for all attribute values" option in the Generate Weights job with "**Execute all remaining steps through end of process**" disabled. The advantage to using this option, rather than executing the Generate Weights script functionality, is that you can take advantage of the Performance Tuning options when generating the frequency tables. When using this option, be sure to direct the frequency output to the correct directory. The Generate Weights job typically expects to find the output in the `\weights\frq` subdirectory. If you later run Generate Weights starting with "**Generate random pairs of members**" you must first create the necessary directories for the weight generation output.	NONE
`-{no}wgtFreqTxt`		This `-wgtMode` flag is used only in combination with `-MI` and `-merge` options, and instructs mpxfreq to generate an `mpx_wgtfreq_ENTTYPE.txt` file.	`-wgtFreqTxt`
`-wgtOutFile`	Name	This `-wgtMode` flag is used only in combination with `-MI` and `-merge` options, specifies the output file for the current patterns.	NONE

Table 56. mpxfreq options (continued)

Option	Type	Description	Default
`-bxmInpDir`	dirName	`.bin`; specifies the bulk cross match (BXM) input directory. This directory is the BXM directory that contains the comparison binary work files. These files were produced by one of the data derivation methods, so match this path to BXM output directory specified by the selected derivation process.	NONE
`-nMemParts`	N	Number of member partitions. This number should correspond to the number of member partitions used when running mpxdata, mpxprep, mpxredvd, or mpxfsdvd (these utilities can also produce the starting BXM data). If you use the `-MI` option and set the `-maxMemPart` flag, the `-nMemParts` setting is overridden.	1
`-frqOutDir`	dirName	`.frq`; specifies the output directory to which frequency results are written.	NONE
`-nthreads`	N	Number of threads. This value should correspond to the number of processors available on the hub. The goal is to take advantage of all the processing resources available. For example, if running the hub on a computer with four processors, set the number of threads to 4. This setting keeps all four processors busy and minimizes the time mpxfreq takes to run. If you were to set it to 2, only two processors would be used, and the processing time would be longer. Setting it too high would cause the hub to switch back and forth between running threads and threads waiting for available processor cycles.	1
`-refSrcCode`	srcCode	Sub-filter by this source code. Enables frequency analysis to be performed on a single source (`-memType` and `–entType` are still required with this option).	NONE
`-minFreqCnt2`	N	Overrides the data dictionary settings. If you are using the `-MI` option, `-minFreqCnt2` defaults to 1 if not explicitly set.	NONE

Table 56. mpxfreq options (continued)

Option	Type	Description	Default
`-hash1Cnt`		Primary hash slot count. This setting is the number of slots available in the hash table. The larger the diversity of the data set, the greater the advantage in increasing this number. Each slot maintains a list of string values with their counts. The fewer string values that need to be parsed in a slot, the faster the program runs. Increasing this number also increases memory requirements.	
`-hash2Cnt`		Compression hash slot count. This setting is the number of slots available during the join of multiple threads into the master frequency list. All records have been processed and the hub is adding up the totals. This number is typically smaller than primary hash slot count. Like primary hash slot count, increasing this number also increases memory requirements.	
`-h12CvPct`		Slot compression ratio. This option allows the hash tables to be compressed down if the ratio of total nodes to nodes per nway falls above the percentage specified. `-h12CvPct` sets scavenging, converting primary to secondary hash.	
`-pageSize`		Page size. This setting is the amount of memory in megabytes initially allocated for string nodes. Base this value on the size of the strings being frequencied and the number of unique string values.	
`-audRecno`	N	Common audRecno for all `.unl` and `.sql`	2
`-encoding`		Encoding of `.unl` files; options are LATIN1, UTF8, or UTF16.	LATIN1

Table 56. mpxfreq options (continued)

Option	Type	Description	Default
-MI		Sets multi-instance mode. Use the -MI option with -fbbNway (but without ⌂merge) on multiple servers to do a single pass of the Nway fbb patterns. By specifying a different -fbbNway on each server, you can reduce the time required to generate frequencies. Bring all the files together on one server and use -MI and -merge together to unify the parts into an output file.	-noMI
-minMemPart	N	This flag is used only with the -MI option and specifies the minimum number of member partitions that you want each instance to work on.	0
-maxMemPart	N	This flag is used only with the -MI option and specifies the maximum number of member partitions that you want each instance to work on.	0
-merge		Use only in combination with -MI, merges results from multiple instance runs.	-noMerge
-mrgInpList	Name	This -merge flag specifies the input list of files from other instance runs to be merged.	NONE
-mrgOutFile	Name	This -merge flag specifies the output file for the merged results.	NONE

mpxfsdvd utility

The mpxfsdvd utility enables the creation of bulk cross match (BXM) files from .unl files.

This utility is a data derivation method that uses pre-existing member unload files to extract and create comparison strings, bucket hashes, and binaries. It is most commonly used when you have made changes to your algorithm but the data itself has not changed. This utility can be run from a command line or from the IBM Initiate Workbench **Initiate menu** > **New Job Set**. See *IBM Initiate Workbench User's Guide* for more information

Keep in mind these items when preparing to run mpxfsdvd:

- All options and flags are case independent; option values are not.
- Both -unlInpdir and -unlInpSegs are required.
- Either one or both of -unlOutdir -unlOutSegs or -bxmOutDir must be specified.

Important: If you want to preserve Enterprise IDs (entrecnos) when running an incremental cross match (IXM), you must use the -ixmmode option when running the mpxfsdvd utility. If -ixmmode is not specified, the downstream mpxlink utility

process starts with the current entity set. Setting `-ixmmode` causes mpxlink to re-evaluate all entity sets, preserving the previous Enterprise ID when all previous members are present in the new entity set.

Table 57. mpxfsdvd options

Option	Type	Description	Default
`-unlInpDir`	dirName	Location of `.unl` files. The mpxfsdvd utility reads the member attribute data from the `.unl` files in the directory specified here. This directory is relative to the project work directory on the hub: `MAD_HOMEDIR\inst\ mpinet_`*`instance_name`*`\work\ `*`project_name`*`\work\`*`UNL_INPUT_DIR`	NONE
`-unlInpSegs`	segList	List of segments contained by the `.unl` files	NONE
`-unlOutDir`	dirName	`.unl` file output directory. The output of the mpxfsdvd utility is the derived data segments (comparison, bucket, and, optionally, query data), which have their own `.unl` files (mpi_memcmpd, mpi_membktd, and mpi_memqryd) written to the directory specified here. This directory is relative to the project work directory on the hub: `MAD_HOMEDIR\inst\ mpinet_`*`instance_name`*`\work\ `*`project_name`*`\work\`*`UNL_OUTPUT_DIR` Used with the ⌂`unlOutSegs` option and indicates whether mpxfsdvd should generate `.unl` files during processing. Also with `-unlOutSegs`, instructs mpxfsdvd to create `.unl` files containing bucket data or comparison data, or instructs mpxfsdvd to generate query `.unl` files during processing (the files are used by the relationship linker).	NONE
`-unlOutSegs`	segList	Attribute segments to output. Used with the `-unlOutDir` option and indicates whether mpxfsdvd should generate `.unl` files during processing. Instructs mpxfsdvd to create `.unl` files containing bucket data or comparison data, or instructs mpxfsdvd to generate query `.unl` files during processing (the files are used by the relationship linker).	NONE
`-encoding`		Encoding of `.unl` files; options are LATIN1, UTF8, or UTF16	LATIN1

Table 57. mpxfsdvd options (continued)

Option	Type	Description	Default
`-bxmOutDir`	dirName	`.bin` output directory. Indicates where you want the BXM output files to be located. This directory is relative to the project work directory on the hub: `MAD_HOMEDIR\inst\ mpinet_`*`instance_name`*`\work\ `*`project_name`*`\work\`*`BXM_OUTPUT_DIR`	NONE
`-{no}bxmBktd`		Generate MEMBKTD output.	`-bxmBktd`
`-{no}bxmCmpd`		Generate MEMCMPD output.	`-bxmCmpd`
`-{no}bxmQryd`		Generate MEMQRYD output. This option is for use with the relationship linker and instructs mpxfsdvd to create BXM files containing query data. The relationship types, attributes, and rules should already be defined so that mpxfsdvd knows what data to include in the BXM file.	`-bxmQryd`
`-nMemParts`	N	Number of member partitions. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the hub. The utility that consumes the mpxfsdvd output (such as mpxfreq) must use a matching "memparts" value. Leave this option at the default unless you need the memory. The higher the member partitions, the slower your mpxcomp process because the hub must do more duplicate comparisons.	1
`-nBktParts`	N	Number of bucket partitions. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the hub. Leave this setting at the default unless you need the memory.	1
`-minBktTag`	N	Minimum bucket tag to use (0=any). The lowest bucketing role designation used in the algorithm to include in the process.	0
`-maxBktTag`	N	Maximum bucket tag to use (0=any). The highest bucketing role designation used in the algorithm to include in the process.	0

Table 57. mpxfsdvd options (continued)

Option	Type	Description	Default
`-nQryParts`	N	Number of query partitions. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the hub. Leave this setting at the default unless you need the memory. This option is enabled only when the option to Generate query BXM is also enabled.	1
`-minQryRole`	N	Minimum query role to use (0=all). The lowest query role designation used in the algorithm to include in the process. This option is enabled only when the option to Generate query BXM is also enabled.	0
`-buffSize`	N	Size for each file input and output (I/O) buffer.	65536
`-memType`	memName	Member type name. If you have multiple member types in the hub database and need to derive data for only one of those member types, select the member type here; otherwise, select ALL.	NONE
`-entType`	entName	Entity type name	NONE
`-skipRecs`	N	Number of member records to skip before re-deriving members from the specified input files. Processing begins with the next member read from MEMHEAD after skipping this number of records. When used with the `-maxRecs` option, this parameter lets you set a range of members from the specified input file to process.	0
`-maxRecs`	N	Maximum number of member records to re-derive from the specified input files. When using this parameter along with skipping member records, this number includes the number skipped. When used with the `-skipRecs` option, this parameter lets you set a range of members from the specified input file to process. This option is useful when running multiple instances of MPXFSDVD against the same set of input files	Unlimited

Table 57. mpxfsdvd options (continued)

Option	Type	Description	Default
`-maxErrs`	N	Maximum errors before halting processing. This option sets a threshold for errors in the data. Once the threshold is reached, the mpxfsdvd utility stops. The intent of this option is to allow you to process a set of input `.unl` files with tolerance for data issues. For example, if the `.unl`file has an incorrect number of fields, the member record is rejected and re-derivation does not complete for that member. The mpxfsdvd utility writes detailed information into the log file, including the line number, input file, and reason for the rejection.	100
`-{no}HeadSql`	flag	Generates SQL output. Instructs the mpxfsdvd utility to generate an SQL file in the specified `.unl` output directory. If a `.unl` output directory is not specified then the output is written to the BXM output directory. This SQL file contains a query against the mpi_memhead table for members that were identified as missing. These members are identified when there is an attribute row that does not have a corresponding head row.	`-noHeadSql`
`-ixmmode`		This true or false option sets the IXM mode. In IXM mode, a subset of members are compared rather than the entire member set. If running a BXM, use the default of false. If running an IXM, set this option to true.	FALSE

mpxitob utility

The mpxitob utility is a weight generation program to generate bucket records from item records.

All options and flags are not case-sensitive; option values are not.

Table 58. mpxitob options

Option	Type	Description	Default
`-bxmInpDir`	dirName	`.bin` input directory	NONE
`-bktOutDir`	dirName	`.bin` output directory	NONE
`-nMxmParts`	N	Maximum number of input partitions	1
`-nBktParts`	N	Number of bucket output partitions	1
`-entType`	Name	Entity type name	NONE

mpxlink utility

The mpxlink utility is a cross match program that enables entity linkage.

The mpxlink utility takes comparison results from the mpxcomp utility and creates entity link and task files (`.unl` files) that can be loaded into the database. This utility can be run from the command line or from IBM Initiate Workbench**Initiate menu** > **New Job Set**. See *IBM Initiate Workbench User's Guide* for more information.

All options and flags are case independent; option values are not independent.

Generating task sets can be a lengthy operation.

If you want to retain existing Enterprise IDs (entrecnos) while doing an incremental cross match (IXM), you must use the correct options:

- `ixmmode`: Specify `-ixmmode` with mpxlink to ensure that entity sets are processed correctly. If `-ixmmode` is not specified, the mpxlink process starts with the current entity set. Setting `-ixmmode` causes mpxlink to re-evaluate all entity sets, preserving the previous Enterprise ID when all previous members are present in the new entity set.
- `bxmxeia`: This option is required if the existing Enterprise IDs are to be considered when entity sets are formed during the IXM.
- `noTskSets`: Specifying `-noTskSets` reduces utility run time without affecting task creation.
- `noTskRelatedMembers`: Specifying `-noTskRelatedMembers` also reduces utility run time.
- `entrecno`: Set an `-entrecno` value to a number higher than any currentrecno column value in any of the mpi_entlink tables in the system. If you do not set `-entrecno` in this way, you risk creating overlapping Enterprise IDs, which can result in new members incorrectly being added to an existing entity.
- `audrecno`: Set an `-audrecno` value to a number higher than any current audrecnos in the mpi_audhead tables. If you do not set `-audrecno` in this way, you risk creating inaccurate entity linkage history data.
- `audhead`: Specify `-audhead` in order to create an `mpi_audhead.unl` file as part of the mpxlink operation.

Table 59. mpxlink options

Option	Type	Description	Default
`-entType`	Name	Entity type name. This option identifies the type of entity being computed. If you are implementing multiple entity types (for example, identity and household), you must run mpxlink for each type. This option is required and there is no default setting.	NONE

Table 59. mpxlink options (continued)

Option	Type	Description	Default
`-bxmInpDir`	dirName	`.bin` input directory. The directory where the input binary (`.bin`) files to link are stored. Input files can be from the mpxcomp utility output, or other processes such as an IXM. This directory is typically the work directory on the server hosting your hub configuration. This option is required and there is no default setting. You can list multiple directories for this option; separate multiple directories with single spaces.	NONE
`-bxmOutDir`	dirName	`.bin` output directory. Indicate where you want the BXM output files to be located. This directory is relative to the projects work directory on the hub: `MAD_HOMEIDR\inst\ mpinet_`*`instance_name`*`\work\ `*`project_name`*`\work\ `*`bxm_output_dir`* Also generate bulk cross match data in the designated BXM output directory.	NONE
`-unlOutDir`	dirName	`.unl` output directory. The directory in which you want the mpxlink output binary files located. Binary output files are used by the relationship linkers. The binary output file is named `mpx_bxmxmem.bin`. This directory is typically relative to the work directory on the server hosting the hub configuration. Generating the output in binary form is optional; specifying an output directory with this option is what causes binary output to be generated. In other words, if no directory is specified here, no binary output is generated.	NONE

Table 59. mpxlink options (continued)

Option	Type	Description	Default
`-nMemParts`	N	Number of member partitions (MemParts). MemParts are used to partition the data set. Typically this partition is done for memory considerations. Because the mpxlink utility requires the entire input data set (for example, the binary files of comparison results) to be read into memory at once, breaking the data set into smaller pieces allows them to fit into available memory. The MemParts option differs from the MxmParts option in that MemParts breaks up the memHead and memCmpd data files, whereas MxmParts breaks up link and task files (the output of the mpxcomp utility). The MemParts value set here must be the same as the MemParts value set in mpxcomp, and in the utility that created the input for mpxcomp (for example, mpxfsdvd, mpxprep, or mpxredvd). In other words, the MemParts setting in mpxcomp determines how many partitioned file segments are passed to mpxlink; the mpxlink MemParts setting must accurately reflect the number of partitioned file segments coming from mpxcomp. There is a performance consideration to partitioning the data set: the higher the MemParts is set, the slower the mpxlink process. Leave this value set to 1 unless memory is an issue. The maximum value is 100.	1

Table 59. mpxlink options (continued)

Option	Type	Description	Default
`-nMxmParts`	N	Number of maximum out partitions. Like MemParts, the MxmParts option partitions the output of the mpxcomp process. As with MemParts, this option is used when the output file is too large to be read into memory in its entirety, and needs to be broken up into smaller sections in order to fit into available memory. The MxmParts option differs from MemParts in that MxmParts breaks up link and task files (the output of the mpxcomp utility), whereas MemParts breaks up the memHead and memCmpd data files. The MxmParts value set here must be the same as the MxmParts value set in mpxcomp, which provides the input to mpxlink. In other words, the MxmParts setting in mpxcomp determines how many partitioned file segments are passed to mpxlink. The mpxlink MxmParts setting must accurately reflect the number of partitioned file segments coming from mpxcomp. Leave this value set to 1 unless memory is an issue. The maximum value is 100.	1

Table 59. mpxlink options (continued)

Option	Type	Description	Default
-{no}bxmDiff		Use explicit different records from entrule. This option controls whether mpxlink uses existing entity rules when forming entities. For example, if two members in an entity are separated in IBM Initiate Inspector, a non-identity rule is created by the Master Data Engine. (Likewise if two members are manually linked, an identity rule is created.) The mpxrule utility captures these rules as "same" (identity) or "diff" (non-identity) rules. If you re-crossmatch an existing database, including these rules prevents the mpxlink utility from reforming linkages (in the case of diff rules), or force members to be in the same entity (in the case of a "same" rule). The input data used here is created with the corresponding mpxcomp -bxmDiff option (**Use explicit different records from entrule** option in IBM Initiate Workbench).	-noBxmDiff
-{no}bxmSame		Use explicit same records from entrule. Like -bxmDiff, this option controls whether the mpxlink utility uses existing entity rules when forming entities. See description for the -bxmDiff option. The input data used here is created with the corresponding mpxcomp utility -bxmSame option (**Use explicit same records from entrule** in IBM Initiate Workbench).	-noBxmSame
-{no}bxmXeia		Use implicit link records from entlink. This option instructs mpxlink to include the output from the mpxxeia utility. The mpxxeia utility captures existing entity data. The input data used here is created with the corresponding mpxcomp utility -bxmXeia option (**Use implicit link records from entlink** in IBM Initiate Workbench).	-noBxmXeia

Table 59. mpxlink options (continued)

Option	Type	Description	Default
-{no}bxmPD		Use potential duplicate task records from entxtsk. The mpxlink utility uses this data to form review identifier tasks that can be loaded into the database. The input data used here is created with the corresponding mpxcomp utility -bxmRvid (**Use reviewid records from mpxcomp** in IBM Initiate Workbench).	-noBxmPD
-{no}bxmPL		Use potential linkage task records from the mpxxtask utility (entxtsk), which captures existing task information from the database.	-noBxmPL
-{no}bxmRI		Use review identifier task records from the mpxxtask utility (entxtsk), which captures existing task information from the database.	-noBxmRI
-{no}bxmRule		Use member rule records from the mpxprep, mpxredvd, or mpxfsdvd utilities. Member rules express the relationship between the survivor and obsolete members in a merge. Because the input data used here is created by default in the mpxprep utility, it is not necessary to specify a corresponding option in mpxprep.	-bxmRule
-{no}bxmLink		Use linkage records from the mpxcomp utility. The mpxlink utility uses this data to form entities that can be loaded into the database. The input data used here is created with the corresponding mpxcomp utility -bxmLink option (**Use linkage records from mpxcomp** in IBM Initiate Workbench).	-bxmLink

Table 59. mpxlink options (continued)

Option	Type	Description	Default
-{no}bxmTask		Use task records from the mpxcomp utility. The mpxlink utility uses this data to form tasks that can be loaded into the database. The input data used here is created with the corresponding mpxcomp utility -bxmTask option (**Use task records from mpxcomp** in IBM Initiate Workbench).	-bxmTask
-{no}bxmRvid		Use review identifier records from the mpxcomp utility. The mpxlink utility uses this data to form review identifier tasks that can be loaded into the database. The input data used here is created with the corresponding mpxcomp utility -bxmRvid option (**Use reviewid records from mpxcomp** in IBM Initiate Workbench).	-bxmRvid
-{no}entLink		Instruct the engine to write new linkages and entity level tasks to a .unl file (mpi_entlink.unl).	-entLink
-{no}entXeia		Instruct the engine to write historical Enterprise ID data to a .unl file (mpi_entxeia.unl).	-entXeia
-{no}entXtsk		Instruct the engine to write information about tasks related to an entity to a .unl file (mpi_entxtsk.unl).	-entXtsk
-{no}seqGen		When specified, this option writes a .unl file containing updated sequence generator numbers that can then be loaded into the database. The engine normally updates this table properly on startup. This option is useful for an installation that, when doing multiple links, needs to update the sequence numbers without starting an engine.	-noSeqGen

Table 59. mpxlink options (continued)

