Loading the Initial Data Set for a Sun Master Index

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–3406–14
December 2008

081223@21808

Contents

Loading the Initial Data Set for a Sun Master Index

The topics listed here provide information about generating the Initial Bulk Match and Load Tool for a master index application, configuring the match and load processes, matching bulk data, and loading bulk data into the master index database.

Note that Java CAPS includes two versions of Sun Master Index. Sun Master Index (Repository) is installed in the Java CAPS repository and provides all the functionality of previous versions in the new Java CAPS environment. Sun Master Index is a service-enabled version of the master index that is installed directly into NetBeans. It includes all of the features of Sun Master Index (Repository) plus several new features, including the Initial Bulk Match and Load Tool. Both products are components of the Sun Master Data Management (MDM) Suite. This document relates to Sun Master Index, however you can use the Initial Bulk Match and Load Tool with a repository-based master index application with some modification (see "Using the Initial Bulk Match and Load Tool With Sun Master Index (Repository)" on page 7).

What You Need to Know

These topics provide information you should know about configuring and using the initial load tool.

What You Need to Do

These topics provide instructions on how to match and load bulk data using the initial load tool.

- "Configuring the Environment" on page 15
- "Creating the Cluster Synchronizer Database" on page 16
- "Configuring the Initial Bulk Match and Load Tool" on page 16
- "Performing a Match Analysis" on page 24
- "Performing the Bulk Match" on page 28
- "Running the Bulk Match and Bulk Load in One Step (SQL*Loader Only)" on page 31
- "Loading Matched Data Using SQL*Loader" on page 34
- "Loading Matched Data Using the Data Integrator Wizard Bulk Loader" on page 36
- "Loading Matched Data Using the Data Integrator Command-Line Bulk Loader" on page 43

More Information

These topics provide additional information you should know when using the initial load tool.

- "Initial Bulk Match and Load Tool Configuration Properties" on page 18
- "Initial Bulk Match and Load Tool Logging Properties" on page 23

  "Data Integrator Bulk Loader Properties" on page 44

Related Topics

Several topics provide information and instructions for implementing and using a master index application. For a complete list of topics related to working with the service-enabled version of Sun Master Index, see "Related Topics" in *Developing Sun Master Indexes* .

Initial Bulk Match and Load Overview

The Initial Bulk Match and Load Tool (IBML Tool) gives you the ability to analyze match logic, match legacy data, and load a large volume of data into a master index application. One of the issues that arises during a data management deployment is how to get a large volume of legacy data into the master index database quickly and with little downtime, while at the same time cleansing the data, reducing data duplication, and reducing errors. The IBML Tool provides a scalable solution that standardizes and matches raw data and that can be run on multiple processors for better performance.

The IBML Tool consists of two components, the Bulk Matcher and the Bulk Loader. The Bulk Matcher compares records in the input data using probabilistic matching algorithms based on the Master Index Match Engine and based on the configuration you defined for your master index application. It then creates an image of the cleansed and matched data to be loaded into the master index. The Bulk Loader uses the output of the Bulk Matcher to load data directly into the master index database. Because the Bulk Matcher performs all of the match and potential duplicate processing and generates EUIDs for each unique record, the data is ready to be loaded with no additional processing from the master index application itself.

The following topics provides additional information about the Initial Bulk Match and Load Tool.

Using the Initial Bulk Match and Load Tool With Sun Master Index (Repository)

To take advantage of the IBML Tool if you are using Sun Master Index (Repository), you need to create a new service-enabled master index application. After you create the new application, copy the configuration files from your Sun Master Index (Repository) project and paste them to *NetBeans_Projects*/*Project_Name*/`src/Configuration`, replacing the existing configuration files. Regenerate the new master index application.

Initial Bulk Match and Load Process Overview

Performing an initial load of data into a master index database consists of three primary steps. The first step is optional and consists of running the Bulk Matcher in report mode on a representative subset of the data you need to load. This provides you with valuable information about the duplicate and match threshold settings for the master index application. Analyzing the data in this way is an iterative process. Based on the information from each run, you can modify the Bulk Matcher configuration file by changing the blocker query used for matching, the match string, and the duplicate and match thresholds. After you reconfigure the Bulk Matcher, rerun the analysis reports to verify the results. You can repeat this procedure as often as needed until you are satisfied with the match process settings. Once the configuration is finalized, make sure to modify the master index application configuration files to match the Bulk Matcher configuration file.

The second step in the process is running the Bulk Matcher in matching mode. The Bulk Matcher processes the data according to the query, matching, and threshold rules defined in the Bulk Matcher configuration file. This step compares and matches records in the input data in order to reduce data duplication and to link records that are possible matches of one another. The output of this step is a master image of the data to be loaded into the master index database. The final step in the process is loading the data into the master index database. This can be done using either Oracle SQL*Loader or the Data Integrator Bulk Loader. Both products can read the output of the Bulk Matcher and load the image into the database.

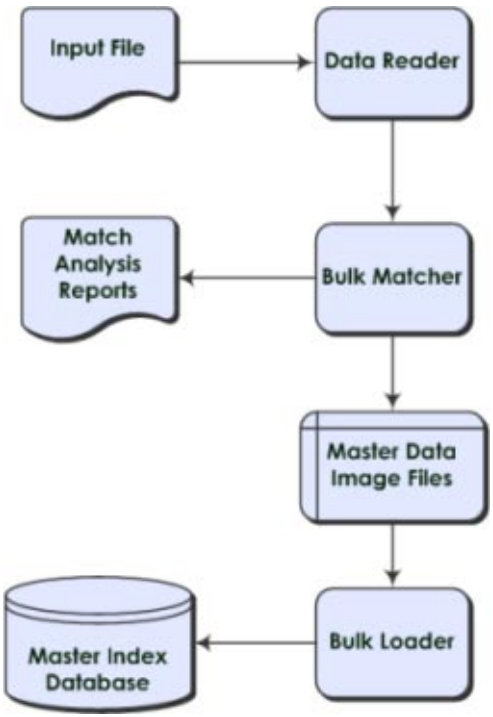


FIGURE 1 Initial Bulk Match and Load Tool Process Flow

Data Preparation, Matching, and Loading Procedure Overview

The IBML Tool was designed to build on the features provided by the Data Cleanser and Data Profiler tools that are also generated from a master index application. These tools help you analyze the quality of your data and fix any data integrity issues prior to matching the data using the Bulk Matcher. The Data Cleanser generates a file from your legacy data that can be read by the Bulk Matcher. Together, the Data Profiler, Data Cleanser, and IBML Tools provide a complete solution for analyzing, cleansing, matching, and finally loading your legacy data into a master index database, no matter the quality of the original data set.

The following steps outline the procedure to follow to prepare, match, and load legacy data.

1. Make sure the master index application is configured and generated and the database is created.
2. Extract the data to either a Data Integrator staging database or a flat file.

3. Begin queuing any transactions that occur from the time you extract the data until the database is loaded and ready to go live.
4. Run the Data Profiler and Data Cleanser against the extracted data. This applies cleansing rules to the data and produces a file that can be read by the Bulk Matcher. This step is optional. See *Analyzing and Cleansing Data for Sun Master Index*.
5. Extract a representative sampling of the data.
6. Configure the Initial Bulk Match and Load Tool properties.
7. Using the Bulk Matcher, perform a match analysis against the sampling, analyze the reports, and modify the match configuration if necessary. This is an iterative process for fine-tuning the matching logic to suit your data.
8. Perform match processing against the complete data set. The result of this step is a set of files containing cleansed and linked data that can be loaded directly into the database.
9. Load the data into the database using the SQL*Loader Bulk Loader or the Data Integrator Bulk Loader.
10. Open the queue to allow transactions into the master index application.