Option	Type	Description	Default
`-{no}tskSets`		Compute full task set information. Assigns a task set number to a member in a task. A task set identifies a group (two or more) of records explicitly identified as being in a task. For example, if memrecnos 1, 2, and 3 are in a potential duplicate task, they are all assigned tskset=1; memrecno, If memrecnos 4 and 5 are in a Potential Linkage task, they are assigned tskset=2, and so on. This data is typically used for reporting purposes.	`-noTskSets`
`-{no}tskRelated Members`		Create a count of members in a task so that when you have a trigger member, you can tell that there are n members in the task. The count is only calculated when a member is cross matched.	`-tskRelatedMembers`
`-{no}strict`		Forces xeia (entity linkage) information to default to existing information (rules and prior data). Setting this option to `-strict` makes the mpxlink utility sensitive to anomalies in the data. Disable this option to instruct mpxlink to ignore anomalies in the data. For example, inconsistencies or discrepancies arising from live updates to the table. (That is, discrepancies that might occur because data is changing from updates as it is being collected by the mpx utilities that create the input for mpxlink.) This option is typically used for reporting purposes.	`-strict`
`-ixmMode`		Indicates IXM mode. Used for IXM only.	FALSE

Table 59. mpxlink options (continued)

Option	Type	Description	Default
-entRecno	N	Used with .unl only. The starting entity record number for the .unl. The option allows for specifying an entity record number to start with for the creation of the mpi_entlink_xx.unl file. The parameter is optional. If not set, then the mpxlink utility defaults to applying 1 as the starting entity record number.	1
-tskRecno	N	Used with .unl only. Allows for specification of a starting task record number in the .unl file. This option reads the tskrecno from the mpi_seqgen table.	mpi_seqgen. tskrecno
-audRecno	N	Used with .unl only. Common audRecno for all .unl files. This option sets the audit record number for the .unl files that are loaded into the mpi_audhead database table. When the -{no}audHead option (**Write mpi_audhead.unl** in IBM Initiate Workbench) is enabled, you can set the -audRecno option to an existing mpi_audhead record number.	2
-usrRecno	N	Used with .unl only. Common usrRecno for all .unl files. This option sets the user record number for the .unl files that are loaded into the mpi_audhead database table. When the -{no}audHead option (**Write mpi_audhead.unl** in IBM Initiate Workbench is enabled, you can set this option to an existing mpi_usrhead user record number.	1

Table 59. mpxlink options (continued)

Option	Type	Description	Default
`-ixnRecno`	N	Used with `.unl` only. This setting is the ixnRecno for audhead record. This option sets the transaction record number for the `.unl` files that are loaded into the mpi_audhead database table. When the `-{no}audHead` option (**Write mpi_audhead.unl** in IBM Initiate Workbench is enabled, you can set this option to an existing mpi_ixnhead user record number.	71
`-evtTypeno`	N	Used with `.unl` only. This setting is the evtTypeno for the audhead record. Use this option to specify an event type for the audhead records. When the `-{no}audHead` option (**Write mpi_audhead.unl** in IBM Initiate Workbench is enabled, you can set this option to an existing mpi_evttype event type number.	0
`-{no}audHead`		Used with `.unl` only. Writes `mpi_audhead.unl` file, and uses the audrecno specified in the `-audRecno` option. (Common audit record number for all `.unl` option.) This option is commonly used in new implementations where no audit records exist yet.	`-noAudHead`
`-bktOutDir`	dirName	Used with NTE only; the output directory for the BXM files.	NONE

Table 59. mpxlink options (continued)

Option	Type	Description	Default
-entBktd		Used with NTE only. Write entBktd information. This option and the -bktOutDir used together allow the mpxlink utility to generate a binary bucket file that is consumed by the mpxcomp utility to rescore members that exist in the same transitive entity. This is leveraged for non-transtive entities to get scores between members who were brought together by a "glue" member and would not have a score generated by our traditional binary bucket file generated during the mpxprep or mpxfsdvd process. This setting allows a second pass using the mpxcomp and mpxlink utilities to produce accurate non-transitive entities. Although non-transitive entities can be produced with a single pass through the mpxcomp and mpxlink utilities, the two-pass approach improves accuracy.	FALSE

mpxpair utility

The mpxpair utility is a cross match program used to generate random pairs for weight generation.

All options and flags are not case-sensitive; option values are not.

Either –memType or –entType must be specified.

Table 60. mpxpair options

Option	Type	Description	Default
-bxmInpDir	dirName	Directory containing mpi_memhead.bin files	NONE
-bktOutDir	dirName	Directory to which random pairs are written	NONE
-nBktParts	N	Number of bucket partitions	1
-npairs	N	Number of random pairs to generate; suggest 10 million	NONE
-memType	Name	Member type name	NONE
-entType	Name	Entity type name	NONE

mpxprep utility

The mpxprep utility is a cross match program used to generate bulk cross match (BXM) data.

This utility can be run from the command line or from IBM Initiate Workbench **Initiate menu** > **New Job Set**. See the *IBM Initiate Workbench User's Guide*.

All options and flags are case independent; option values are not.

For incremental cross match (IXM), use `–srcRecno` or `–[min/max]MaudRecno`, but not both.

`-memType` and `-entType` cannot be specified at the same time.

Table 61. mpxprep options

Option	Type	Description	Default
`-bxmOutDir`	dirName	Directory where you want any `.bin` output files located. This directory is relative to the instance work directory on the server hosting the hub.	NONE but required
`-{no}bxmBktd`		Generate MEMBKTD output	`-bxmBktd`
`-{no}bxmCmpd`		Generate MEMCMPD output	`-bxmCmpd`
`-{no}bxmQryd`		Generate MEMQRYD output. This option is for use with the relationship linker and instructs the mpxprep utility to create BXM files containing query data. The relationship types, attributes, and rules should already be defined so that mpxprep knows what data to include in the BXM file.	`-bxmQryd`

Table 61. mpxprep options (continued)

Option	Type	Description	Default
`-nMemParts`	N	Number of member partitions. Member partitions (MemParts) are used to partition up the data set. Typically option is set because of memory considerations. Because the mpxlink utility requires the entire input data set (for example, the binary files of comparison results) to be read into memory at once, breaking the data set into smaller pieces allows them to fit into the available memory on the server. MemParts breaks up the memHead and memCmpd data files. If you set a value other than 1 here, you must set a matching MemParts value for any downstream utility that uses the output of the mpxprep utility (such as mpxcomp or mpxlink). In other words, the MemParts setting in those downstream utilities must accurately reflect the number of partitioned file segments coming from the mpxprep utility. Leave this value set to 1 unless memory is an issue.	1
`-nBktParts`	N	Number of bucket partitions. Like the `-nMemParts` option (**Maximum number of Member Partitions** in IBM Initiate Workbench), the maximum number of bucket partitions option partitions the output of the mpxprep process. As with MemParts, this option is used when the output file is too large to be read into memory in its entirety, and needs to be broken up into smaller sections in order to fit into available memory. BktParts differs from MemParts in that it breaks up the membktd data. This option is the most common one used for reducing your memory footprint (and can also help sort performance on large data sets).	1
`-minBktTag`	N	Minimum bucket tag to use (0=any). This setting specifies the lowest bucketing role to be included in the operation.	0
`-maxBktTag`	N	Maximum bucket tag to use (0=any). This setting specifies the highest bucketing role to be included in the operation.	0

Table 61. mpxprep options (continued)

Option	Type	Description	Default
`-nQryParts`	N	Number of query partitions. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the hub. Leave this setting at the default unless you need the memory. This option is enabled only when the option to generate query BMX (`-bxmQryd`) is also enabled.	1
`-minQryRole`	N	Minimum query role to use (0=all). The lowest query role designation used in the algorithm to include in the process. This option is enabled only when the option to generate query BMX (`-bxmQryd`) is also enabled.	0
`-minMemRecno`	N	Minimum memRecno filter (0=any). Specifies the lowest MEMRECNO to include in the process.	0
`-maxMemRecno`	N	Maximum memRecno filter (0=any). Specifies the highest MEMRECNO to include in the process.	0
`-blkSize`	N	Bulk size (number of members)	1000
`-buffSize`	N	Size (in bytes) for each file input and output (I/O) buffer	65536
`-memType`	Name	Member type name. If you have multiple member types in the hub database and need to generate BXM data for only one of those member types, the Member Type filter can be used. All entity types for that member are processed.	NONE
`-entType`	Name	Entity type name	NONE
`-ixmMode`		Used with IXM only. Indicates IXM mode	FALSE
`-minMaudRecno`	N	Used with IXM only. IMinimum audRecno filter (0=any)	0
`-maxMaudRecno`	N	Used with IXM only. IMaximum audRecno filter (0=any)	0
`-srcRecno`	N	Used with IXM only. IsrcRecno filter (0=any)	0

mpxrebkt utility

The mpxrebkt utility regenerates bucket data into the database, or `.unl` files, or bulk cross match (BXM) files.

All options and flags are case independent, option values are not.

`-dbupdate` and, or `-unlOutdir` `-unlOutSegs` and, or `-bxmOutDir` must be specified.

-memType and -entType cannot be specified at the same time.

Table 62. mpxrebkt options

Option	Type/Argument	Description	Default
-unlOutDir	dirName	Output directory for .unl files	NONE
-unlOutSegs	segList	The segments to output in the .unl file	NONE
-encoding	encoding	Encoding of .unl files; options are LATIN1, UTF8, or UTF16	LATIN1
-bxmOutDir	dirName	.bin output directory	NONE
-{no}bxmBktd		Generate MEMBKTD output	-bxmBktd
-{no}bxmCmpd		Generate MEMCMPD output	-bxmCmpd
-nMemParts	N	Number of member partitions	1
-nBktParts	N	Number of bucket partitions	1
-minBktTag	N	Minimum bucket tag (0 = any)	0
-maxBktTag	N	Maximum bucket tag (0 = any)	0
-nQryParts	N	Number of query partitions	1
-minQryRole	N	Minimum query role to use (0=all)	0
-minMemRecno	N	Minimum memrecno at which to start the rebucket (0 = any)	0
-maxMemRecno	N	Maximum memrecno at which to end the rebucket (0 = any)	0
-blkSize	N	Number of members in a block	1000
-buffSize	N	Size for each input and output (I/O) buffer	65536
-{no}dbUpdate		Database update	-noDbUpdate
-memType	Name	Member type name	NONE
-entType	Name	Entity type name	NONE

mpxredvd utility

The mpxredvd utility regenerates derived data into the database, or .unl files, or bulk cross match (BXM) files.

This utility can be run from the command line or IBM Initiate Workbench**Initiate menu** > **New Job Set**. See the *IBM Initiate Workbench User's Guide*.

All options and flags are case independent, option values are not.

-dbupdate and, or -unlOutdir -unlOutSegs and, or-bxmOutDir must be specified.

-memType and -entType cannot be specified at the same time.

Table 63. mpxredvd options

Option	Type	Description	Default
`-unlOutDir`	dirName	`.unl` file output directory. The output of the mpxredvd utility is the derived data segments (comparison, bucket and, optionally, query data), which have their own `.unl` files (`mpi_memcmpd`, `mpi_membktd`, and `mpi_memqryd`) written to the directory specified here. This directory is relative to the project work directory on the hub: `MAD_HOMEDIR\inst\ mpinet_`*`instance_name`*`\work\ `*`project_name`*`\work\`*`unl_output_dir`* Use `-unlOutDir` with`-unlOutSegs` to indicate whether the mpxredvd utility should generate `.unl` files during processing. Also with `-unlOutSegs`, create `.unl` files containing bucketing data or comparison data, or instruct mpxredvd to generate query `.unl` files during processing (the files are used by the relationship linker).	NONE
`-unlOutSegs`	segList	List of segments to include in the output. Use with `-unlOutDir` to indicate whether mpxredvd should generate `.unl` files during processing. Also with `-unlOutDir`, creates `.unl` files containing bucketing data or comparison data, or instructs mpxredvd to generate query `.unl` files during processing (the files are used by the relationship linker).	NONE
`-encoding`	encoding	Encoding of `.unl` files; options are LATIN1, UTF8, or UTF16	LATIN1
`-bxmOutDir`	dirName	`.bin` output directory. Indicates where you want the BXM output files to be located. This directory is relative to the project work directory on the hub: `MAD_HOMEDIR\inst\ mpinet_`*`instance_name`*`\work\ `*`project_name`*`\work\`*`bxm_output_dir`	NONE
`-{no}bxmBktd`		Generate MEMBKTD output. In combination with `-{no}bcmCmpd`, instructs mpxredvd to generate output files for bulk cross matching.	`-bxmBktd`

Table 63. mpxredvd options (continued)

Option	Type	Description	Default
-{no}bxmCmpd		Generate MEMCMPD output. In combination with -{no}bxmBktd, instructs mpxredvd to generate output files for bulk cross matching.	-bxmCmpd
-{no}bxmQryd		Generate MEMQRYD output. Indicates whether mpxfsdvd should generate query BXM files during processing. These files are used by the relationship linker.	-bxmQryd
-nMemParts	N	Number of member partitions. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the hub. The utility that consumes the mpxredvd output (such as the mpxfreq utility) must use a matching MemPart value. Use the default unless you need the memory. The higher the member partitions the slower your mpxcomp process, as the hub must do more duplicate comparisons.	1
-nBktParts	N	Number of bucket partitions. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the hub. Use the default unless you need the memory.	1
-minBktTag	N	Minimum bucket tag (0=any). The lowest bucketing role designation used in the algorithm to include in the process.	0
-maxBktTag	N	Maximum bucket tag (0=any). The highest bucketing role designation used in the algorithm to include in the process.	0
-nQryParts	N	Number of query partitions. Setting this partition depends on the size of your data set, your algorithms, and how much memory you have access to on the hub. Use the default unless you need the memory. This option is enabled only when the option to Generate query BXM is also enabled.	1
-minQryRole	N	Minimum query role to use (0=all). The lowest query role designation used in the algorithm to include in the process. This option is enabled only when the option to Generate query BXM is also enabled.	0

Table 63. mpxredvd options (continued)

Option	Type	Description	Default
`-minMemRecno`	N	Minimum memrecno at which to start the re-derivation	0
`-maxMemRecno`	N	Maximum memrecno at which to end the re-derivation	0
`-blkSize`	N	Number of members in a block	1000
`-buffSize`	N	Size for each input and output (I/O) buffer	65536
`-{no}dbUpdate`		Database update	`-nodbUpdate`
`-memType`	memName	Member type name. If you have multiple member types in the hub database and need to derive data for only one of those member types, select the member type here; otherwise, select ALL.	NONE
`-entType`	entName	Entity type name	NONE

mpxrule utility

The mpxrule utility is a cross match program that generates bulk cross match (BXM) data.

The mpxrule utility is used by the incremental cross match (IXM) process and you need to run this program only for IXM.

All options and flags are case independent; option values are not.

Table 64. mpxrule options

Option	Type	Description	Default
`-entType`	Name	Entity type name	NONE
`-bxmOutDir`	dirName	`.bin` output directory	NONE
`-{no}bxmDiff`		Write non-identity information from database	`-bxmDiff`
`-{no}bxmSame`		Write identity information from database	`-bxmSame`

mpxsmooth utility

The mpxsmooth utility is used during the weight generation process.

Previously when weights were generated by the weight generation process, there was almost always a need for manually editing the weights to make them optimal for the implementation. This process is called weight "smoothing". Smoothing is accomplished by reading the matched and unmatched sampling binary data generated by the mpxcomp utility. The utility reads this binary data and applies any necessary adjustments in order to ensure a smoothed output. The data is considered smoothed when the sample data is monotonically decreasing for msamps (matched samples) and increasing for usamps (unmatched samples). This output can then be used with the mpxwgts utility to generate the smoothed

weights used by the engine. The smoothing process applies only to data that represents one-dimensional (1DIM), two-dimensional (2DIM), and three-dimentional (3DIM) weights.

The reason the sampling data files are generally non-monotonically increasing and decreasing is due to irregular counts in the unmatched and matched samples where they are not expected. Fixing the discrepancy before running the weight generation step where the final weights are computed ensures smoothed weights. The process begins by setting all input values to wgtFLR if they are less than this value. Next, each value is compared with its nearest neighbor to determine if an adjustment must be made. If an adjustment is needed, the slope (derivative) and the mid point of these two values are calculated. The result of these calculations is used to correct the current and next sampling data value. This process is repeated until the data monotonically increases or decreases.

The mpxsmooth utility eliminates the need for manual intervention. The weight generation utility, which is started from IBM Initiate Workbench or by using the `madconfig generate_weights` target, automatically runs this utility. The only time you might run mpxsmooth is when you already have the matched sample and unmatched sample binary files from a previous weight generation and want to smooth them. Manually running mpxsmooth requires that you also run the mpxwgts utility.

If the `bxmOutDir` option is not provided, then a view of the before and after smoothing process is dumped to stdout (console). Both `Usamp` and `Msamp` options can be provided at the same time. Remember that this utility replaces the original binary and text files if the `bxmOutDir` setting is the same as `bxmInpDir`.

Table 65. mpxsmooth options

Options and targets	Description	Default
`entType`	Identifies the entity type for which the sampling data applies.	
`bxmInpDir`	This setting is the directory containing the matched (msamp) and unmatched (usamp) binary files.	
`bxmOutDir`	This setting is the directory you want the smoothed matched and unmatched files written to.	
`slopeReduction`	This parameter specifies the slope rate of change between data points. The slope calculation is inversely proportional to this value. This means that larger values result in smaller rates of change. The default value is 3.	3
`Usamp`	This option directs the utility to smooth the unmatched samples file.	
`Msamp`	This option directs the utility to smooth the matched samples file.	

mpxsort utility

The mpxsort utility is used to reorder a binary file generated from the bulk cross match (BXM) and incremental cross match (IXM) utilities.

Specifically, mpxsort reorders the bxmlink file when there are multiple member parts or multiple threads used during the creation of the bxmlink file. This sort order is required by the non-transitive logic to keep the transitive entity sets grouped together so that members can be removed from the set (and possibly form additional entities) for the non-transitive phase.

The mpxsort utility is run between the second mpxcomp and mpxlink phase. The input to mpxsort is the output of the mpxcomp utility. When using the mpxsort utility, match the number of parts (`-mpxparts`) with the number of parts specified for the mpxcomp utility. The mpxsort output is then consumed by the mpxlink utility.

A command-line parameter unique to mpxsort is the `-{no}radix` sort option. A radix sort, also known as a binary sort, is an extremely fast method of sorting binary records. While a radix sort is faster than a quick sort (which is our default sorting algorithm), the radix sort consumes twice as much memory as a quick sort. On servers where memory is a constraint, the `-noradixsort` option can be specified and a quick sort is used to conserve memory. On servers where memory is not an issue and maximum performance is required, the default `-radixsort` option can be used.

Again, the mpxsort utility supports only bxmlink files which are the output of the mpxcomp utility. I

Usage example:

```
mpxsort -enttype hh -bxmlink -bxminpdir /bxminp -bxmoutdir /bxmout
```

This example sorts the `mpx_bxmlink_xx.XXX` file for the household (hh) entity type.

All options and flags are case independent; option values are not.

`-nthreads` option defaults to the number of processors on the server.

Table 66. mpxsort options

Option	Type	Description	Default
`-entType`	name	entity type name	NONE
`-bxmInpDir`	dirName	`.bin` file input directory	NONE
`-bxmOutDir`	dirName	`.bin` file output directory	NONE
`-nMxmParts`	N	Number of maximum partitions. Match this setting to the number of parts specified in the output of the BXM utility used to generate the file being used as input to the mpxsort utility.	1
`-nThreads`	N	Number of threads	the number of CPUs
`-{no}bxmLink`		Use linkage records from the mpxcomp utility. Currently, the mpxsort utility supports only bxmlink files which are the output of the mpxcomp utility. Use the `-bxmLink` to avoid errors.	`-nobxmLink`
`-{no}radixSort`		Use quick sort instead of radix sort.	`radixSort`

mpxstd utility

The mpxstd utility is a diagnostic utility that displays the standardized output from various standardization routines.

This command always reads from `stdin` to get the strings for standardization.

You must choose one of `addr`, `bxnm`, `pxnm`, `phone2`, `email`, or `dump` options.

Table 67. mpxstd options

Option	Type	Description	Default
`-addr`		Show address standardization	NONE
`-bxnm`		Show business name standardization	NONE
`-pxnm`		Show person name standardization	NONE
`-phone2`		Show phone standardization (as used by the PHONE2 standardization function)	NONE
`-email`		Show email standardization	NONE
`-dump`		Dumps the string tables used for standardization. The output can be large.	NONE

mpxwgts utility

The mpxwgts utility generates weight tables.