Distributed Processing

You can use multiple processors when running the IBML Tool to process data concurrently and improve performance. This is useful when processing very large data sets because the matching process can be very time and resource intensive depending on the size of the data to be processed. The number of processors required to optimize the matching process depends on the number of records you are processing and the configuration of the blocking query. If the criteria blocks defined for the query are not very selective and result in a large number of matching records, the number of records processed for each block will be larger and the number of processors should be increased. Using less selective query blocks can improve the accuracy of your matching results because it casts a wider net when retrieving possible matches to a record. Using multiple processors can help offset the cost. For smaller data sets, you can run the process on one high performance processor. In a distributed environment, one processor, known as the master IBML Tool, controls the entire process. This computer should be a very high performance machine. The remaining machines simply carry out the processes as directed by the master IBML Tool. There is no limit to the number of processors you can use. The master IBML Tool needs to be located on an FTP server in order to connect to each of the remaining IBML Tools, which use FTP to retrieve the files that they process.

About the Bulk Match Process

The Bulk Matcher performs the following sequence of tasks:

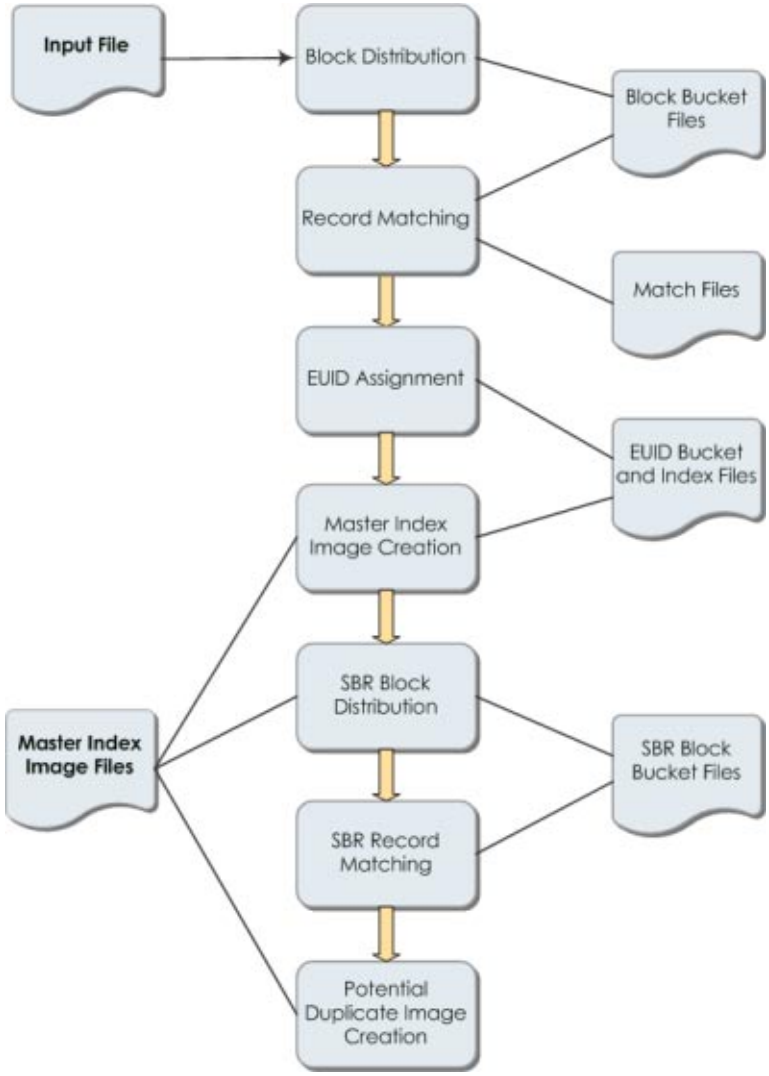


FIGURE 2 Bulk Matcher Internal Process

Block distribution and EUID assignment are both handled by the master Bulk Matcher. Matching and master index image generation are performed by all matchers. At any given time, all matchers perform the same task. When that task is complete for all matchers, they move on to the next task, obtaining the necessary files and information from the master Bulk Matcher. The cluster synchronizer (see "About the Cluster Synchronizer" on page 13) determines when a task is complete and coordinates the tasks for all matchers.

Block Distribution

The master Bulk Matcher reads the input file and then writes records to *block bucket* files to be distributed to each matcher. Before writing the data, the block distributor reads the configuration of the query, match string, and duplicate and match thresholds. It then reads in the input data and writes the data to the block files based on the defined blocking query. The number of files created is dependent on the total number records, record size, and the memory of the processor. Once the data files are created for all blocks, the cluster synchronizer indicates that the matchers can begin the match process.

Record Matching

Once the input data is distributed into individual block buckets, each matcher retrieves a bucket and proceeds to compare each record to every other record in a block and assign a weight that indicates the probability that the records match. The matching logic used here is identical to the matching logic used by the master index application. Any matches that are found during this process are written to a file. Once a matcher completes matching on a block bucket, the cluster synchronizer copies the match file to the master matcher's working directory.

EUID Assignment

When all blocks are matched, the master Bulk Matcher merges the match files from all the other matchers into one master match file. The master Bulk Matcher then assigns an EUID to the system records, assigning any records that are linked as matches the same EUID. Once system records are assigned an EUID, they are distributed to EUID files. Any system records with the same EUID are stored in the same file. Once EUID assignment is complete, the cluster synchronizer indicates that the next phase, generating the master data images, can begin.

Master Index Image Creation

The master index image generator reads from the EUID bucket files to create the master data image files to be loaded into the master index database. These images include complete enterprise records with SBRs, system records, and child objects. The SBR is determined based on the survivor calculator defined for the master index application. The image files also include assumed matches and transaction information. Each matcher processes one of the EUID buckets at a time until all buckets are processed.

Potential Duplicate Creation

The potential duplicate generator reads from the SBR bucket files created by the master index generator and generates any potential duplicate linkages. The potential duplicate generator creates block buckets from the SBR records and matches the SBRs of each record. If two records have a matching weight that is above the duplicate threshold but below the match threshold, they are added to the potential duplicate table in the master index image files. This table is loaded directly into the master index sbyn_potentialduplicates table. Once the Bulk Matcher completes this phase, match processing is complete.

About the Bulk Load Process

After the matching process is complete, you can load the data using either a SQL*Loader bulk loader or the Data Integrator Bulk Loader. Both are generated from the loader files created for the master index application. Like the Bulk Matcher, the Bulk Loader can be run on concurrent processors, each processing a different master data image file.

You can use the Data Integrator Bulk Loader to load data for an Oracle, MySQL, or SQL Server database. The SQL*Loader bulk loader can only be used for an Oracle database. With SQL*Loader, you need to drop the unique constraints and indexes using the SQL script provided before loading data, and then reinstate the constraints and indexes after loading the data.

About the Cluster Synchronizer

The cluster synchronizer coordinates the activities of all IBML processors The cluster synchronizer database, installed within the master index database, stores activity information, such as bucket file names and the state of each phase. Each IBML Tool invokes the cluster synchronizer when they need to retrieve files, before they begin an activity, and after they complete an activity. The cluster synchronizer assigns the following states to a bucket as it is processed: new, assigned, and done. The master IBML Tool is also assigned states during processing based on which of the above phases is in process.

Required Format for Flat Data Files

The default data reader for the IBML Tool is designed to read data in the format provided by the Data Cleanser. This format is based on the object definition defined for the master index application along with certain requirements for the IBML Tool. You can also extract your data to a flat file using the extractor of your choice.