All options and flags are case independent; option values are not.

If –`bootwgts` option is used, frequency files are might be required based on the comparison functions used.

If –`bootwgts` option is used, the matched sample file is not needed.

If –`bootwgts` option is used, frequency files are required.

Table 68. mpxwgts options

Option	Type	Description	Default
`-entType`	Name	Entity type name	NONE
`-frqInpDir`	dirName	Frequency input directory	NONE
`-uSampFile`	fileName	Unmatched sample (usamp) file	NONE
`-mSampFile`	fileName	Matched sample (msamp) file	NONE
`-wgtOutDir`	dirName	Weight results output file	NONE
`-bootwgts`		Generate bootstrap weight tables	FALSE
`-audRecno`	number	Use a specific audit record number (audRecno) for auditing	2
`-bootfrqa`		Generate similarity weight tables	FALSE

mpxxeia utility

The mpxxeia utility is a cross match program used to unload existing entity linkage data from the database.

The utility unloads contents of the `mpi_entxeia_enttype` table to binary format. If the mpxlink utility is run in IBM Initiate® Workbench with the "Force xeia information to default to existing information" option enabled, run the mpxxeia utility first. See the *IBM Initiate Workbench User's Guide*. Used by the incremental cross match (IXM) process.

All options and flags are case independent; option values are not.

You only need to run this program for IXM.

Table 69. mpxxeia options

Option	Type	Description	Default
`-entType`	Name	Entity type name. This option identifies the type of entity being computed. If you are implementing multiple entity types (for example, identity and household), you must run mpxlink for each type. This option is required and there is no default setting.	NONE
`-bxmOutDir`	dirName	`.bin` file output directory. The directory in which you want the mpxxeia output binary files located. Binary output files are used by the relationship linkers. This directory is typically relative to the work directory on the server hosting the hub, such as: `MAD_HOMEDIR\inst\ mpinet_instance_name\work\ project_name\work\bxm_output_dir` Generating output in binary form is optional. Specifying an output directory with this option is what causes binary output to be generated. If no directory is specified here, no binary output is generated.	NONE
`-{no}bxmXeia`		Write entity record number (entRecno) information from database	`-bxmXeia`

mpxxtsk utility

The mpxxtsk utility is a cross match program that unloads existing task data.

This utility is used by the incremental cross match (IXM) process. You need to run this program for IXM only.

All options and flags are case independent; option values are not.

Table 70. mpxxtsk

Option	Type	Description	Default
`-entType`	Name	Entity type name	NONE
`-bxmOutDir`	dirName	`.bin` file output directory	NONE
`-{no}bxmPD`		Write potential duplicate (PD) task information from database	`-bxmPD`
`-{no}bxmPL`		Write potential linkage (PL) task information from database	`-bxmPL`
`-{no}bxmRI`		Write review identifier (RI) task information from database	`-bxmRI`

Chapter 13. Configuring SSL

By default, engine instances are not configured to use SSL communication. To configure SSL communication, you must complete this procedure.

Before you begin

Review the SSL security topic.

Procedure

1. Procure production certificates from a certificate authority (CA). For information about creating `.pem` files for a production certificate, see the OpenSSL documentation at http://www.openssl.org/.
2. Import certificate data into the applicable runtime environments.
3. Configure the instance SSL environment variables in the configuration files.

   Two-way SSL communication is enabled with useSSL variable set to true within the `com.initiate.server.net.cfg` file and the `javax.net.ssl.keystore` and `javax.net.ssl.trustStore` variables set within the `com.initiate.server.system.cfg` file.
4. Restart the engine instance.

   Important: To complete the two-way SSL configuration, the clients such as IBM Initiate Workbench or IBM Initiate Inspector also must be configured for two-way SSL communication.

SSL security

Socket Layer Security (SSL) certificates enable secure, encrypted communication between the IBM Initiate Master Data Service software and clients.

An SSL certificate is made up of a public key that is used to encrypt information and a private key that is used to decipher the encrypted information. A virtual "handshake" authenticates the server to the client and syncs the encryption methods and keys that are used to transmit information. Security is further enhanced by session renegotiation to ensure that the same encryption key is not used for a persisted connection.

There are two options for certificate presentation: one-way SSL and two-way SSL.

In a one-way SSL configuration, the server must present a certificate to the client, but the client is not required to present a certificate to the server. To successfully negotiate an SSL connection, the client must authenticate the server. However, the server accepts any client into the connection. With one-way SSL, the client is required to provide its user name and password credentials to the Master Data Engine when executing an interaction. One-way SSL is common on the Internet where customers want to create secure connections before sharing personal data.

With two-way SSL, the server presents a certificate to the client, and the client also presents a certificate to the server. Two-way SSL enables "trusted user" behavior. Once the certificates have been passed between server and client, the Master Data Engine does not validate the password provided in the MPINET protocol. By

enabling two-way SSL, the IBM Initiate Master Data ServiceAPIs can be used inside of an application server such that authentication can be deferred to the application server. In this case, the Master Data Engine performs interactions as that user without authentication.

All IBM Initiate Master Data Service clients support SSL communication.

Important: If you enable SSL communication for IBM Initiate Master Data Service clients, you must also enable SSL for communication with LDAP. Enabling or disabling SSL is a configuration-wide setting. It must be enabled for IBM Initiate Master Data Service clients and for LDAP, or for neither.

Libraries required to enable SSL are installed with the Master Data Engine. Certificate data is stored in `.pem` files. For example, the default IBM Initiate Master Data Service certificate is stored in the `ibmcorporationcert.pem`, with its private key data stored in `ibmcorporationkey.pem`. Both `.pem` files are located in the engine installation `\conf` directory. For example:

Microsoft Windows: `C:\Program Files\IBM\Initiate\Engine10.0.0\conf`

IBM AIX, Linux, or Solaris: `/opt/IBM/Initiate/Engine10.0.0/conf/`

Table 71 describes the SSL environment variables set in the `com.initiate.server.system.cfg` file. This file is written to the instance `conf` directory at instance creation time. For example:

Microsoft Windows: *MAD_HOMEDIR*`\inst\mpinet_`*name*`\conf`

IBM AIX, Linux, or Solaris: `/`*MAD_HOMEDIR*`/inst/mpinet_`*name*`/conf`

where *MAD_HOMEDIR* is the full path to the directory created for the associated runtime instances (for example, `prod` or `qa`), and *name* is the engine instance name.

By default, engine instances are not configured to use SSL communication.

Table 71. SSL environment variables in the Engine configuration files

Variable and description	Value set at instance creation and guidelines
`useSSL` Configures JMX communication profile. Set within the `com.initiate.server.jmx.jmxmp.cfg` file.	`false` To configure SSL, change this value to: `true`.
`useSSL` Indicates whether to use SSL communication*. Set within the `com.initiate.server.net.cfg` file.	`false` To configure SSL, change this value to `true`.

Table 71. SSL environment variables in the Engine configuration files (continued)

Variable and description	Value set at instance creation and guidelines
javax.net.ssl.keystore IBM Initiate Master Data Service keystore file (collection of keys and certificates); typically, the implementation-specific files are stored in the /conf directory of the Master Data Engine software. For example: **Microsoft Windows** C:\Program Files\IBM\Initiate\Engine10.0.0\conf **IBM AIX, Linux, or Solaris** /opt/IBM/Initiate/Engine10.0.0/conf Set within the com.initiate.server.system.cfg file.	${mad.root.dir}/conf/ibmcorporation.p12
javax.net.ssl.trustStore The ibmcorporationtrust.jks file (trusted entities and certificates); this file is usually the \conf folder in the engine installation MAD_ROOTDIR directory. Set within the com.initiate.server.system.cfg file.	${mad.root.dir}/conf/ibmcorporationtrust.jks
javax.net.ssl.keyStorePassword Specifies the password for the keystore. Set within the com.initiate.server.system.cfg file.	rmi+ssl
javax.net.ssl.trustStorePassword Specifies the password for the truststore. Set within the com.initiate.server.system.cfg file.	rmi+ssl
javax.net.ssl.keyStoreType Indicates keystore type. Set within the com.initiate.server.system.cfg file.	PKCS12
javax.net.ssl.trustStoreType Indicates truststore type. Set within the com.initiate.server.system.cfg file.	JKS

*Two-way SSL communication is enabled with useSSL set to true within the com.initiate.server.net.cfg file and the javax.net.ssl.keystore and javax.net.ssl.trustStore variables set within the com.initiate.server.system.cfg file. To complete the two-way SSL configuration, the clients such as IBM Initiate Workbench or IBM Initiate Inspector also must be configured for two-way SSL communication.

Sample com.initiate.server.system.cfg configured for SSL

The com.initiate.server.system.cfg contains most of the SSL configuration variables.

```
...
javax.net.ssl.keyStore=${mad.root.dir}/conf/ibmcorporation.p12
javax.net.ssl.trustStore=${mad.root.dir}/conf/ibmcorporation.jks
javax.net.ssl.keyStorePassword=rmi+ssl
javax.net.ssl.trustStorePassword=rmi+ssl
javax.net.ssl.keyStoreType=PKCS12
javax.net.ssl.trustStoreType=JKS
...
```

Chapter 14. Configuring globalization of the Master Data Engine

IBM Initiate Master Data Service can be configured to run in languages other than U.S. English.

About this task

To accomplish alternate language implementation of the IBM Initiate Master Data Service software, interaction messages from the Master Data Engine to a client are translated. Internal message logs are not translated because typically the messages are only used by IBM Software Support.

Use this procedure to configure the engine for a language other than U.S. English.

Procedure

1. Configure Unicode settings for your database before installing the engine.
2. Set the default language. This step is done during the creation of the engine instance.
3. Set the `MAD_ENCODING` variable. This variable is not set during instance creation and must be manually set in the `com.initiate.server.system.cfg` configuration file.

   Attention: Log files created by the Master Data Engine are in ASCII encoding. Code points not encompassed by ASCII are in the standard Unicode form of U+XXXX.

Database prerequisites for using Unicode in the Master Data Engine

Unicode enables the Master Data Engine to process customer data in any language and allows a single Master Data Engine instance to store data in multiple languages. The Master Data Engine supports UTF16, UTF8, and ISO-8859-1 (Latin1) (Cp1252) encoding.

You must have these database requirements in place for Unicode:

- SQL Server: new MAD_DBTYPE is “mssqlu”
- Oracle: CREATE DATABASE *dname*...CHARACTER SET AL32UTF8. You must also set the character length semantics for Unicode. Set the variable NLS_LANG_SEMANTICS to CHAR (the default setting is BYTE). Use the command:
  ```
  ALTER SYSTEM SET NLS_LENGTH_SEMANTICS=CHAR SCOPE=BOTH
  ```
  If you are using a non-wire connect driver with an Oracle client, you must also set this variable for the user connecting to the hub.
  ```
  NLS_LANG=AMERICAN_AMERICA.AL32UTF8
  ```
- DB2: CREATE DATABASE *dname* USING CODESET UTF-8 TERRITORY *territory code*. For example: `create database prod using codeset UTF-8 territory us`, where `prod` is the database name and `us` is the territory.

Default language setting for the Master Data Engine

Translated strings are stored in the \smt directory. These files, such as fr_FR.smt or en_US.smt, contain the interaction messages for return to clients.

To configure the software, the environment variable MAD_SMTLIST must be set in the com.initiate.server.system.cfg configuration file. This variable points to the appropriate *.smt file. This variable is typically set when you respond to the madconfig utility locale prompt during Master Data Engine instance creation.

Environment variable	Description	Default
MAD_SMTLIST	Comma-separated list of language (SMT) codes	Optional; default = "us_EN' for U.S. English

While setting the MAD_SMTLIST option to multiple languages (smtcode), the Master Data Engine can potentially load multiple languages (strings) at one time. However, the IBM Initiate Master Data Service components display the strings for only one language at a time. For example, the same Master Data Engine is configured to send a French client French messages while sending an English client English messages.

If client software is not configured to use an alternate language, only Master Data Engine level information is returned in the chosen language. Translation or globalization of the data stored in the Master Data Engine database, such as dates, is not converted when displayed in IBM Initiate Master Data Service clients. Rather, this information displays in the locale in which it was received from the source.

Appendix A. LDAP Directory Server for the Master Data Engine

All authentication to the Master Data Engine is done through an LDAP directory server installed during the normal engine installation process.

There are multiple ways to set up a Master Data Engine environment with the bundled LDAP directory server. You can embed the IBM Initiate LDAP servers in the Master Data Engine itself or configure a stand-alone server. The IBM Initiate LDAP server can be internally managed by using IBM Initiate Workbench. A corporate LDAP server is externally managed through your LDAP management tool. Some of the possible configurations are described in this topic.

All information and instructions regarding LDAP assume that you have a basic understanding of the configuration and support of LDAP.

Attention: User and group management is done in IBM Initiate Workbench. For details, see *IBM Initiate Workbench User's Guide*.

IBM Initiate LDAP configurations:

- Embedded LDAP - The LDAP server is part of the Master Data Engine server. The embedded server is created by using the madconfig utility create_instance target.
- Stand-alone LDAP - The stand-alone LDAP server exists externally to the Master Data Engine. During instance creation, you respond 'n' to the embedded option and then point to the existing LDAP server when prompted. If you have not created the stand-alone LDAP servers, create them by running the madconfig utility create_ldap target before creating your engine instance and starting the Master Data Engine.

Restriction: The embedded LDAP directory server is not intended for advanced deployment scenarios. For example, you cannot enforce corporate password policy (such as password format or expiration rules) through an embedded LDAP. If your implementation requires password policy enforcement, you must use an external corporate LDAP server.

Corporate LDAP configuration:

- By using the corporate LDAP option you can leverage your existing security, like your corporate directory, to manage users and groups that access the IBM Initiate Master Data Service components.

Restriction: Because the Master Data Engine manages users and groups jointly, all users and groups must be maintained together either on an IBM Initiate provided LDAP server or on a separate corporate LDAP server. The Master Data Engine does not support maintaining users on the IBM Initiate provided LDAP server with groups on the corporate LDAP server, or vice versa.

Combination configurations:

There are some combined configurations that are supported.

- Embedded IBM Initiate LDAP and stand-alone combination: The Master Data Engine server has an embedded LDAP server and can also be connected to a stand-alone IBM Initiate LDAP server.
- Embedded Initiate LDAP and corporate combination: The Master Data Engine server has an embedded IBM Initiate LDAP server and is also connected to a corporate LDAP server.
- Stand-alone IBM Initiate LDAP and corporate combination: The Master Data Engine server uses a stand-alone IBM Initiate LDAP server and is also connected to a corporate LDAP server.
- Cluster: Implementations with multiple Master Data Engine instances often use a cluster of embedded and stand-alone LDAP servers for high availability and disaster recovery (HA/DR). Clustering is only supported among IBM Initiate provided directory servers (embedded or stand-alone). Clustering is not supported among corporate directory servers or if any of the components is a corporate LDAP server (external).

Important: All implementations must enable either the embedded or stand-alone IBM Initiate LDAP directory server, even if you are using a corporate LDAP for your main user and group repository. This restriction is because the Master Data Engine needs to define internal system users and groups that are required for the engine to operate. If you use a corporate directory server, the Master Data Engine communicates with its directory server by using the LDAP protocol to request user and group information. The Master Data Engine supports the use of any LDAPv3 compliant server.

The illustrations in this topic show three basic examples of Master Data Engine configurations. In these graphics:

- ADS = Apache Directory Studio
- While only IBM Initiate Inspector is shown, all IBM Initiate Master Data Service web applications, including IBM Initiate Enterprise Viewer and IBM Initiate Web Reports, communicate directly with the engine through MPINET.
- Master Data Engine 2 can treat the embedded server on Engine 1 as a stand-alone instance

The first graphic depicts a configuration that uses both an embedded and a stand-alone directory server.

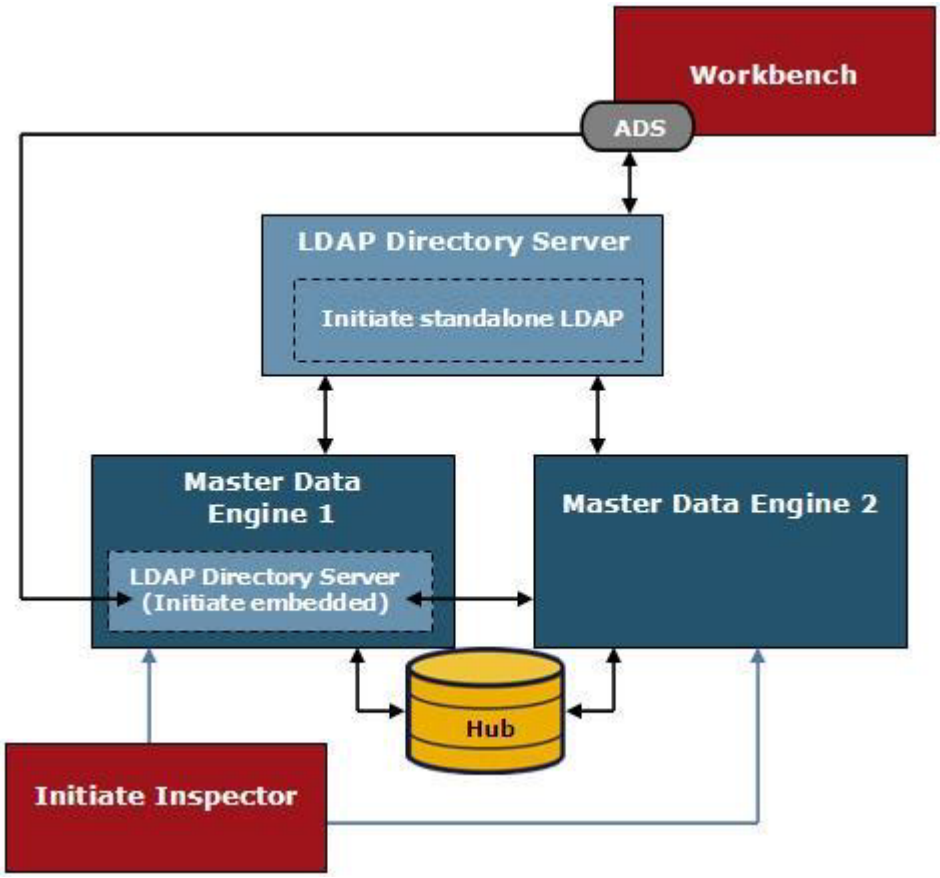

Figure 3. IBM Initiate embedded and stand-alone directory servers

The next graphic shows a configuration that uses both an embedded LDAP directory server with an external corporate directory server.

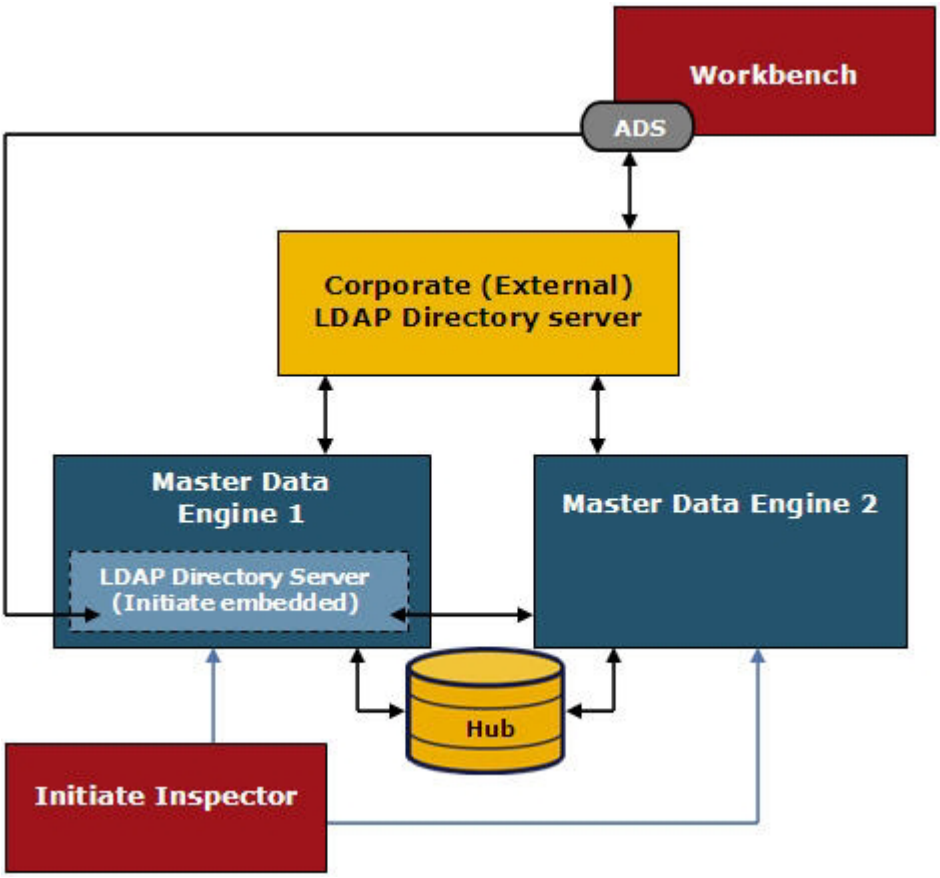

Figure 4. Embedded LDAP directory server with corporate (external) directory server

The configuration shown in this graphic uses an embedded and a stand-alone directory server along with an external directory server.

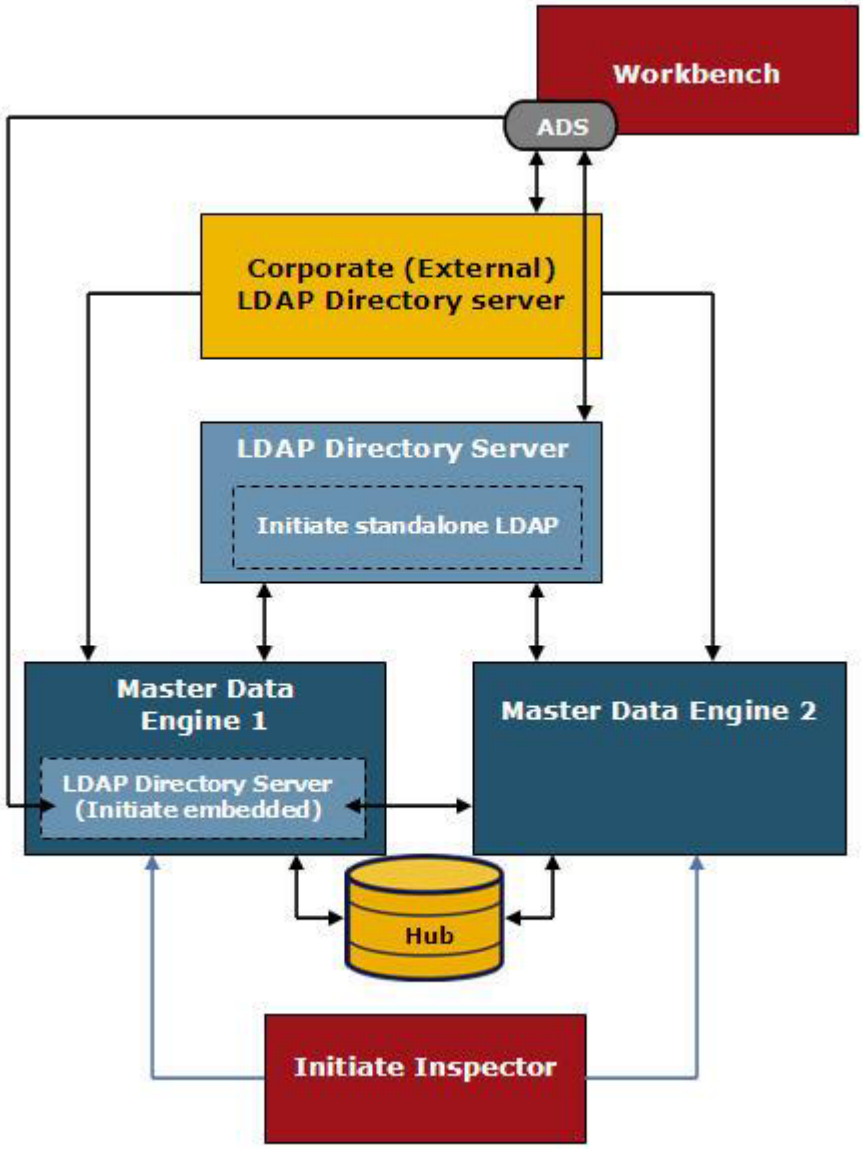

Figure 5. Embedded and stand-alone LDAP directory server with corporate (external) directory server

IBM Initiate LDAP directory server communicates with other IBM Initiate® Master Data Service® directory servers only if replication is enabled. Clustering is available only among IBM Initiate provided directory servers.

From version 8.1 forward, the move to a bundled LDAP server of the IBM Initiate Master Data Service dictates that all permissions are now group-based rather than user and group-based. Enhancements to support a group-based model have been made to the mpi_grphead and mpi_usrhead tables. Additionally, mpi_usrprop and mpi_usrxgrp tables were removed in an earlier release.

Configuration flow for the Master Data Engine LDAP directory server

Before beginning the installation of the Master Data Engine, review the task flow for configuring your LDAP directory server.

1. Determine your LDAP implementation requirements. For example, if you already have an external corporate directory server, whether you are going to use it with an embedded or stand-alone IBM Initiate Master Data Service provided LDAP directory server. If you are not using an external server, you must enable at least one embedded or stand-alone IBM Initiate LDAP directory server.
2. Install the Master Data Engine.
3. Create an engine instance by using the madconfig utility. Answer the prompts for the type of IBM Initiate Master Data Service provided LDAP directory server you are planning to use. You are prompted to indicate whether you want to use an embedded directory server, the directory server host (if you are not using an embedded server), the directory server port number and server admin port number, and whether the directory server is in a cluster with other LDAP directory servers.
4. If your implementation is using a corporate LDAP directory server, you must edit the `com.initiate.server.ldap.cfg` file. Information about enabling the Master Data Engine to communicate with the corporate server is provided in the configuring an external corporate LDAP Directory Server task.
5. Install the IBM Initiate Workbench application and include the User and Group Management plug-in.
6. Create your connections and add or modify your users and groups in IBM Initiate Workbench. The IBM Initiate Master Data Service provided LDAP directory server ships with a set of pre-configured groups, although you can add additional groups as necessary. These groups, which are provided primarily for compatibility with earlier versions, are discussed in the *IBM Initiate Workbench User's Guide*.

Attention: If you remove a Master Data Engine instance, the associated embedded LDAP directory server instance is also removed.

Related reference

"LDAP directory server worksheet" on page 14

"Master Data Engine instance worksheet" on page 6

"com.initiate.server.ldap.cfg file" on page 203

Related task

"Configuring an external corporate LDAP Directory Server" on page 207

Upgrade considerations for Master Data Engine LDAP directory server

When you upgrade your Master Data Engine, the madconfig utility upgrade_instance target generates all the necessary SQL and LDIF (LDAP Data Interchange Format) files needed to complete the upgrade.

All users, groups, and associated permissions defined in earlier versions of the Master Data Engine data model are promoted to the new IBM Initiate® Master Data Service® LDAP directory server during the upgrade process. Users who are not already members of any group are granted the default group for the interaction session. However, such users are visible to administrators within the LDAP tools; membership to this default group is maintained internally.

During an upgrade from 8.5 to 8.7, the madconfig utility prompts you to enter the direct path to the LDAP folder located under the instance being upgraded.

During upgrades, the LDAP data migration occurs only for instances that use embedded LDAP servers. No LDAP data migration occurs if stand-alone LDAP servers are implemented. Any superseding Master Data Engine instance being used as the upgrader must follow the same LDAP setup scheme as the instance being upgraded. In other words, upgrade from embedded LDAP servers to embedded LDAP servers, rather than from embedded LDAP servers to stand-alone LDAP servers.

com.initiate.server.ldap.cfg file

During creation of a Master Data Engine instance, an `com.initiate.server.ldap.cfg` file is stored in the *instanceName*`\inst\mpinet_`*instanceName*`\conf` directory. This file contains all the properties necessary to enable embedded, stand-alone, and corporate LDAP configuration scenarios.

When using the madconfig utility to create your engine instance, the answers you give provide the property settings for the embedded and stand-alone (internal) LDAP directory server. Settings for external corporate LDAP directory servers are manually added to the configuration file.

Validation of group authorization flows from embedded LDAP server to stand-alone LDAP server to corporate LDAP server. In other words, the mechanism checks first to determine whether the embedded LDAP server is enabled. If not, it checks to determine whether the stand-alone LDAP server is enabled. If not, it checks to determine whether the corporate LDAP server is enabled.

The `com.initiate.server.ldap.cfg` properties are described in this topic. Properties listed as multi-value support multiple entries for the property to be used. For example, by using the property `external.ldap.user.search.basedn.1`, you can define multiple user search base dns like so:

```
external.ldap.user.search.basedn.1=ou=Users1,dc=example,dc=com
external.ldap.user.search.basedn.2=ou=Users2,dc=example,dc=com
external.ldap.user.search.basedn.n=ou-=UsersN,dc=example,dc=com (where n is
an arbitrary number)
```

Embedded LDAP server properties

`embedded.ldap.enabled=true`
Indicates if the embedded LDAP server is enabled or disabled. If this property is set to “true,” do not change it for this instance. Otherwise, the authentication process can fail.

Responding "y" to this prompt during the create_instance process marks this property as true:

```
## Will this Initiate Master Data Engine instance use an embedded IBM Initiate
LDAP Server?
# Valid Options:  (y, n)
```

Remember, you must have either embedded or internal marked as true.

Stand-alone (internal) LDAP server properties

`internal.ldap.enabled=true`
Indicates if the stand-alone LDAP server is enabled or disabled. Responding "n" to this prompt during the create_instance process marks this property as true:

```
## Will this Initiate Master Data Engine instance use an embedded IBM Initiate
LDAP Server?
# Valid Options:  (y, n)
```

`internal.ldap.host.1=localhost`
Specifies the hostname of the stand-alone LDAP server in which to connect. This property is a multi-value option to allow for clustering.

`internal.ldap.port.1=1389`
Specifies the port number of the stand-alone LDAP server to connect to. This property is a multi-value option to allow for clustering.

`internal.ldap.ssl.enabled=true`
Specifies that socket connections to this stand-alone LDAP server use SSL. The stand-alone LDAP server must be configured to listen for SSL connections.

`internal.ldap.referral.type=ignore`
Specifies how referrals (or aliases) are handled. Valid options are: follow, ignore, or throw.

`internal.ldap.security.authentication.type=simple`
Specifies the type of authentication to use when connecting to this stand-alone LDAP server. Valid options are: none, simple, CRAM-MD5, or DIGEST-MD5.

`internal.ldap.security.binddn=cn=binduser,ou=Users,dc=exampledomain,dc=com`
Specifies the DN of the user to use for searching, reading, and comparing LDAP entries on this server. By default this property is cn=Bind,ou=System,ou=Users,dc=initiatesystems,dc=com.

`internal.ldap.security.bindpassword=bindpassword`
This property is used if the bindpassword is stored in plaintext. The default password for the bind user is ‘initiate’.

`internal.ldap.security.bindpassword2=856F383EB11CF91507442F342FFDE9F3`
This property is used if the bindpassword is requested to be encrypted.

Corporate (external) Master Data Engine LDAP server properties

Important: Corporate (external) LDAP DN settings must be added manually to the property file before configuring LDAP connections in IBM Initiate Workbench.

`external.ldap.enabled=true`
Indicates whether a corporate LDAP server is enabled or disabled.

`external.ldap.host.1=examplehost`
Specifies the hostname of the corporate LDAP server to connect to. This property is a multi-value option.

`external.ldap.port.1=389`
Specifies the port number of the corporate LDAP to connect to. This property is a multi-value option.

external.ldap.ssl.enabled=true
Specifies that socket connections to this corporate LDAP server use SSL. The corporate LDAP server must be configured to listen for SSL connections.

external.ldap.referral.type=follow
Specifies how referrals (or aliases) are handled. Valid options are: follow, ignore, and throw.

external.ldap.security.authentication.type=simple
Specifies the type of authentication to use when connecting to this corporate LDAP server. Valid options are: none, simple, CRAM-MD5, DIGEST-MD5.

external.ldap.security.binddn=cn=binduser,ou=Users,dc=exampledomain,dc=com
Specifies the bind user or service account to use for searching, reading, and comparing LDAP entries on this server.

external.ldap.security.bindpassword=bindpassword
This property is used if the bindpassword is stored in plaintext.

external.ldap.security.bindpassword2=madpwd2_encrypted_bindpassword
This property is used if the bindpassword is requested to be encrypted.

external.ldap.user.default.groupdn=cn=Default,ou=Group,dc=example,dc=com
Specifies the default group to assign users who are not members of any other group. This option is not required.

external.ldap.user.search.basedn.1=ou=Users,dc=example,dc=com
Specifies the search basedn to use for locating users. This property is a multi-value option.

external.ldap.user.search.filter.pattern=(
&(objectclass=person)(sAMAccountName={0}))
Specifies the filter pattern to use when searching for users. The parameter {0} designates the user name of the authenticating user that is substituted. The value varies depending on schema and back-end configuration.

external.ldap.user.search.filter.scope=subtree
Specifies the depth of the tree to do a user search. Valid options are subtree, object, and onelevel.

external.ldap.user.attribute.name=cn
Specifies the attribute that contains the full name of the user.

external.ldap.user.attribute.firstname=givenName
Specifies the attribute that contains the first name of the user.

external.ldap.user.attribute.lastname=sn
Specifies the attribute that contains the last name of the user.

external.ldap.user.attribute.email=mail
Specifies the attribute that contains the user email address.

external.ldap.user.attribute.userid=sAMAccountName
Specifies the attribute that contains the user ID.

`external.ldap.group.search.basedn.1=ou=Groups,dc=example,dc=com`
Specifies the search basedn to use for locating groups. This property is a multi-value option.

`external.ldap.group.search.filter.pattern.all=(objectclass=group)`
Specifies the filter to use when searching for all groups that are not member-specific. Common values include GroupOfNames, group, and groupOfURLs, but any number of values are possible.

`external.ldap.group.search.filter.pattern.member=(&(objectclass=group)(member={0}))`
Specifies the filter to use when searching for groups that a user is explicitly a member of. The parameter `{0}` designates the full DN of the authenticated user.

`external.ldap.group.search.filter.scope=subtree`
Specifies the depth of the tree to do a group search. Valid options are subtree, object, and onelevel.

`external.ldap.group.attribute.name=cn`
Specifies the attribute that contains the group name.

`external.ldap.group.attribute.description=displayName`
Specifies the attribute that contains the group description.

`external.ldap.group.attribute.member=member`
Specifies the attribute that contains the group member information.

Related task

"Configuring an external corporate LDAP Directory Server" on page 207

Changing the port setting for a stand-alone (internal) Master Data Engine LDAP Directory Server

If you need to change the port designation of a stand-alone (internal) LDAP server, editing the `config.ldif` configuration file can save you the trouble of reinstalling a new LDAP server.

Procedure

1. On the server running the Master Data Engine, go to the configuration directory for your stand-alone LDAP servers: *MAD_HOMEDIR*`\inst\mpildap_`*name*`\ldap\config\`

   where *MAD_HOMEDIR* is the full path to the directory created for the associated runtime instances (for example, `prod` or `qa`), and *name* is the name of the LDAP instance.
2. Open `config.ldif` file for editing. The file is divided into sections by DN.
3. Search within the `config.ldif` file for the `ds-cfg-listen-port` entry. This entry appears several times within the file. Specifically, check within these DNs:
   - `cn=LDAP Connection Handler,cn=Connection Handlers,cn=config`
   - `cn=LDAPS Connection Handler,cn=Connection Handlers,cn=config`
4. Edit the instance of the `ds-cfg-listen-port` parameter that appears within the DN that is enabled. (The enabled DN is the one with the `ds-cfg-enabled` parameter set to true.)

5. After you make the change, save and close the `config.ldif` file.
6. On the same server, go to: *MAD_HOMEDIR*`\inst\mpinet_`*name*`\conf\`
7. Open the `com.initiate.server.ldap.cfg` file for editing.
8. If the `internal.ldap.enabled` property is set to true, edit the value for the `internal.ldap.port.#` parameter (or parameters), where # indicates a number.

   Important: You do not need to edit the `com.initiate.server.ldap.cfg` configuration file unless the engine instance references the stand-alone LDAP server. A setting of `internal.ldap.enabled= false` indicates that the engine does not reference any stand-alone LDAP servers.
9. Save and close the `com.initiate.server.ldap.cfg` file.
10. Restart the LDAP server by using the madconfig utility stop_ldap and start_ldap commands.
11. If you make changes to the `com.initiate.server.ldap.cfg` file, restart the Master Data Engine by using the madconfig utility stop_instance and start_instance commands.

Configuring an external corporate LDAP Directory Server

If you use an external corporate LDAP directory server, there are a few steps required to enable communication between the Master Data Engine and your corporate directory server.

Before you begin

Before beginning the configuration process, groups must be defined in the corporate directory server and all users must be assigned to groups. Group and user definition must be done before implementation because the Master Data Engine associates internal permissions with an LDAP group. Because the IBM Initiate® Master Data Service® LDAP services provide default groups and users, implementing a corporate LDAP server requires that you have either an embedded or stand-alone (internal) IBM Initiate LDAP configured.

Procedure

1. Go to the engine instance *MAD_HOMEDIR*`\inst\mpinet_`*name*`\conf\` directory.
2. Open the `com.initiate.server.ldap.cfg` file.
3. Make sure that the `external.ldap.enabled` property is set to “true.”
4. These substeps define specific connection properties.
   a. In order for the Master Data Engine to create a connection to the corporate LDAP directory server instance, you must edit the `external.ldap.host` and `external.ldap.port` properties. Both of these properties allow multiple entries to be specified by appending a dot (.) followed by a number for each additional entry. If you want the engine to attempt connection to two servers on ports 389 and 1389 you would set the properties like this example:

```
external.ldap.host.1=examplehost1.com
 external.ldap.host.2=examplehost2.com
external.ldap.port.1=389
 external.ldap.port.2=1389
```

Attention: The Master Data Engine does not provide replication or load-balancing of corporate LDAP servers. It is assumed that corporate system administrators configure and enable replication and load-balancing according to their specific needs. Allowing multiple entries for corporate LDAP hosts and ports is intended primarily to enable fail-over.

b. If the directory server supports SSL encryption, set the `external.ldap.ssl.enabled` property to "true." You are also required to set up the necessary system properties in the `com.initiate.server.system.cfg` file to enable location of your certificate trust and keystores. Information about using SSL for communications with a corporate server is provided in the "Configuring SSL communications with corporate LDAP directory server" task.

c. Some directory instances might use referrals or aliases that reference entries in the LDAP directory server, but live in a different part of the tree or in a different LDAP server. To instruct the Master Data Engine on how to handle references, set the `external.ldap.referral.type` property to `follow`, `ignore`, or `throw`.

d. If the directory server uses more advanced authentication mechanisms, they can be specified through the `external.ldap.security.authentication.type` property. Typical values are "none," "simple," "CRAM-MFD5", or "DIGEST-MD5."

e. The Master Data Engine requires certain credentials for a user to make the initial connection to the directory server to execute reads, searches, and compares on users and groups. This user is called a bind user and has basic directory rights that anyone can use. A bind user typically has basic directory rights such as compare, search, and read. These properties are specific to this user: `external.ldap.security.binddn`
`external.ldap.security.bindpassword`
`external.ldap.security.bindpassword2`

   The binddn is the fully qualified DN of the bind user. If storing the password in encrypted format is not required, specify the password in the bindpassword property. If encryption is required, use the madpwd2 utility to encrypt the password and place the result in the bindpassword2 property.