If you use a data extractor other than Data Integrator, the data needs to be placed in a flat file a format the IBML Tool can read. If your data is in a different format, you can define a custom data reader to read the flat file. The IBML Tool can read a flat file in the following format without any additional configuration:

```
GID|SystemCode|LocalID|UpdateDate|UserID|ObjectFields
```

where:

- GID is a unique global ID. This is automatically generated for files created by the Data Cleanser. If you are not using the Data Cleanser, you can generate unique numbers to fill these fields.
- SystemCode is the processing code for the system from which the record originated.

- LocalID is the object's local ID in the given system.
- UpdateDate is the most recent update date for the record. This field can be empty.
- UserID is the login ID of the user who last updated the record. This field can be empty.
- ObjectFields includes the objects and fields that are defined in `object.xml`. Be sure to include every field defined in the object structure in the order they are defined (include standardized, phonetic, and parsed fields even if they are empty). Child object types are delimited by a pound sign (#) and multiple child objects of one type are delimited by a dollar sign ($).

Below is an example of a valid input record based on the standard master index Person template, which includes alias, address, and phone objects.

```
28|ORACLE|00160419|11/14/1999 08:41:10|GSMYTHE|P|ELIZABETH|ELIZABETH|E421|ANN|WARREN
|WARREN|WARAN||MRS|554-44-55555|08/18/1977|Y|F|M|W|13|BAP|ENG|STEVE|ANN|MARCH|GEORGE
|CAHILL|SHEFFIELD|CT|USA|E|Y||C4411444|CA|07/21/2018||ENG|USA#$BETH||CAHILL$LIZ|ANN
|CAHILL#$H|1519 SHORELINE DR.|1519||SHORELINE|SARALAN|Dr|Unit 5|SHEFFIELD|CT|09876
|1075|CAPE BURR|USA#$CH|9895557848|$CB|9895551500|19
```

Generating the Initial Bulk Match and Load Tool

In order to use the IBML Tool, you need to generate the tool from the master index application for which it will be used. You generate the tool from the master index project in NetBeans. The tool is generated based on the information you specified in the wizard and any changes you made to the configuration files before generating.

▼ To Generate the Initial Bulk Match and Load Tool

Before You Begin Make sure the master index application and database are created and configured. The configuration for the object structure, blocking query, match string, and matching rules should be as close to final as possible. You can use the Bulk Matcher to perform the final tuning on the blocking query and matching rules.

1. **In the NetBeans project window, right-click the main project of the master index application and then select Generate Loader Zip.**

   The file is generated and downloaded to *NetBeans_Projects**Project_Name*\`loader-generated`.

2. **On your computer, navigate to** *NetBeans_Projects**Project_Name*\`loader-generated` **and extract the contents of** `loader.zip`**.**

3. **If you are distributing the match and load processes across multiple processors, extract the contents of** `loader.zip` **to each machine processing the data.**

4. **If the IBML Tools are running on UNIX, make sure the `run-loader.sh` and `generate-sql-loader.sh` files have execute permission. Use `chmod` to grant permission if necessary.**

5. **Continue to "Configuring the Environment" on page 15.**

Configuring the Environment

Before you start working with the IBML Tool, make sure the environment is configured to handle the requirements of the load process.

▼ To Configure the Environment

1. **Complete the steps under "Generating the Initial Bulk Match and Load Tool" on page 14.**

2. **If you are using a distributed processing environment, set up an FTP server on the master processor using the FTP provider of your choice.**

3. **Install the database driver for the master index database platform on each processor running the IBML Tools.**

4. **If you are using the SQL*Loader bulk loader to load the data, install SQL*Plus and SQL*Loader on the machines performing the load.**

5. **If you are using the Data Integrator Bulk Loader instead of SQL*Loader to perform the load process, install NetBeans on the machines performing the load.**

Note – For information about using the Data Integrator Bulk Loader, see "Loading the Matched Data Into the Master Index Database" on page 33.