5. Next you must add information specific to user search base DNs, filters, and attribute names. Defining parameters to search only the relevant portions of the LDAP hierarchy, rather than the entire hierarchy, helps to improve performance.

   a. To specify a default group for users who are not a member of any registered group, set the property `external.ldap.user.default.groupdn` to the fully qualified DN of the default group.

   b. Search DNs must be specified to enable the Master Data Engine to locate users during authentication through the `external.ldap.user.search.basedn` property. This property allows for multiple entries by appending a dot and a number for each additional entry. For example:

```
external.ldap.user.search.basedn.1=ou=User1,dc=exampledomain,dc=com
 external.ldap.user.search.basedn.2=ou=User2,dc=exampledomain,dc=com
 external.ldap.user.search.basedn.3=ou=User3,dc=exampledomain,dc=com
```

   c. Use the `external.ldap.user.search.filter.pattern` property to tell the Master Data Engine how to match a specific user during authentication. This property can be set to any LDAP-compliant filter. The only restriction is that at least one attribute must be used to match the login name. For example, you have a DN for user John Doe set to `cn=John`

Doe,cn=Users,dc=labsvcs1,dc=exampledomain,dc=com.. You can either use the common name (cn), sAMAccountName, or any other attribute that can uniquely identify the user John Doe.

d. To tell the Master Data Engine how far down the tree to search for a user, edit the external.ldap.user.search.filter.scope property. Typical values are:
   - subtree- runs the search starting at the search base DN and recursively drills down to every leaf in the tree. This method is the most often used.
   - onelevel - runs the search starting at the search base DN and searches only those entries one level beneath the search base.
   - object - searches for exact DN and filter.

e. For the Master Data Engine to understand how to map user attributes in the directory server to domain objects used internally, set these properties to the corresponding attributes in your directory server.

```
external.ldap.user.attribute.name
 external.ldap.user.attribute.firstname
 external.ldap.user.attribute.lastname
 external.ldap.user.attriubte.email
 external.ldap.user.attribute.userid
```

6. The final group of properties enable the Master Data Engine to locate group search base DNs, filters, and attribute name information. Defining parameters to search only the relevant portions of the LDAP hierarchy, rather than the entire hierarchy, helps to improve performance.

   a. Specify the group search base DN by using the external.ldap.group.search.basedn property. This property allows multiple entries by appending a dot and number for each additional entry.

   b. Use the external.ldap.group.search.filter.pattern.member property to tell the engine how to match a specific group for a user. This property can be set to any LDAP-compliant filter and the only restriction is that at least one attribute must be used to match the DN of the user. For example, if the DN for user John Doe is cn=John Doe,cn=Users,dc=exampledomain,dc=com and this user is a member of the group with DN cn=Builtin,dc=exampledomain,dc=com, then the group must contain an attribute that has John Doe as an assigned member.

   c. To filter the list of groups found during a group synchronization, set a filter in the external.ldap.group.search.filter.pattern.all property. This property can be any LDAP-compliant filter, however no substitutions are completed on this filter.

   d. Next, tell the Master Data Engine how far down the tree to search for a group by setting the external.ldap.group.search.filter.scope property. Typical values are:
      - subtree - runs the search starting at the search base DN and recursively drills down to every leaf in the tree. This method is most often used.
      - onelevel - runs the search starting at the search base DN and searches only those entries one level beneath the search base.
      - object - searches for the exact DN and filter.

   e. To tell the Master Data Engine how to map group attributes in the directory server to domain objects used internally, set these properties to the corresponding attributes in your directory server:

```
external.ldap.group.attribute.name
 external.ldap.group.attribute.description
 external.ldap.group.attribute.member
```

Results

After these steps have been completed, restart the Master Data Engine server so that it reads the new values in the `com.initiate.server.ldap.cfg` file. Then use IBM Initiate Workbench to create and test your connections.

Related task

"Configuring SSL communications with a corporate LDAP directory server"

Configuring SSL communications with a corporate LDAP directory server

Enabling SSL for a corporate LDAP directory server does not require that you enable SSL for communications with the Master Data Engine.

About this task

These instructions describe the configuration of one-way SSL, between your corporate LDAP server and the Master Data Engine. Enabling SSL communications for LDAP directory servers that support encryption requires that you configure values within the `com.initiate.server.ldap.cfg` and `com.initiate.server.system.cfg` files. You can use these instructions to configure SSL for a corporate LDAP server regardless of whether SSL is configured for the Master Data Engine.

If you enable SSL for the Master Data Engine after you have enabled SSL for your corporate LDAP directory server, you do not need to repeat the process of enabling SSL for the corporate LDAP directory server. Instead add the corporate LDAP certificate to the truststore used by the Master Data Engine.

SSL communication also requires that you specify a truststore file. The Master Data Engine provides a default truststore (`ibmcorporationtrust.jks`), but you can choose to generate your own using the Java keytool utility. When generating a new truststore, the certificate can be self-signed, though it is suggested that you use a certificate from a trusted certificate authority (CA). The certificate is used for the handshake between your corporate LDAP server and the Master Data Engine.

Procedure

1. If you have not already done so, set the `external.ldap.ssl.enabled` property to "true" within the `com.initiate.server.ldap.cfg` file.
2. Also within the `com.initiate.server.ldap.cfg` file, change the value of `external.ldap.port.1` from "389" to the port that your environment uses for encrypted LDAP. The default port for LDAP is 389; encrypted LDAP typically uses port 636.
3. If you do not want to use the default `ibmcorporationtrust.jks` as the truststore, create a truststore file with keytool. Keytool is the native Java key and certificate management utility. Use the `-import` command to add your corporate LDAP certificate to the truststore. For complete information about the keytool utility, see the documentation at http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html.

After you have generated the truststore file, copy it to the *MAD_HOMEDIR*\inst\mpinet_*name*\conf directory, where *MAD_HOMEDIR* is the full path to the directory created for the associated runtime instances (for example, prod or qa, and *name* is the engine instance name.

4. Edit the com.initiate.server.system.cfg file within the *MAD_HOMEDIR*\inst\mpinet_*name*\conf directory:

   ```
   javax.net.ssl.trustStore
    javax.net.ssl.trustStorePassword
    javax.net.ssl.trustStoreType
   ```

   If you are using the default truststore, ibmcorporationtrust.jks, set the values to:

   ```
   javax.net.ssl.trustStore=${mad.root.dir}/conf/ibmcorporationtrust.jks
    javax.net.ssl.trustStorePassword=rmi+ssl
    javax.net.ssl.trustStoreType=JKS
   ```

   If you are using a new truststore, set the properties to the values you provided to the keytool utility.

   In addition to the trustStore properties, the com.initiate.server.system.cfg file contains keyStore properties. Those properties are not required for this configuration.

5. Restart the Master Data Engine server so that it reads the new values in the com.initiate.server.ldap.cfg and com.initiate.server.system.cfg files.

High-availability and replication configuration for the Master Data Engine LDAP directory server

The IBM Initiate® Master Data Service® LDAP directory uses a multi-master replication scheme for synchronizing data between multiple IBM Initiate® Master Data Service® provided LDAP directory instances.

A multi-master directory server topology enables all servers within the cluster to do both read and write operations by publishing their changes to a central replication service. Each directory server in this topology can act as a replication service which provides high availability to directory resources without the possibility of a single point of failure. Each instance of a Master Data Engine can be configured to use one or more LDAP directory instances for authentication purposes. In situations where the software must be available 24 by 7, replication provides a way to meet this use case.

Important: High-availability and replication is available only among IBM Initiate provided LDAP directory servers. It is not available if your configuration includes a corporate LDAP server.

To allow for failover, the LDAP configuration files on each LDAP server must list the connection information for all other LDAP servers in the cluster.

The replication technology used in the embedded and stand-alone LDAP directories is specific to the vendor used (OpenDS) and cannot be used with other directory server implementations (for example, Active Directory). For more architectural information about the directory server replication abilities, visit https://www.opends.org/wiki/page/ArchitecturalOverview.

Excessive writes to a directory with replication enabled can influence performance, and the effect on performance becomes more apparent as the number of nodes in the cluster increases.

Enabling replication for the Master Data Engine LDAP directory server

Replication is enabled for a directory instance through the madconfig utility during Master Data Engine or LDAP instance creation.

Before you begin

Make sure that you have a Master Data Engine that uses an existing embedded or stand-alone IBM Initiate LDAP directory instance running and listening on port 1389. Make sure that when you created this engine instance that you responded no to the prompt for participation in a cluster with other IBM Initiate LDAP servers.

You can answer yes to this question only after you have at least one embedded or stand-alone IBM Initiate LDAP instance installed.

Procedure

1. Create an LDAP directory instance by using the madconfig utility create_instance or create_ldap target.
2. Type a name for the instance and provide the appropriate database information.
3. Type the server host name or IP address.
4. Type a value for the listening port of this LDAP instance. This port number must be different from any other if you are added directories on the same server.
5. Type y to identify that the LDAP server is participating in a cluster.
6. Press Enter to accept the default replication port number.
7. Type the cluster peer host name.
8. Type the name or IP address of the server that has the peer located on it.
9. Enter a cluster peer port number. The default is 1389. Unless your peer LDAP server is listening on another port, press Enter to accept the default.
10. Enter the peer replication port number. The default is 8989.

Results

After answering all prompts, the output can be similar to this example:

```
Executing c:\ldap2\inst\mpildap_ldap2\ldap\bin\dsreplication.bat

Establishing connections ..... Done.
Checking Registration information ..... Done.
Configuring Replication port on server
examplehost620.Initiatesystems.com:1389 .....Done.
Configuring Replication port on server
examplehostd620.Initiatesystems.com:2389 .....Done.
Updating replication configuration for baseDN dc=initiatesystems,dc=com on
server examplehostd620.Initiatesystems.com:1389 .....Done.
Updating replication configuration for baseDN dc=initiatesystems,dc=com on
server examplehostd620.Initiatesystems.com:2389 .....Done.
Updating Registration configuration on server
examplehostd620.Initiatesystems.com:1389 .....Done.
Updating Registration configuration on server
examplehostd620.Initiatesystems.com:2389 .....Done.
Updating replication configuration for baseDN cn=schema on server
examplehostd620.Initiatesystems.com:1389 .....Done.
Updating replication configuration for baseDN cn=schema on server
examplehostd620.Initiatesystems.com:2389 .....Done.
Initializing Registration information on server
```

```
examplehostd620.Initiatesystems.com:2389 with the contents of server
examplehostd620.Initiatesystems.com:1389 .....Done.
Initializing schema on server examplehostd620.Initiatesystems.com:2389 with
the contents of server examplehostd620.Initiatesystems.com:1389 .....Done.
Return Code: 0, Time elapsed: 18.201 sec


Executing c:\ldap2\inst\mpildap_ldap2\ldap\bin\dsreplication.bat

Initializing base DN dc=initiatesystems,dc=com with the contents from
examplehostd620.Initiatesystems.com:1389:
12 entries processed (44 % complete).
27 entries processed (100 % complete).
Base DN initialized successfully.
```

For each additional node added, you must give the coordinates to only one other node in the cluster. The nodes can self-discover all other nodes through the replication service. All data replicated over the replication ports is encrypted. The only data that is not encrypted, unless SSL is enabled, is the information sent over your registered LDAP port number.

You can test your configuration by using the madconfig utility start_ldap target.

Related reference

"madconfig utility" on page 102

Appendix B. Sample com.initiate.server.system.cfg file

The com.initiate.server.system.cfg file contains most of the configuration parameters for the instance including global settings for logging, the language locale setting (C layer), and the detail settings for SSL communication.

On an engine host, the com.initiate.server.system.cfg properties file is located in the instance conf directory. For example:

Microsoft Windows: *MAD_HOMEDIR*\inst*type_name*\conf

IBM AIX, Linux, or Solaris: /*MAD_HOMEDIR*/inst/*type_name*/conf

where *MAD_HOMEDIR* is the full path to the instance directory, *type* is one of the these options, whichever applicable; and *name* is name of the runtime instance:

- mpinet for an engine instance.
- mpientmgr for an entity manager.
- mpildap for an IBM Initiate LDAP directory server.

The back slash (\) is the escape character, so you must use a double back slash (\\) to introduce one as demonstrated in the mad.root.dir definition.

The example is for an engine instance with the name prod100_1.

```
MAD_CONNSTR=DSN=prod100_1;UID=prod100_dbuser;PWD=prod100_dbpassword
MAD_DBTYPE=mssqlu
MAD_DBXTEST=1
MAD_CTXLIB=ODBC
MAD_SRVNO=54422837
MAD_SMTLIST=en_US
mad.root.dir=C:\\Program Files\\IBM\\Initiate\\Engine10.0.0
mad.home.dir=C:\\IBM\\initiate\\home\\prod100_1
mad.inst.dir=C:\\IBM\\initiate\\home\\prod100_1\\inst\\mpine
t_prod100_1
mad.inst.name=prod100_1
mad.log.name=mpinet_prod100_1-%s.mlg
mad.perflog.name=mpinet_prod100_1-%s.plg
mad.triggerlog.name=mpinet_prod100_1-%s.tlg
mad.ant.interactive=false
mad.jmx.objectname=com.customer:service=MPINETPROD100_1
cloveretl.properties=cloveretl.properties
felix.fileinstall.dir=C:\\Program Files\\IBM\\Initiate\\Engine10.0.0\\conf
felix.fileinstall.filter=org\\.apache\\.felix\\.fileinstall-.*\\.cfg
felix.fileinstall.tmpdir=work
javax.net.ssl.keyStore=C:\\Program
Files\\IBM\\Initiate\\Engine10.0.0\\conf\\ibmcorporation.p12
javax.net.ssl.trustStore=C:\\Program
Files\\IBM\\Initiate\\Engine10.0.0\\conf\\ibmcorporationtrust.jks
javax.net.ssl.keyStorePassword=rmi+ssl
javax.net.ssl.trustStorePassword=rmi+ssl
javax.net.ssl.keyStoreType=PKCS12
javax.net.ssl.trustStoreType=JKS
MAD_CALLBACKLIB=
```

For more information about Java-based property files, see http://java.sun.com.

Appendix C. Master Data Engine storage files (stofiles)

Storage files (or stofiles), are text files used to specify RDBMS-specific syntax related to table and index creation. This information can include sizing, locking mode, and physical location.

The format of a stofile is: `S|stoname|stoclause|optional comment|`

`stoname` is the name of the object to be modified. This object is a table name or index name.

`stoclause` is the RDBMS-specific text that is appended to the create table or create index statement.

`comment` is any text or note you want to include. This comment is optional. If left empty, however, you must include the trailing pipe character.

Any row that begins with a number sign (#) is treated as a comment line and is ignored. Blank lines are acceptable.

See your RDBMS-specific documentation on what is allowed in the stoclause.

This example shows an Oracle storage file:

```
S|mpi_syskey|TABLESPACE MYDATA01 storage (initial 10M next 5M)||
S|mpi_syskey1|TABLESPACE MYINDEX1 storage (initial 10M next 5M PCTINCREASE 0)|No
increase on the index|
```

Appendix D. Thread count settings

Service thread count settings are tuned during the implementation process to allow for determination of the best use of available resources.

As a starting point, these settings are suggested:

- One context pool object (service thread) for each inbound and outbound Message Broker Suite interface that is going to have consistent traffic.
- For data stewardship (for example, task and entity management), one context pool object for every 10 users in a low volume setting, or a one-to-five ratio for higher volume.
- If there are other pool-based resources connecting to the Master Data Engine each one has a connection pool of their own that rides over the top of the Master Data Engine. A starting point of one service thread for every two contexts pooled is suggested.

After setting the starting point, measure response time and adjust the thread count settings up or down as required.

An example a configuration consisting of two IBM Initiate Inspector users (high volume), 50 IBM Initiate® Enterprise Viewer users with a pool of six contexts, two inbound broker interfaces, and one outbound broker yields a total of seven service threads based on this definition:

- Three service threads for inbound and outbound brokers.
- Three service threads for IBM Initiate® Enterprise Viewer.
- One service thread for IBM Initiate Inspector.

Appendix E. Data source prompt examples

You can use the madconfig utility to create data sources. Each database platform yields different prompts.

These examples show the specific prompts per database platform.

IBM DB2 data source prompts

prompt: (Type the database data source name:)
response example: prod

prompt: (Type the database type:)
response example: db2

prompt: (Type the DB2 database host:)
response example: dbprod.customer.com

prompt: (Type the DB2 database port:)
response example: 50000

prompt: (Type the DB2 database name:)
response example: prod

MSSQL data source prompts

prompt: (Type the database data source name:)
response example: prod

prompt: (Type the database type:)
response example: mssqlu

prompt: (Type the SQL Server server name:)
response example: dbprod.customer.com

prompt: (Type the SQL Server database name (this value is case sensitive):)
response example: prod

Oracle Net data source prompts

prompt: (Type the database data source name:)
response example: prod

prompt: (Type the database type:)
response example: oracle

prompt: (Type the Oracle database host (for wire protocol only, otherwise leave blank:)
response example:

prompt: (Type the Oracle server name)
response example: prod

Oracle Wire data source prompts

prompt: (Type the database data source name:)
response example: prod

prompt: (Type the database type:)
response example: oracle

prompt: (Type the Oracle database host (for wire protocol only,

otherwise leave blank:)
response example: dbprod.customer.com

prompt: (Type the Oracle database port:)
response example: 1521

prompt: (Type the Oracle database SID (leave blank to specify RAC service name instead of SID:)
response example: ora10264

Appendix F. Uninstall the Master Data Engine environment

Uninstalling a Master Data Engine environment includes removing all engine runtime instances and datasources, and running the uninstaller.

Before beginning the uninstall process:

- If you are planning to re-install this runtime environment using the same database instance that it uses, make sure that you create a backup image of the database as a precaution.

  If the runtime environment was not in production and the database has no member data, you do not need to back up the database.
- In the environment that you want to uninstall, stop each runtime engine instance and each stand-alone entity manger instance.

After completing the prerequisites, continue with these tasks in this order:

1. Remove all engine runtime instances.
2. Remove the engine data source.
3. Run the uninstaller.

Related tasks

"Stopping an engine instance from the Microsoft Windows Control Panel" on page 56

"Stopping an engine instance with the madconfig utility" on page 57

"Stopping an engine instance with its batch or script file" on page 57

"Stopping an entity manager instance from Microsoft Windows Control Panel" on page 60

"Stopping an entity manager instance with the madconfig utility" on page 60

"Stopping an entity manager instance with its batch or script file" on page 61

"Removing Master Data Engine runtime instances"

"Removing Master Data Engine data sources" on page 224

"Running the Master Data Engine uninstaller" on page 224

Removing Master Data Engine runtime instances

Before you uninstall a Master Data Engine, you must remove any associated engine instances. The madconfig utility is used to remove instances.

About this task

Repeat this procedure to remove each runtime instance associated with the environment you are planning to uninstall.

Attention: If you remove a Master Data Engine instance, the associated embedded LDAP directory server instance is also removed.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the madconfig utility command applicable to your operating system:

   Microsoft Windows: `madconfig remove_instance`

   IBM AIX, Linux, or Solaris: `madconfig.sh remove_instance`
3. For each prompt, review the information; type a value; and then press Enter.

Results

In the output, confirm that a `BUILD SUCCESSFUL` message displays. If the instance is successfully removed, continue with removing the datasource.

Related task

"Removing Master Data Engine data sources"

Removing Master Data Engine data sources

Use the madconfig utility to remove data sources.

About this task

Repeat this procedure for each data source configuration that is created for the runtime environment you plan to uninstall. Typically, there is only one data source per runtime environment.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the command applicable to your operating system:

   Microsoft Windows: `madconfig remove_datasource`

   IBM AIX, Linux, or Solaris: `madconfig.sh remove_datasource`
3. Type the name of the data source configuration you want to remove, and then press Enter.

Results

In the output, confirm that the `BUILD SUCCESSFUL` message displays.

Running the Master Data Engine uninstaller

After you have removed your Master Data Engine instance and data source, you can run the Master Data Engine uninstaller.

About this task

On Microsoft Windows, IBM AIX, Linux, or Solaris systems, you can run GUI- or CLI-based uninstall processes.

Procedure

1. Be sure that you have read and followed the prerequisites described in "Uninstall the Master Data Engine environment".
2. On the host with the runtime environment you want to uninstall, verify that you have removed your engine instance and data source.
3. Browse to the root of the corresponding installation directory. For example:

   Microsoft Windows

   ```
   cd C:\Program Files\IBM\Initiate\Engine10.0.x\uninstall
   ```

   IBM AIX, Linux, or Solaris

   ```
   cd /opt/IBM/Initiate/Engine10.0.x/uninstall
   ```
4. Start the uninstaller:
   - **Microsoft Windows**: Double-click `uninstaller.exe`
   - **XWindows**: From the command-line terminal, run `uninstaller`

     For complete X Windows information, see its documentation.
5. In the uninstall wizard, follow the instructions to complete the engine installation.

Related concept

Appendix F, “Uninstall the Master Data Engine environment,” on page 223

Related tasks

“Removing Master Data Engine runtime instances” on page 223

“Removing Master Data Engine data sources” on page 224

Appendix G. Performance planning for the Master Data Engine

Performance is a critical but complex subject. The performance topics are intended to give you an overview of performance considerations as they apply to IBM Initiate Master Data Service applications.

The topics include a general discussion of performance concepts, hardware and software considerations, and IBM Initiate Master Data Service workload profiles and performance implications of each profile. This information is offered in the context of optimizing performance for IBM Initiate Master Data Service applications, and is not meant to be a primer on performance in general.

When planning, monitoring, and tuning performance, bear in mind that there are many factors to consider, and many tradeoffs. These factors include not only performance itself, but also considerations like hardware costs, administrative costs, and maintainability.

Many factors can affect the effectiveness of hardware configuration or software settings. Consult with your project team or architect before making significant changes to your hardware or software configuration.

Performance evaluation and tuning considerations

As you plan your performance strategy, there are some key considerations to keep in mind.

- What works well in one installation or software and hardware combination stack might not work as well in a different one. Always take your specific applications and hardware configuration into account as you benchmark and tune your performance.
- Start with a standard, "plain vanilla" installation of software, such as operating system and database. Baseline your performance on the standard platform so that you have a baseline from which to evaluate the effects of changes to your setup.
- Be clear on what your performance goals are. Understand the difference between latency and throughput, and know which one is more important in your particular installation. Likewise, understand the different IBM Initiate® Master Data Service® workload profiles, and plan appropriately to meet your performance targets for each.
- As you modify your configuration, test at regular intervals to validate each change. Be sure that you have a clear strategy for backing out any changes that do not yield the wanted results.

Performance benchmarking

Users must not disclose the results of any benchmark test of the IBM Initiate® Master Data Service® or its subcomponents to any third party without prior written permission from IBM.

Users can disclose the results of any benchmark test of the IBM Initiate® Master Data Service® or its subcomponents to any third party if the user meets these requirements:

1. Publicly discloses the complete methodology used in the benchmark test (for example, hardware and software setup, installation procedure, and configuration files).
2. Contacts IBM to set up benchmark testing running the IBM Initiate® Master Data Service®. The IBM Initiate® Master Data Service® must be run in its specified operating environment by using the latest applicable updates, patches, and fixes available for the IBM Initiate® Master Data Service® from IBM or third parties that provide IBM products.
3. Follows all performance tuning guidance available by contacting IBM.

Performance key concepts

Before beginning your performance evaluation and tuning, it is helpful to understand some general performance concepts.

The most common performance concepts include:

- work
- latency
- throughput
- CPU
- memory
- storage
- networks

Work

The concept of work is the most important factor to consider when evaluating and optimizing performance.

In the simplest terms, servers perform work, and each subsystem in a server supports this role:

- CPUs perform the work, manipulating data.
- Memory caches work for faster access. It optimizes the flow of information from persistent storage to the CPUs, ensuring the CPUs can perform work efficiently.
- Storage persists the work product, so you can reference it at a later time.
- Networks allow servers that perform different work to communicate with each other.

The IBM Initiate® Master Data Service® applications ask servers to perform work. Server performance depends on how much work the server is asked to do and how appropriately sized that server is for the workload.

Related concepts

"CPU" on page 229

"Memory" on page 230

"Storage" on page 231

"Networks" on page 232

Latency

Latency is a measure of how long a transaction takes to complete.

This measure is speed on an atomic transaction level: how fast can a server complete a transaction? Latency is measured in time per transaction. Typical transaction latencies are measured in milliseconds (ms) per transaction.

An analogy might be, how long does it take someone to drive from point A to point B? Typically the trip can be made in 3 hours without traffic; the transaction latency is 3 hours for this trip. However, the average latency for this trip might be 3.5 hours per trip because there is typically traffic congestion that slows the driver down. This analogy illustrates how latency is a representation of an atomic event; the drive from point A to point B.

Latency can be critical for end-user driven workloads.

Throughput

Throughput is a rate. A rate is the measure of how many actions are completed in a unit of time.

This measure is most often characterized as TPS (transactions per second), TPM (transactions per minute), TPH (transactions per hour), or TPD (transactions per day). Throughput is not the inverse of latency. Your installation might have a high latency of 2 seconds, but a tremendous throughput rate of 2,000 TPS. This throughput might be acceptable in a batch environment where there is not a major concern with individual transaction durations but overall record processing rates.

As an analogy, consider the trip from point A to point B in the "Latency" example from the highway department perspective. How many cars travel from point A to point B in a day? The highway department concern is not with the latency of an individual car, but with the overall throughput of the highway, measured in cars per day. The department might find that by lowering the speed limit from 70 miles per hour to 50 miles per hour they get more cars from point A to point B in a day (for example, 10,000 versus 12,000). The increase in throughput is because there are fewer accidents that cause overall delays. From an atomic perspective, an individual car is going 29 percent slower, resulting in a longer trip (higher latency), but the overall throughput for the road has risen by 20 percent. Now the highway department might choose to build two more traffic lanes. The overall throughput then doubles from 12,000 to 24,000 cars per day, but the latency is still 3 hours per car. Individual drivers are not going any faster, but a lot more people are now getting from point A to point B in the same time period.

Throughput is typically critical for automated workloads, such as harness-driven inputs or bulk search operations.

CPU

The CPU is responsible for performing the work completed on a server.

While CPU capacity typically receives much attention when a server system is sized, an IBM Initiate Master Data Service application requires a great deal of processing power only during the bulk cross match (BXM) process.

Apart from the BXM, CPU processing power is not as significant a performance bottleneck as memory and storage subsystems.

Related concept

For information about BXM and run time operations, see “Master Data Engine workload profiles” on page 232.

Memory

Memory plays a key role in optimal performance.

When considering performance, memory and “Storage” on page 231 subsystems are linked. To truly understand server performance, you must think of a server not as a machine, but as a tool for getting work done. In order to complete work, you must continually feed the server processors with information, to ensure maximum efficiency. Since parts within the server operate at different speeds, a series of memory buffers are used to perform speed matching between these components. This illustration shows the presence of memory buffers throughout a server:

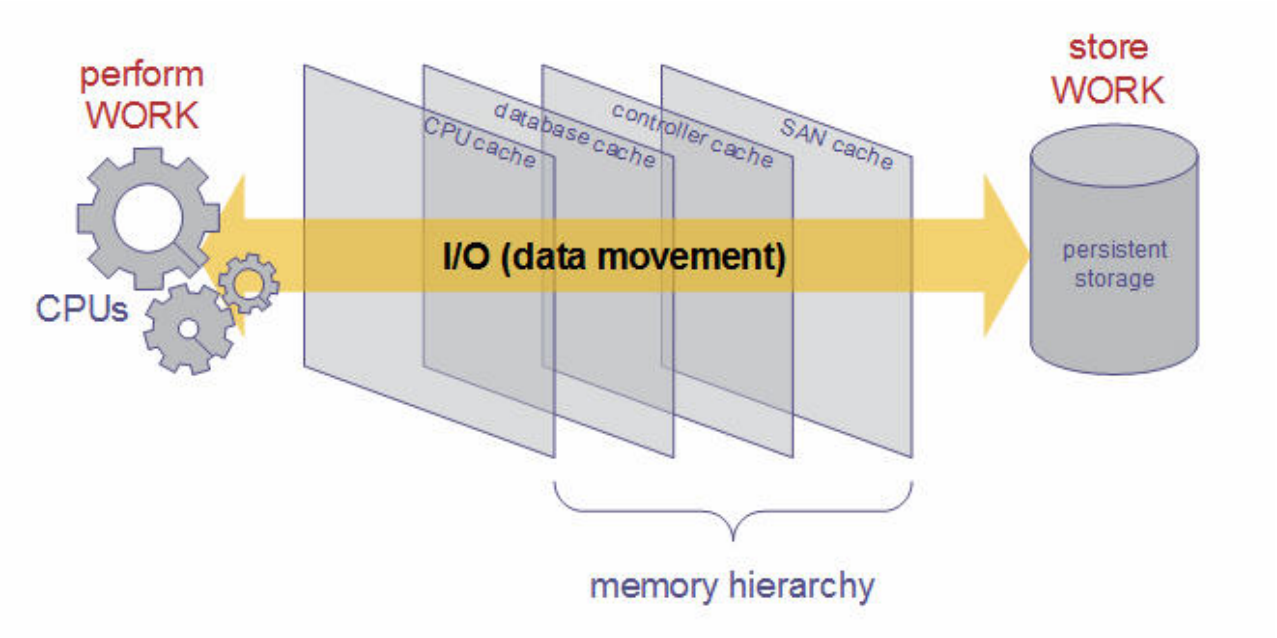

Figure 6. Memory buffer caches

These buffer caches minimize the differences in speed of components within a server. The buffer caches allow for the efficient flow of data back and forth between persistent storage where information is stored, and the CPU, where work is performed on the data. If these buffer caches are not sized appropriately, then a server might not operate efficiently while moving data, leading to a performance problem.

The most critical buffer that you have direct influence over is main memory. Main memory is typically the largest buffer cache in a server. Main memory provides a mechanism to satisfy read requests to persistent storage from “near memory” and reducing the number of trips required to “far memory”. (“Near memory” is faster than “far memory.”) In terms of database server performance, large read buffer caches residing in main memory dramatically boost the performance of a

read-intensive application. The boost is due to a reduction in the number of trips made to physical disk to retrieve information.

To avoid undersizing main memory buffer cache, ensure that you have enough available room to expand the buffers on both the application and the database servers.

Memory requirements for the Master Data Engine differ during bulk and run time operations.

Related concept

"Master Data Engine workload profiles" on page 232

Storage

The storage subsystem is a critical server component that is often overlooked. Input and output (I/O) bandwidth is a key consideration when evaluating storage needs.

I/O bandwidth is composed of spindles (the physical disks backing the system) and I/O channels (the number and width of the connections from the server to the storage array). I/O is critical for moving data from persistent storage up the memory hierarchy to the CPUs to get work done. If an I/O subsystem is undersized, performance suffers dramatically.

Here are some key terms for storage:

Table 72. Storage terminology

Acronym	Term	Description
DAS	Direct attached storage	This storage type is the most common, where the spindles are attached directly to the server. These systems can be fast and well-suited to high performance environments. However, as database sizes grow and spindle requirements increase, their efficiency drops. Most large scale, high-performance database environments are deployed on SANs rather than DAS.
SAN	Storage area network	The SAN is the most common high performance storage environment used today. It centralizes spindles into a storage array that is available to multiple servers through a private storage network. SAN connections are persistent and well-suited to high availability (HA) deployments. SAN fabric speeds are typically 2 - 4 Gbps (gigabits per second) and array sizes can grow to thousands of spindles.
NAS	Network-attached storage	NAS systems are commonly used for non-persistent storage connections such as file sharing and network backups. They are typically not used for database applications due to the non-persistent state of the connection.
iSCSI		iSCSI is a protocol that is used with less expensive NAS devices to make connections persistent, suitable for database applications. This protocol makes SAN type flexibility available at a lower price point. However, most enterprise level deployments leverage SAN technology as their foundation.

Networks

Networks allow you to communicate with servers.

From a performance perspective, most applications are concerned with the level of traffic that is generated by users or other servers over a network. Typically the Master Data Engine does not generate a significant load on this subsystem. The Master Data Engine typically has a low user count, and most of the data it sends "over the wire" requires a fraction of the bandwidth that is available. A typical busy installation uses about 30 percent of the bandwidth available on a gigabit network. Performance tuning efforts focus instead on the CPU, memory, and storage subsystems.

Master Data Engine workload profiles

The Master Data Engine has two distinct workload profiles: bulk processing and run time.

Bulk processing typically refers to the bulk cross match (BXM) process, which is most commonly performed during the initial stage of an implementation, and again right before the system goes live.

Run time processing refers to the day-to-day operations of the Master Data Engine, after the initial bulk phase has been completed.

Each workload profile has its own performance profile and suggestions.

Bulk processing

Bulk processing typically refers to the bulk cross match (BXM) process, which is most commonly performed during the initial stage of an implementation, and again right before the system goes live.

BXM is a Master Data Engine process that enables the comparison and linkage of thousands of records per second. This process loads the binary `.unl` extracts from any of the derivation utilities and then measures the comparison scores against your threshold settings to create entity assignments (linkages) and initial tasks.

The typical performance "footprint" of the bulk processing phase is:

- CPU-intensive
- Memory-intensive
- Storage demands focus on sequential input and output (I/O) operation

CPU considerations in bulk processing

Bulk processing (BXM) tends to be CPU intensive.

The mpxcomp utility is the most computationally intensive phase of the BXM process. Larger customers (100 million records plus) might leverage 8 -16 cores during the mpxcomp utility process to execute a thread per core to speed up the comparison phase by using parallel threads of execution.

Memory considerations in bulk processing

The Master Data Engine server memory requirements for the application server are highest during the bulk processing (BXM) phase.

Several key phases (specifically when running the mpxcomp and mpxlink utilities) require enough memory to house several key objects into memory. This leads to a requirement for 8 GB or more of available memory for typical customers, and 64+ GB for large-scale implementations.

Attention: Database memory requirements are discussed separately from application server memory requirements. Memory requirements for the application server are high during bulk processing and relatively low for run time operations. Database memory requirements are typically higher during run time operation than during bulk processing. However, the overall memory footprint for each profile might be similar.

Storage considerations in bulk processing

Storage access follows a typical bulk profile, with large sequential reads and writes following large blocks of computation time on the server running the bulk cross match (BXM).

The footprint of the BXM phase can be large, as storage is required for the source data, intermediate files, and the final `.unl` files. These footprints are determined by a host of variables specific to your data; estimates can be generated during implementation.

Database considerations in bulk processing

A database is not required for bulk cross match (BXM) processing, except for the final data load phase. Database performance is generally not a critical consideration for this phase.

For customers seeking to minimize the bulk processing window, database loads can be done in parallel to certain long running phases. This BXM timeline diagram illustrates the overlap of the membktd utility load with the mpxcomp utility phase for a sample 50 million member BXM. The shaded regions in the timeline could be compressed to save further time.

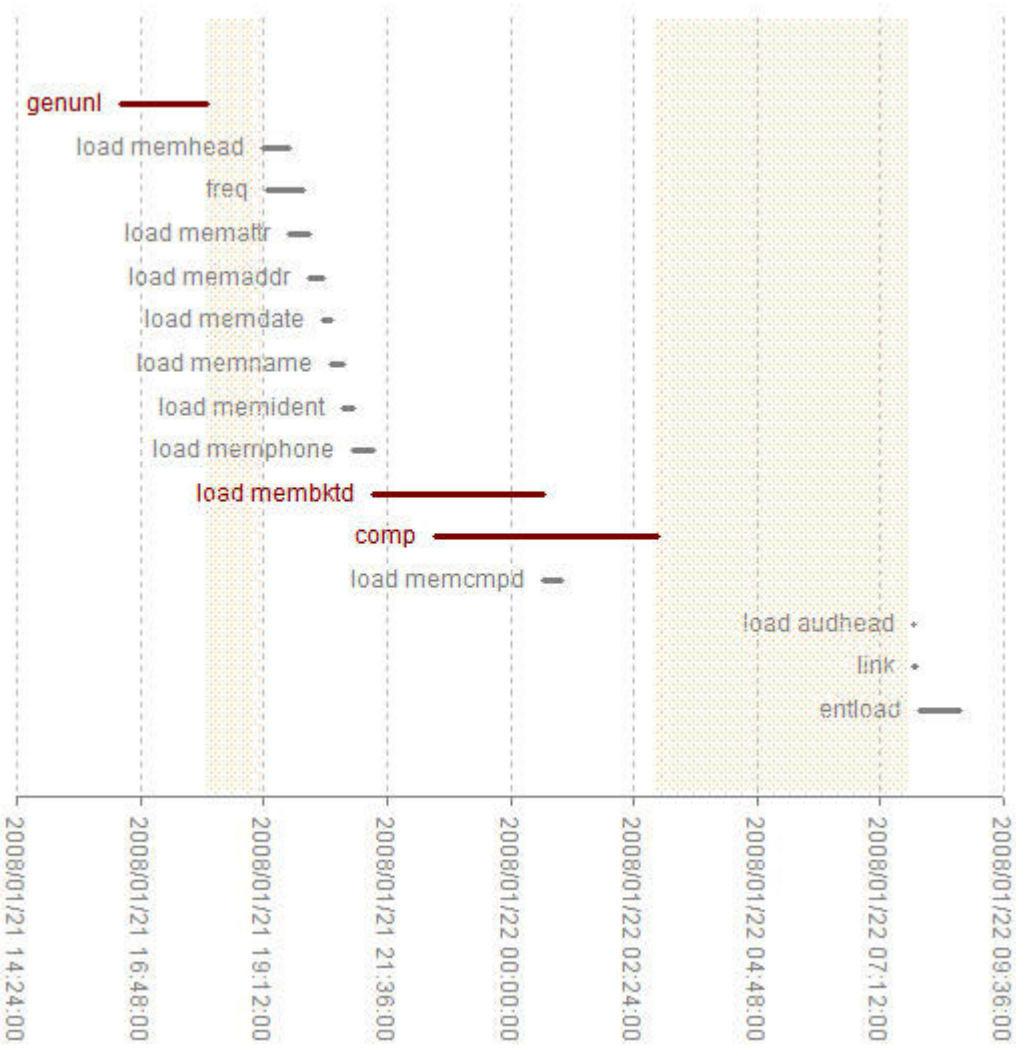

Figure 7. A sample BXM timeline

Run time processing

Run time processing refers to the day-to-day operations of theMaster Data Engine, once the initial bulk cross match (BXM) phase has been completed.

While cross matching is still performed as new sources are brought online, or as records from existing sources are added to the Master Data Engine database, it is done incrementally (by using incremental cross match [IXM]) as opposed to in bulk cross match (BXM).

The typical performance footprint of run time processing is:

- CPU loads are relatively light, but dependent on customer scaling requirements
- Application server memory demands are relatively light
- Database CPU loads are heavier than application server loads, in search-intensive environments
- Database memory demands are greater than for bulk processing

- Storage demands shift from sequential operation to random

CPU considerations in run time processing

Run time processing CPU loads are relatively light in comparison to bulk loads.

Typically, the Master Data Engine places a larger burden on the server CPU subsystem when compared to the application server. Generally, IBM Initiate® Master Data Service® installations deploy ~2 cores on a typical application server, and 4 cores on the database server. In larger scale deployments (greater than 20 transactions per second), a custom sizing might be required to determine the appropriate application and database server core counts. Typical large-scale installations deploy 2 - 4 quad core application servers and mirrored 8-way to 16-way database servers. Though the CPU ratio is 1:1, the application server CPU workload is lighter-weight. The application server cores are not saturated and the nodes are used primarily for load balancing and high availability purposes.

Memory considerations in run time processing

The Master Data Engine memory profile flips when making the transition from bulk to run time operations.

Memory requirements on the application server tend to drop, while the memory footprint on the database server increases. There are exceptions to this theory in deployments with unique context count demands, but in general, the application server memory footprint is much smaller than the database server footprint. The primary reason for the larger database server memory footprint is the active working set size; the "front end" SEARCH and MATCH activities work extensively against CMPD1 (the index for CMPD), CMPD (the database table that stores standardization and comparison strings), and BKTD2 (index to BKTD table that stores bucketing data). The interaction rates are much higher when these objects are cached in near memory. Thus, the preferred strategy is to size the database server buffer cache to allow these objects to reside in memory to maximize the buffer cache hit rate.

Depending on the way you search, "back end" attribute data retrieval can become a significant workload also. In performance-sensitive environments, pay special attention to how much data you are retrieving on average per search. This information can be tracked through the Master Data Engine TIMER log.

Storage considerations in run time processing

Storage performance is critical to the run time processing phase.

The key characteristics of the Master Data Engine input and output (I/O) footprint for the run time workload are:

- Runtime workload is online transaction processing (OLTP) style
- Reads are random (not sequential like most data warehouses)
- 80 percent to 90 percent reads, and 20 percent to 10 percent writes
- The engine interaction mix (that is, the distribution of interactions like MEMGET, MEMPUT, and MEMSEARCH) determines the size of the I/O workload. Different interactions have different I/O footprints.

Consult your IBM services representative for an I/O footprint estimate for your particular deployment. You can use that footprint estimate to determine what your SAN level requirements are. I/O subsystems are much easier to build at the start of a project (by using horizontally scalable solutions) than they are to redeploy once the installation has gone live. Corrective actions are often limited to caching

solutions at the database tier, so focus on implementing a correctly sized I/O subsystem that is scalable at the start of your project.

Database considerations in run time processing

The storage underlying the database can typically be a performance bottleneck.

High search throughputs typically drive high input and output (I/O) rates on the database server. This either overwhelms an undersized I/O subsystem, or puts CPU pressure on the database as it services requests from the application server.

Appendix H. Operational Monitoring with JConsole

The JConsole JMXMP (Java Management Extensions) tool is packaged with the Master Data Engine and provides a method for monitoring performance and resource consumption.

JConsole (officially known as the Java Monitoring and Management Console) offers access to dynamic MBeans whose attributes contain up-to-date data that is of interest to System Administrators.

When you create your Master Data Engine instance, the madconfig utility prompts you to enter an engine management port number. This number is the port used by JConsole to monitor the Master Data Engine from a remote system. The default value is 1199.

The JConsole application is located in the `/_jvm/bin` directory where you installed the Master Data Engine (*MAD_ROOTDIR*`/ibm/initiate/engine`*x.x.x*`/_jvm/bin/jconsole.exe`).

JMXMP is the protocol used for JMX (Java Management Extensions) communication. JMX replaces the previously used RMI protocol. JMXMP is based on TCP sockets and relies on SASL (Simple Access Security Layer) for security. This method is generally more secure than RMI. When a Master Data Engine instance is created, the useSSL variable is set to false in the `com.initiate.server.jmx.jmxmp.cfg` configuration file. If your implementation uses SSL, you must change this setting.

Table 73. Settings for useSSL within com.initiate.server.jmx.jmxmp.cfg

Using SASL only:	useSSL=false
Using SSL:	useSSL=true

If you are using a version of JConsole that is different from the one included with the Master Data Engine, you must copy one file before launching JConsole. Copy the `jmxremote_initiate.jar` file from your *MAD_ROOTDIR*`/lib/jvmext` directory into the JRE `lib/ext` directory.

Any vendor applications that connect to the Master Data Engine through JMX must have the `jmxremote_initiate.jar` file in the application classpath.

To continue availability of an RMI JMX connection, you can uncomment the RMI example in the `jmx.xml` file. The `jmx.xml` file is located in your `type_name/conf` directory. This scenario occurs only in implementations where a customer external integration already uses RMI.

Important: If you uncomment the RMI example, do not remove the JMXMP connector as IBM Initiate® Workbench always uses this method.

Additional monitoring options: In addition to JConsole, other monitoring options include the JMX Browser, which is embedded in IBM Initiate Workbench, and the web-based Performance Log Manager.

Additional information

To learn about the JMX Browser, see "Operational monitoring" in *IBM Initiate Workbench User's Guide*.

To learn about the Performance Log Manager, see *IBM Initiate Master Data Service Software Operations Guide*.

To learn about the JConsole architecture and features, see http://java.sun.com

Accessing JConsole

Use this procedure to access the JConsole monitoring tool. The JConsole JMXMP (Java Management Extensions) tool provides a method for monitoring Master Data Engine performance and resource consumption.

Procedure

1. From the *MAD_ROOTDIR*/ibm/initiate/engine*x.x.x*/_jvm/bin/ directory, run the `jconsole.exe` file.
2. Choose **Remote Process** and supply connection information in the form of:

   `service:jmx:jmxmp://host:port`

   where *host* is the server running the Master Data Engine and *port* is the associated Master Data Engine Management port.
3. Enter your username and password. The username must be part of the IBM Initiate® Master Data Service® Administrator group.

Results

The connection information is written to the Master Data Engine log file during startup.

For additional information about connection on various platforms, see: http://java.sun.com/j2se/1.5.0/docs/guide/management/faq.html.

JConsole Mbeans tab

JConsole includes the MBeans tab.

Under the com.initiatesystems tree are nodes for LDAP, Interactions, Log4j, and a node for the Master Data Engine instance that you requested on the connection dialog. Within the node for the instance name are nodes for Jobs, Listeners, Metadata, and ThreadPools. For each item, you can view an Attributes or Info tab. For monitoring purposes, view the contents of the Attributes tab. The Info tab provides the MBean name and the associated Java classname. Some nodes also have an Operations tab.

If you have implemented third-party callouts, you also see a Callouts node.

Listeners node

To view information about Master Data Engine transactions, go to com.initiatesystems > Interactions. Within the MPINET *instance_name* node is the Master Data Engine port number node, and below that is the Interactions node.

If the Interactions node does not appear, go to com.initiatesystems > MPINET *instance_name*> Listeners. Under the port number for the instance, choose the Operations node and click the refreshIxnStats button to refresh the Interactions node.

To view the activity of an interaction, select the Interactions node to see a list of all interactions. Select an interaction from the list, expand the node, and select the Attributes node. Each time an interaction is requested, it is logged in this view. The view does not automatically change if it is in focus. Click the Refresh button to update the display.

This table describes each Interaction MBean attributes.

Table 74. Interaction MBean attributes

MBean attribute	Description
AvgBktCands	Average number of member candidates returned from buckets
AvgRcvSize	Average size of messages received by the Master Data Engine
AvgSndSize	Average size of messages sent by the Master Data Engine to clients
AvgTicks	Average number of milliseconds
MaxBktCands	Maximum number of member candidates returned from buckets
MaxRcvSize	Maximum size of messages received by the Master Data Engine
MaxSndSize	Maximum size of messages sent by the Master Data Engine to clients
MaxTicks	Maximum number of milliseconds
MinRcvSize	Minimum number of member candidates returned from buckets
MinSndSize	Minimum size of messages received by the Master Data Engine
MinTicks	Minimum number of milliseconds
TotBktCands	Total number of member candidates returned from buckets
TotBktSrchs	Total number of bucket searches
TotExecs	Total number of times this interaction was called
TotGood	Total number of times this interaction returned a response with no error
TotRcvSize	Total size of messages received by the Master Data Engine
TotSndSize	Total size of messages sent by the Master Data Engine to clients
TotTicks	Total number of milliseconds

Log4j node

Select Log4j to view log settings and error log notifications. Log4j handles logging for the Master Data Engine. Within Log4j, there is a root Appenders tree. From this tree, you can control how logging is done in the Java application.

ThreadPools node

Through the ThreadPools node within com.initiatesystems > MPINET *instance_name*, you can monitor the service threads or context objects connected to the Master Data Engine. From the ThreadPools node, expand the Listener node and select Attributes.

The context pool size is the number of Master Data Engine threads that are started concurrently. Each context pool has its own connection to the database and can operate independently of the others. If you have your context pool set at 5, for example, you can send in five searches, gets, or puts at the exact same millisecond, and they are all processed concurrently. If six are sent, then the first five process while the sixth waits for the next free context.

CurrentContexts is the number of context threads currently in use. MaxContexts shows the peak number of contexts that have been in use at one time.

By double-clicking the value field for ContextPoolSize, CurrentContexts, or MaxContexts, you can view an activity chart. Click Discard chart to return the normal display.

Additional information: For additional information about the MBeans displayed in JConsole and each interaction, see "Operational monitoring" in *IBM Initiate Workbench User's Guide*.

For additional documentation about JConsole, see to http://java.sun.com and search for JConsole.

For more information about Log4j, see http://java.sun.com.

JConsole administrative actions

If you log in to JConsole with an administrative user name and password, you can bounce or stop the Master Data Engine, or perform a thread dump. These actions are performed from the org.tanukisoftware.wrapper node.

System attributes

Expand the WrapperManager node and choose Attributes to view current settings.

System operations

Expand the WrapperManager node and choose Operations to view the available operations.

To stop the Master Data Engine instance, click stop. The instance is stopped and connection to JConsole is terminated. You must restart the Master Data Engine and reconnect JConsole.

To bounce the Master Data Engine, click restart. The instance stops and then restarts. Connection to JConsole is lost, but you can reconnect after waiting a few moments.

In some instances when you are working with IBM Software Support, they might need a thread dump. By clicking requestThreadDump, the JVM thread information is output into the *instance_name*.out log directory in the instance log directory.

System overview

If you log in to JConsole with an administrative password, you can view additional information from the Overview tab. Click the Memory, Threads, or Classes tabs to access graphs of current processes.

For more information about the Overview tab, see http://java.sun.com.

Appendix I. AES encryption

Advanced Encryption Standard (AES) is an additional method for encrypting passwords for the Master Data Engine, Message Broker Suite, and IBM Initiate Web Reports.

After you install your Master Data Engine, you must generate an AES key. Next, you must encrypt a password. The madpwd3 utility is used to encrypt password and requires the AES key and initialization vector (iv).

After you have generated your AES key and encrypted password, you then use the madconfig utility to create your Master Data Engine or Message Broker Suite instances. The madconfig create_instance process prompts you to indicate your use of an AES-encrypted password by entering `pwd3`. The madconfig utility then requires you to provide AES key and iv files.

When creating the Master Data Engine instance, you must also identify your JSSE/JCE AES cipher provider. You can accept the default setting from the `java.security` file, unless your organization uses a JCE provider other than the default included with the provided JVM.

When prompted to enter the password, you type the encrypted output obtained from running the madpwd3 utility.

After your Master Data Engine instance is created, these variables are set in your `com.initiate.server.system.cfg` file.

```
MAD_SSLLIB=ssleay32.dll
MAD_SSLCRYPTOLIB=libeay32.dll
MAD_SSLAESKEYFILE=MAD_ROOTDIR\\conf\\initiateaeskey.
dat
MAD_SSLAESIVFILE=MAD_ROOTDIR\\conf\\initiateaesiv.da
t
```

For your broker instances, the same variables are set in your `services.ini` file. The values set for the variables depend upon your responses to the madconfig utility prompts.

Also for your Master Data Engine, these Java properties are added in your `jdbc.properties` file. Again the values might be different for your installation.

```
mad.password.scheme=PWD3
mad.aes.key.file=MAD_ROOTDIR\\conf\\initiateaeskey.d
at
mad.aes.iv.file=MAD_ROOTDIR\\conf\\initiateaesiv.dat
mad.aes.provider=SunJCE

password3=99CA56BDF62638567F456941650237AB
aeskeyfile=MAD_ROOTDIR\\conf\\initiateaeskey.dat
aesivfile=MAD_ROOTDIR\\conf\\initiateaesiv.dat
aesprovider=SunJCE
```

The `ldap.properties` use the default IBM Initiate® Master Data Service® AES key and IV files. By default, the globaladmin and bind user passwords are encrypted with AES 256-bit. This cypher-strength requires the unrestricted policy files to be installed.

```
embedded.ldap.security.adminpassword3=99CA56BDF62638567F456941650237AB
embedded.ldap.security.aeskeyfile=MAD_ROOTDIR\\conf\
\initiateaeskey.dat
embedded.ldap.security.aesivfile=MAD_ROOTDIR\\conf\\
initiateaesiv.dat
embedded.ldap.security.aesprovider=SunJCE

internal.ldap.security.bindpassword3=99CA56BDF62638567F456941650237AB
internal.ldap.security.aeskeyfile=MAD_ROOTDIR\\conf\
\initiateaeskey.dat
internal.ldap.security.aesivfile=MAD_ROOTDIR\\conf\\
initiateaesiv.dat
internal.ldap.security.aesprovider=SunJCE
```

By default only the embedded and internal prefixes are set. You can also use AES for your external prefixes. The aesprovider property is empty if you decide to use settings from your `java.security` file for JCE cipher selections. Otherwise, this property contains the value that you provide to the madconfig utility prompts.

Related task

"Generating AES keys and password"

Related reference

"AES policy JAR files" on page 246)

Generating AES keys and password

Use the OpenSSL command-line tool, which is included with the Master Data Engine, to generate AES 128-, 192-, or 256-bit keys. The madpwd3 utility is used to create the password.

Before you begin

Verify that these environment variables are set in your `com.initiate.server.system.cfg` file:

- On Microsoft Windows, set `MAD_SSLLIB=ssleay32.dll` and `MAD_SSLCRYPTOLIB=libeay32.dll`
- On IBM AIX, Linux, or Solaris, set `libssl.so` and `libcrypto.so`
  - Microsoft Windows command example: `set MAD_SSLLIB=ssleay32.dll`
  - IBM AIX, Linux, or Solaris command example: `export MAD_SSLLIB=libssl.so`

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `\bin` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\bin`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/bin`
2. On the command line, type:
   - For 128-bit key:
     `openssl enc -aes-128-cbc -k secret -P -md sha1`
   - For 192-bit key:
     `openssl enc -aes-192-cbc -k secret -P -md sha1`
   - For 256-bit key:
     `openssl enc -aes-256-cbc -k secret -P -md sha1`

"secret" is a passphrase for generating the key.

The output from the command is similar to:

- 128-bit:

```
salt=92AE31A79FEEB2A3
key=770A8A65DA156D24EE2A093277530142
iv=F5502320F8429037B8DAEF761B189D12
```

- 192-bit:

```
salt=D495560961CCCFE0
key=4D92199549E0F2EF009B4160F3582E5528A11A45017F3EF8
iv =35B2FF0795FB84BBD666DB8430CA214E
```

- 256-bit:

```
salt=263BC60258FF4876
key=B374A26A71490437AA024E4FADD5B497FDFF1A8EA6FF12F6FB65AF2720B59CCF
iv =7E892875A52C59A3B588306B13C31FBD
```

3. Copy the key output into a `.dat` file, excluding the "key" characters. For example, a "`myaeskey.dat`" file for 256-bit can contain:

   ```
   B374A26A71490437AA024E4FADD5B497FDFF1A8EA6FF12F6FB65AF2720B59CCF
   ```

   Do not have a carriage return or line feed in the file.

4. Place the iv output into another `.dat` file, excluding the "iv" characters. For example, a `myiv.dat` file can contain:

   ```
   7E892875A52C59A3B588306B13C31FBD
   ```

   Do not have a carriage return or line feed in the file.

   Important: If the key and iv are generated with another tool, you must verify that the result is hex-encoded and that the size of the key for 128 is 32 characters, 192 is 48 characters, and 256 is 64 characters. The hex-encoded iv is 32 characters in length. Hex encoding means that each character in the key and iv are converted to its hexadecimal equivalent. For example, the letter "A" is "41" in hexadecimal. Hex encoding eases the storage and transport of the key and iv because the non-encoded versions of these items can contain ASCII control character sequences.

   The IBM Initiate Master Data Service implementation expects the Cipher-Block-Chaining (CBC) method. The "salt" is not used in any future decryption operations and can be discarded.

5. Run the madpwd3 utility to generate the encrypted password. The madpwd3 utility allows for the key and iv to be entered either from a file or directly on the command line. Use the `-keyfile` and `-ivfile` options to specify as a file or use the `-key` and `-iv` options to enter them at the command prompt. There is no limit on the length of the password input and the output length is variable. For example:

   ```
   madpwd3 -keyfile myaeskey.dat -ivfile myaesiv.dat -in foopass
   ```

   generates this output:

   ```
   PLAINTEXT = (foopass)
   ENCRYPTED = (99CA56BDF62638567F456941650237AB)
   DECRYPTED = (foopass)
   ```

   The madconfig utility prompts for this information when creating Master Data Engine and Message Broker Suite instances.

Results

AES decryption in the Master Data Engine is supported via native OpenSSL APIs as well as Java APIs.

Related reference

"madpwd3 utility" on page 128

AES policy JAR files

Depending on your choice of cipher strength (128, 192 or 256) for your AES password, you might need to add the unrestricted policy JAR files.

The unrestricted policy JAR files are added to either the*$JAVA_HOME*/jre/lib/security or *$JAVA_HOME*/lib/security directories. AES 128 does not require use of the policy files. The policy JAR files are named: local_policy.jar and US_export_policy.jar.

These policy files are included with the JVM embedded with the Master Data Engine. However, IBM Initiate Web Reports use a different JVM from the Master Data Engine and you might need to download these files for bit strengths greater than 128.

For the Sun JVM, download the policy JAR files from:

- JDK 1.5

  https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jce_policy-1.5.0-oth-JPR@CDS-CDS_Developer
- JDK 1.6

  https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jce_policy-6-oth-JPR@CDS-CDS_Developer

For the IBM JVM, download from:

- JDK1.5

  http://www.ibm.com/developerworks/java/jdk/security/50/
- JDK 1.6

  http://www.ibm.com/developerworks/java/jdk/security/60/

AES password test

You can test the usage of your AES password by running the madconfig utility test_datasource target, and supplying the encrypted database password.

AES usage appears in the engine log similar to this example:

```
TRACE MAD_DbxFixConnStr:
szConnStrIn='DSN=foodsn;UID=username1;PWD3=99CA56BDF62638567F456941650237AB'.
```

Passwords can also be decrypted by using the Java class file com.initiatesystems.common.util.Pwd3Helper. The class file is contained in madcommon.jar. Usage:

```
com.initiatesystems.common.util.Pwd3Helper -keyfile filename -ivfile
filename -in encrypted -provider provider optional
```

Appendix J. Interceptor tool

When performing an upgrade or routine maintenance, recording interactions executed on one Master Data Engine and replaying those interactions on other Engines can reduce downtime.

Important: If you have not previously used the Interceptor tool, consult IBM Software Support for guidance.

The Interceptor tool combines two processes: the Recorder and the Replayer. The Recorder intercepts interactions destined for a Master Data Engine, logs them to an "interaction data" file, and then forwards the interactions on to the engine in a transparent manner. The interactions in the file can then be replayed on a destination engine to facilitate routine maintenance or upgrade.

1. Maintenance purposes: Within a multi-engine configuration, you can use Recorder as you step through a maintenance process for one engine, and then use Replayer to execute the process identically for other engines. (This functionality is available only with engines running releases 9.2, 9.5 and 10.0.)

   If you are performing maintenance, you might find it easier to use the replayer targets of the madconfig command-line utility (rather than using the Replayer API). Choosing to use the madconfig utility replayer targets or the Replayer API depends on whether you need to manipulate the recorded interactions before replaying them.
2. Upgrade purposes: You can use Recorder and Replayer to facilitate upgrades from release 8.7, 9.0, 9.2, or 9.5 to release 10.0.

   If you are performing an upgrade, use the Replayer API (rather than the madconfig utility replayer targets). See "Replayer API" on page 251.

Inputs to Recorder can come only from Master Data Engine releases 8.7, 9.0, 9.2, 9.5, or 10.0. Replayer can output to release 9.2, 9.5, or 10.0 only.

Whether performing maintenance or upgrade, use the madconfig utility recorder targets to create the interaction data file.

The Interceptor process is:

1. Upon starting the Recorder, interactions intended for a source Master Data Engineare directed to the Recorder process.
2. The Recorder writes the interactions out to the interaction data file.
3. The Recorder passes the interactions transparently to the engine.
4. When you are ready to replay the interactions to a destination engine, start the madconfig utility replayer or your own application created with the Replayer API. The Replayer reads from the interaction data file created by the Recorder in step 2.
5. The madconfig utility replayer (or your own application) iterates through the interactions in the file and replays them to the destination engine.

The Interceptor process figure illustrates steps 1 through 5 in flow chart format.

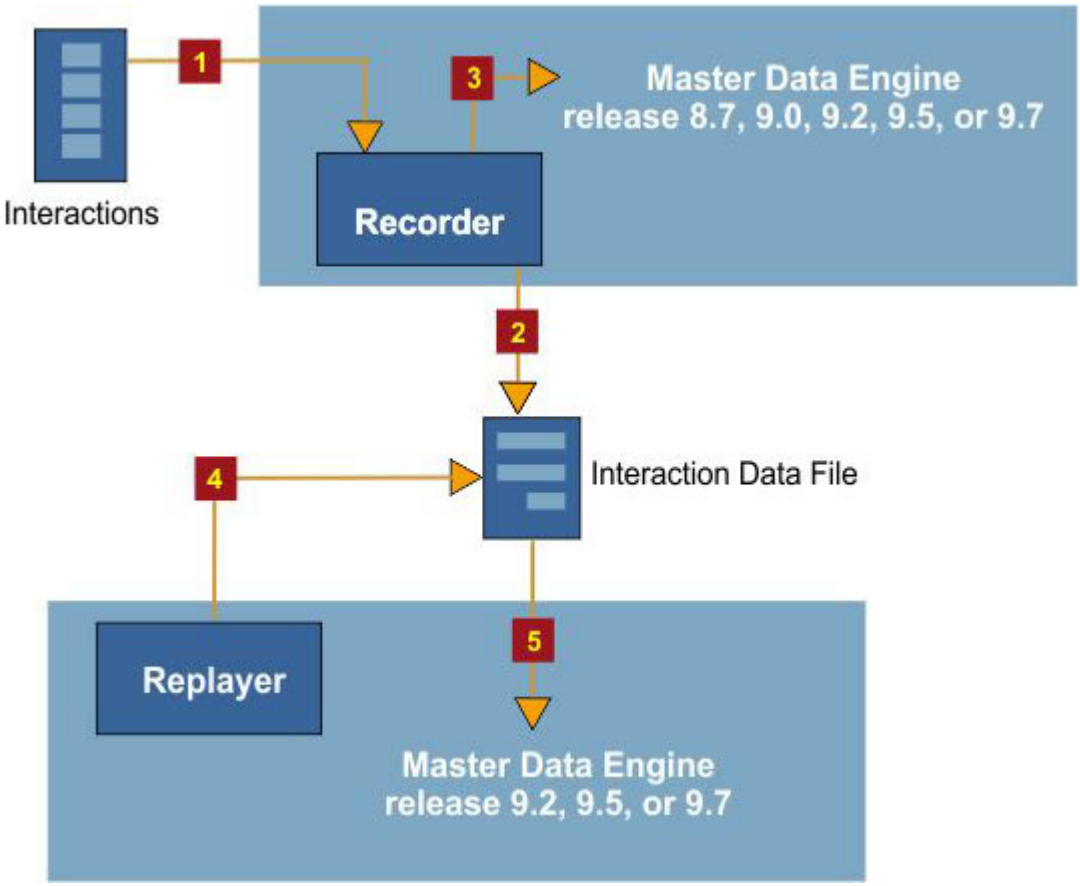

Figure 8. The Interceptor process

To the Master Data Engine, both the Replayer and the Recorder appear as clients issuing interactions. These two processes run independently of each other. The speed of the replay is dependent on the destination engine.

Starting the Interceptor Recorder with the madconfig utility

You can use the madconfig utility to start the Interceptor Recorder.

About this task

Only one instance of the start_recorder target can be run at any given time on a single Master Data Engine.

Until you stop the Recorder, it continues to write interactions to the interaction data file, gradually increasing the size of the file. Make sure that you have adequate space on the file system and monitor the file as it grows.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the applicable command:

   Microsoft Windows: `madconfig start_recorder`

IBM AIX, Linux, or Solaris: `madconfig.sh start_recorder`

3. Enter an existing Master Data Engine instance name. This name is the "source" Master Data Engine.
4. Enter the Master Data Engine user password.
5. Enter interaction types. Default value: [ ] By default, interactions of all types are recorded. For a complete list and descriptions of interactions, see the *IBM Initiate Workbench User's Guide* .
6. Enter the file name. By default, the file name is `Record.rcd` and it is written to the *MAD_HOMEDIR*`\inst\mpinet_`*name* directory. The file can be written to an alternative path by supplying the full path and file name. Do not include spaces in the file name.
7. Leave the command line open while the Recorder is running.

Stopping the Interceptor Recorder with the madconfig utility

When you have finished recording your Master Data Engine interactions, you can use the madconfig utility to stop the Interceptor Recorder.

About this task

Until you stop the Recorder, it continues to write interactions to the interaction data file, gradually increasing the size of file. Make sure that you have adequate space on the file system and monitor the file as it grows.

Procedure

1. From the command line you used to start the Recorder (the engine installation *MAD_ROOTDIR*`\scripts` directory, run the applicable command:

   Microsoft Windows: `madconfig stop_recorder`

   IBM AIX, Linux, or Solaris: `madconfig.sh stop_recorder`
2. Enter the Master Data Engine name.
3. Enter the Master Data Engine user name.
4. Enter the Master Data Engine user password.

Starting the Interceptor Replayer with the madconfig utility

You can use the madconfig utility to start the Interceptor Replayer if you are replicating interactions from one 10.0 Master Data Engine to another 10.0 engine.

About this task

If you are performing an upgrade, use the Replayer API.

You can run only one instance of the madconfig utility start_replayer target at any given time on a single engine.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Run the applicable command:

   Microsoft Windows: `madconfig start_replayer`

IBM AIX, Linux, or Solaris: `madconfig.sh start_replayer`

3. Enter an existing Master Data Engine instance name. This version of the instance must be release 10.0 or later.
4. Enter the Directory with Recorded file. Point to the location of the file created by the Recorder.
5. Enter the Recorded file to Replay. Default value: `Record.rcd`
6. Enter the Start Line Number. Indicate where in the recorded file you want the Replayer to begin executing interactions to be sent to the destination engine. The Replayer executes all interactions from the line number you supply until the end of the file. The default is 1.

Results

The Replayer creates a `replay.log` file to log the `start_replayer` actions if these conditions are met:

- that `log4j-*.jar` is in the classpath upon class creation
- that `log4j-*.jar` is in the classpath for the replayer script
- that you have this VM argument set

  `-Dlog4j.configuration=interceptor.log4j.xml`

The Replayer creates the `replay.log` file in the same directory as the recorded file.

The Replayer, upon starting, establishes a connection with the destination engine, scans the interaction data file (one interaction per line), and sends the interaction to the new engine. There is no madconfig utility stop_replayer target; the Replayer stops when it reaches the end of the interaction data file.

Related reference

"Replayer API" on page 251

The fields of the interaction data file

The Interceptor Recorder intercepts interactions that are destined for a Master Data Engine and logs them to an interaction data file.

Each line in the interaction data file represents an individual interaction. Each interaction contains these fields:

- system time
- message size
- message number
- version number
- message
- interaction type
- user ID
- member ID in the source system

For example:

```
1258673233759|342|0|98|4d454d505554000000006e00000002007379737465 6d00737
97374656d
00000000000000000000000000000000000000000000000000000000000000
00000003000000010000000100000000000000000|MEMPUT|system|[RMC:870504]
```

- The system time is: `1258673233759`

  The entry is a UNIX and Linux system epoch time that, in this case, converts to December 4, 2009 7:28:29pm
- The message size is: `342` bytes
- The message number is: `0`

  The 0 indicates a native message type. The other possible message type is 1, which is a Java message. This message distinction is internal to IBM Initiate® Master Data Service®.
- The version number is: `90`

  This number identifies the Master Data Engine version number.
- The message, a hex-encoded string containing the message for the interaction, is:

  ```
  4d454d505554000000006e0000000200737973746564d0073797374656d
  00000000000000000000000000000000000000000000000000000000
  00000000000000003000000010000000100000000000000000
  ```
- The interaction type is: `MEMPUT`

  Other possibilities include MEMDROP, MEMDELETE, MEMMERGE, and MEMUNMERGE. By default, the Recorder intercepts all interactions.
- The user ID is: `system`
- The member ID in the source system is: `[RMC:870504]`

  The format is`[`*`Source`*`:`*`MemberID`*`]`

Replayer API

Use the Replayer API to facilitate upgrade from an 8.7, 9.0, 9.2, or 9.5 engine to a 10.0 engine.

If you are performing maintenance that does not require manipulating recorded interactions before replaying them, use the madconfig utility replayer target.

The Replayer API allows you to account for changes in the format of interactions between product releases as well as format changes you might have made to suit your implementation.

Use the Replayer API to create Java applications to decode, map, and execute the interactions of the interaction data file. Specifically, your application:

1. Decodes (in other words, reads) the interaction from the interaction data file.
2. Maps the interaction from the format used by the source Master Data Engine to the format used by the destination engine.
3. Executes the interaction by sending the newly formatted interaction to the destination engine.

Of the three steps, IBM Initiate® Master Data Service® provides code to perform decode and execute steps, as well as sample code for the mapping step. Use that code to create custom mappings to accommodate the interactions of your own configuration.

IBM Initiate® Master Data Service® does provide several sample applications that cover most upgrade cases.

The Replayer API is made available through `Interceptor.zip` file, which is installed by default in this directory:

- **Microsoft Windows**: `[`*`engine`*`]\lib\sdk\`

- **IBM AIX, Linux, or Solaris**: [*engine*]/lib/sdk/

The compressed file includes:

1. A lib directory containing JAR files. When you add the JAR files to your classpath, you must specify each JAR file by name. Classes are created if you include .../lib/* in the classpath.
2. An example directory of sample Java programs for replaying 8.7, 9.0, 9.2, and 10.0 interactions along with sample mappings for 8.7 and 9.0. The examples differ only in how the mapping is performed.
3. A doc directory contains Javadocs for the Replayer API.
4. The replayer.bat and replayer.sh executable programs can be used to demonstrate the execution of the sample applications.
5. A readme file with basic information about the Recorder API.

The Replayer API consists of a single Replayer class (com.initiatesystems.hub.interceptor.replayer) with various methods for:

- decoding the interaction data file and replaying interactions, and
- getting and setting the host and port of the engine, the name of the interaction data file, and the starting line for replaying the interactions.

For detail about the individual methods, see the *IBM Initiate Java SDK Javadoc Information* available within the doc directory of the Interceptor.zip file.

The decodeIxns method returns an iterator. To take advantage of the iterator, ensure that your application:

1. Sets up a loop that makes a call to hasNext() of the returned iterator. Set up the loop to iterate as long as hasNext() returns true.
2. Within the loop, obtain each interaction by calling next() on the returned iterator.

Related task

"Starting the Interceptor Replayer with the madconfig utility" on page 249

Related reference

"Interceptor mapping files"

Interceptor mapping files

Mapping files are used to map interactions between versions of the Master Data Engine.

The Interceptor.zip file contains sample Java applications for replaying interactions recorded with Master Data Engine releases 8.7, 9.0, 9.2, 9.5, and 10.0. The interactions can be replayed only to a 9.2, 9.5, or 10.0 release version of the engine. The Interceptor.zip file also contains two mapping files (MapIxn87.java and MapIxn90.java) that you can use to map from version 8.7 to 10.0 and from version 9.0 to 10.0. You can use these files if you have not customized the interaction formats for your installation.

Specifically, ReplayerClient87.java calls the decodedIxns method to decode the interaction data file created by the 8.7 engine. A call to the decodeIxns method returns an iterator by which the code obtains each interaction. The iterator invokes

the mapIxn method of the MapIxn87 class to perform the mapping from 8.7 to 10.0. Finally, the code calls the replay method to replay the interactions against the 10.0 Master Data Engine.

The same mechanism applies to all subsequent releases. For example, ReplayerClient90.java executes the same steps except that it invokes the MapIxn90 class to perform the mapping from 9.0 to 10.0. Similarly,

- ReplayerClient92.java invokes MapIxn92.
- ReplayerClient95.java invokes MapIxn95.
- ReplayerClient97.java invokes MapIxn97.

Because the source and destination engines are both 10.0, no mapping is required.

Use MapIxn87.java and MapIxn90.java as templates, as you devise the mapping required for the upgrade of your particular installation.

Other mapping scenarios

In addition to the standard mapping from one release to another, you might need to write custom mapping code to manage these complexities:

- To account for miscellaneous business logic that has changed between the two releases. For example, between the 9.2 release and the 10.0 release, you might have added a new attribute or changed from one type of MEMPUT interaction to another.
- To parse the interaction data file based on an interaction user ID, interaction type, version number, and so on. For example, you might want to specify that the interactions associated with one source system are to be replayed to one engine, while interactions from another source system are to be replayed to a different engine.

To create custom mapping, use the Replayer API. The ReplayerClient*.java and the MapIxn*.java are sample files, so you can either modify those files or use them as a base for new files. You can also write your own version of ReplayerClients. Use the information provided in the Java docs that are included in the Interceptor.zip file to write your own clients.

Running your custom Replayer application

If you have written a custom Replayer application, you must edit the replayer.bat or replayer.sh script so that the script calls your application.

About this task

The replayer.bat and replayer.sh scripts that are included within interceptor.zip file call ReplayerClient90. Edit the script if you want to call a different version of ReplayerClient or to call your own Replayer application.

You can run multiple Replayer applications simultaneously, though the Replayer itself is single threaded.

Procedure

1. Extract the interceptor.zip file.
2. Edit the replayer.bat or replayer.sh script available with in interceptor.zip file so that the script calls your custom application, or another version of ReplayerClient.

3. Open a command line from the directory in which you extracted the `interceptor.zip` file and run the command for your operating system:

   Microsoft Windows: `replayer.bat [targetHost] [targetPort] [logFile] [startLine]`

   IBM AIX, Linux, or Solaris: `replayer.sh [targetHost] [targetPort] [logFile] [startLine]`

   where

   - `targetHost` is the host name for the destination Master Data Engine.
   - `targetPort` is the port for the destination engine.
   - `logFile` is the interaction data file. By default this setting is `record.rcd`
   - `startLine` is the line in interaction data file to begin processing; the default is 1. This value is optional.

Appendix K. FIPS compliance

Federal Information Processing Standards (FIPS) is a security standard developed by the U.S. federal government. FIPS compliance is required in many U.S. federal government installations.

If you are installing in a U.S. federal government environment, your implementation must be FIPS-compliant. The Master Data Engine, Message Broker Suite, and engine command-line utilities that communicate over SSL can be FIPS140-2 enabled.

The madconfig utility includes additional prompting for creating a FIPS-enabled instance when you start the utility using this command:

```
madconfig -Dmad.sec.show.prompts=true create_instance
```

Two important items to note when creating FIPS-compliant instances:

- Several of the Master Data Engine Java Runtime Engine files are updated when creating a FIPS-compliant instance. Because of this configuration, you cannot have a FIPS-compliant instance and a non-FIPS compliant instance sharing the same Master Data Engine (`MAD_ROOTDIR`).
- Also, if you uninstall and then reinstall a Master Data Engine, you must first remove the FIPS-compliant instance and recreate it after reinstallation of the engine.

Related information

For guidelines on installing the IBM Initiate Master Data Service components in a U.S. government environment, see *IBM Initiate Master Data Service Security Technical Implementation Guide (STIG)* .

To create FIPS-compliant brokers, see *IBM Initiate Master Data Service Message Broker Suite Reference* .

If you are installing IBM Initiate Web Reports or IBM Initiate Inspector, see *IBM Initiate Web Reports Installation and Configuration Guide* or *IBM Initiate Inspector Installation and Configuration Guide*

Enabling FIPS compliance in the Master Data Engine

If you are installing in a U.S. federal government environment, your Master Data Engine instance must be FIPS-compliant.

Before you begin

Review the FIPS compliance topic.

About this task

To enable FIPS compliance in the Master Data Engine, use the madconfig utility to create your instance. FIPS prompts are not a standard part of the madconfig create_instance process. To display the necessary SSL and FIPS prompts, you must use a specific command. This command instructs madconfig to expose specific SSL

and FIPS prompts during the instance creation. These prompts appear after entering the engine instance home directory.

The default key and truststore password is: rmi+ ssl.

Procedure

1. On the command line, go to the Master Data Engine installation directory (*MAD_ROOTDIR*). Then go to the `scripts` directory. For example:

   Microsoft Windows: `cd C:\Program Files\IBM\Initiate\Engine10.0.x\scripts`

   IBM AIX, Linux, or Solaris: `cd /opt/IBM/Initiate/Engine10.0.x/scripts`
2. Type this command:

   ```
   madconfig -Dmad.sec.show.prompts=true create_instance
   ```
3. The first two prompts are standard madconfig utility create_instance prompts. Provide the name of the engine instance and the instance home directory.
4. Type `y` to enable SSL when communicating with the Master Data Engine.
5. Enter the full path and name of the JSSE truststore for the Master Data Engine instance. The default value is *MAD_ROOTDIR*`/conf/ibmcorporationtrust.jks`
6. Enter the password for accessing the JSSE truststore for the Master Data Engine instance. Password characters are not displayed.
7. Enter the JSSE truststore type for the Master Data Engine instance. The default value is `JKS`.
8. Enter the full path and name of the JSSE keystore for the Master Data Engine instance. The default value is *MAD_ROOTDIR*`/conf/ibmcorporation.p12`.
9. Enter the password for accessing the JSSE keystore for the Master Data Engine instance.
10. Enter the JSSE keystore type for the Master Data Engine instance. The default value is `PKCS12`.
11. Enter `y` if you want to enable FIPS 140-2 compliance mode for the v instance.
12. Continue with the remaining create_instance prompts.

Results

The madconfig utility automatically updates your JRE and instance folders with the proper configuration for FIPS enablement as follows:

- `com.initiate.server.system.cfg` with `MAD_SSLFIPSMODE=1`
- `wrapper.conf` with `wrapper.java.additional.16=-Dcom.ibm.mdshs.jsse2.JSSEFIPS=true` (This example shows the numeric suffix of "16." The exact number in your file might vary.)
- java.security with these security providers if they do not exist, starting at order 2:
  - `security.provider.2=com.ibm.mdshs.crypto.fips.provider.IBMJCEFIPS`
  - `security.provider.3=com.ibm.mdshs.crypto.provider.IBMJCE`
  - `security.provider.4=com.ibm.mdshs.jsse2.IBMJSSEProvider2`
- java.security with these ssl socket providers if they do not exist:
  - `ssl.SocketFactory.provider=com.ibm.mdshs.jsse2.SSLSocketFactoryImpl`
  - `ssl.ServerSocketFactory.provider=com.ibm.mdshs.jsse2.SSLServerSocketFactoryImpl`

These JAR files are installed into your JRE `lib/ext` directory.

- `ibmjcefips.jar`

- ibmjcefw.jar
- ibmjceprovider.jar
- ibmjsseprovider2.jar
- ibmpkcs.jar

The embedded JVM that comes with the installation contains the appropriate Unlimited Strength Jurisdiction Policy Files in order to unlock high-strength ciphers. These files are located in your JRE lib/ext directory.

- local_policy.jar
- US_export_policy.jar

Related concept

Appendix K, "FIPS compliance," on page 255

Enabling FIPS compliance for command-line utilities

The MAD_SSLFIPSMODE variable is used to enable FIPS compliance within Master Data Engine command-line utilities that communicate over MPINET and SSL.

Procedure

1. In your Master Data Engine instance directory (*MAD_HOMEDIR*/*instance_name*/mpinet_*instance_name*/conf), open the com.initiate.server.system.cfg configuration file.
2. Set the MAD_SSLFIPSMODE variable to 1. The default is 0 (not enabled). For instance, to FIPS-enable the mpinetget utility you can use this environment configuration:

```
MAD_SECLIB=SSL
MAD_SSLLIB=libssl.so
MAD_SSLCRYPTOLIB=libcrypto.so
MAD_SSLCACERTFILE=/local/install/engine/conf/ibmcorporationcert.pem
MAD_SSLFIPSMODE=1
MAD_CONNSTR=localhost|16000
MAD_CTXLIB=MPINET
AuditLog=1
```

3. Setting the AuditLog variable to 1 places an entry in the log file specifying that SSL has been configured for FIPS mode. An SSL version of TLSv1 is only applicable to FIPS mode. Any other version is logged as incompatible and reset to TLSv1, as shown in this example log snippet from executing the FIPS-enabled mpinetnget utility:

```
16:06:27 mpinetget AUDIT MAD_SSL_load_crypto_lib: SSL configured for FIPS
140-2 compliance mode.
16:06:27 mpinetget AUDIT MAD_ComSetup: SSL version 'SSLv3' cannot be used
with FIPS 140-2 compliant mode. SSL version has been changed to TLSv1.
```

Debugging SSL and FIPS configuration

SSL debugging can be enabled by editing your instance wrapper.conf file.

Procedure

1. In your engine instance *MAD_HOMEDIR*/inst/mpinent_*instance_name*/conf directory, open the wrapper.conf file.
2. Add this line as a wrapper.java.additional property:

```
wrapper.java.additional.N=-Djavax.net.debug=ssl
```

Where N is the next logical number that can be used. When using the madconfig utility to start the instance, output similar to this example can display for your instance .out file. Notice the entry which states that IBMJSSE is in FIPS mode.

```
INFO | jvm 1| 2009/07/29 16:01:17 | IBMJSSEProvider2 Build-Level: -20090506
INFO | jvm 1| 2009/07/29 16:01:23 | setting up default SSLSocketFactory
INFO | jvm 1| 2009/07/29 16:01:23 | class
com.ibm.mdshs.jsse2.SSLSocketFactoryImpl is loaded
INFO | jvm 1| 2009/07/29 16:01:23 | IBMJSSE is in FIPS mode
```

Legal Statement

Notices and trademarks

This information was developed for products and services offered in the U.S.A.

Notices

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web

sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Other product and service names might be trademarks of IBM or other companies.

A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Index

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Contacting IBM

You can contact IBM for customer support, software services, product information, and general information. You also can provide feedback to IBM about products and documentation.

The following table lists resources for customer support, software services, training, and product and solutions information.

Table 75. IBM resources

Resource	Description and location
IBM Support Portal	You can customize support information by choosing the products and the topics that interest you at www.ibm.com/support/entry/portal/Overview/Software/Information_Management/IBM Initiate_Master_Data_Service
Software services	You can find information about software, IT, and business consulting services, on the solutions site at www.ibm.com/businesssolutions/
My IBM	You can manage links to IBM web sites and information that meet your specific technical support needs by creating an account on the My IBM site at www.ibm.com/account/
Training and certification	You can learn about technical training and education services designed for individuals, companies, and public organizations to acquire, maintain, and optimize their IT skills at http://www.ibm.com/software/sw-training/
IBM representatives	You can contact an IBM representative to learn about solutions at www.ibm.com/connect/ibm/us/en/

Providing feedback

The following table describes how to provide feedback to IBM about products and product documentation.

Table 76. Providing feedback to IBM

Type of feedback	Action
Product feedback	You can provide general product feedback through the Consumability Survey at www.ibm.com/software/data/info/consumability-survey

Table 76. Providing feedback to IBM (continued)

Type of feedback	Action
Documentation feedback	To comment on the information center, click the Feedback link on the top right side of any topic in the information center. You can also send comments about PDF file books, the information center, or any other documentation in the following ways: • Online reader comment form: www.ibm.com/software/data/rcf/ • E-mail: comments@us.ibm.com

Printed in USA

GI13-2612-00