---

6. **Make sure the Java installation you are using matches the Java version used by the NetBeans installation that generated the Initial Bulk Match and Load Tool.**

Tip – To find the Java version for NetBeans, select Tools from the main menu and then select Java Platforms. To find the Java version you are using, type `Java -version` at a command prompt. If they do not match, either locate or install the correct version and change your PATH variable to point to the correct version.

---

7. **Continue to "Creating the Cluster Synchronizer Database" on page 16.**

Creating the Cluster Synchronizer Database

The cluster synchronizer database helps manage the various processes perform by the IBML Tool. The database is required whether you run the IBML Tool in a distributed environment or on a single processor. The database consists of only three tables, and must be installed in the master index database.

Tip – Before you begin each phase of the match and load process, make sure to truncate these tables with the script provided to avoid unique constraint errors in subsequent processes.

▼ To Create the Cluster Synchronization Tables

1. **Complete the steps under "Configuring the Environment" on page 15.**
2. **Obtain information about the master index database, such as the login ID and password of the user who created the master index database, the SID name, the port number, and the server name.**
3. **Open a SQL editor, and log in to the master index database using the above information.**
4. **Run** *NetBeans_Projects/Project_Name*`/loader-generated/cluster-synchronizer.sql` **against the database instance.**
5. **Continue to "Configuring the Initial Bulk Match and Load Tool" on page 16.**

Configuring the Initial Bulk Match and Load Tool

Before you can run the IBML Tool, you need to define certain runtime parameters, such as how to distribute the processing, FTP server properties, database properties, logging properties, and so on. You can also modify the configuration of the query used for matching, the configuration of the match string, and the weighting thresholds used by the Bulk Matcher.

Note – If you are using the Data Integrator Bulk Loader, the properties you set here only apply to the Bulk Matcher.

- The following topics provide instructions for configuring the IBML Tool.
- "Configuring the Initial Bulk Match and Load Tool Processing" on page 17
- "Configuring Initial Bulk Match and Load Tool Logging" on page 18

Configuring the Initial Bulk Match and Load Tool Processing

The bulk match process is configured by `loader-config.xml`, which is located in the `conf` subdirectory in the directory where you extracted the IBML Tool files. The process must be configured on each machine that is running a Bulk Matcher.

▼ To Configure the IBML Tool

Perform the following steps on each machine processing bulk data.

1. **Complete the steps under "Creating the Cluster Synchronizer Database" on page 16.**
2. **Navigate to the location where you extracted the IBML Tool.**
3. **Open `conf/loader-config.xml`.**
4. **Configure processing attributes by modifying the properties described in "Initial Bulk Match and Load Tool Processing Configuration" on page 20.**
5. **Configure the cluster synchronizer database connection properties by modifying the properties described in "Cluster Synchronizer Database Configuration" on page 22.**
6. **Configure the data reader and enter the name and location of the input file (see "Data Reader Configuration" on page 23 for more information).**
7. **If the IBML Tool is running on multiple processors, modify the properties described in "FTP Server Configuration" on page 21.**
8. **If you are using SQL*Loader to load the master image into the master index database, modify the properties described in "SQL*Loader Configuration" on page 22.**
9. **When using the Bulk Matcher in match analysis mode, do any of the following:**
   - **To modify the match and duplicate thresholds for match analysis, enter new values for the `duplicateThreshold` and `matchThreshold` elements.**
   - **To modify the blocking query for match analysis, modify the query builder section (described in "Initial Bulk Match and Load Tool Blocking Query Configuration" on page 19).**
   - **To modify the match string for match analysis, modify the MatchingConfig section (described in "Initial Bulk Match and Load Tool Match String Configuration" on page 20).**
10. **Save and close the file.**

11 **Repeat the above steps for each load processor in the distributed environment.**

12 **To configure logging properties, continue to "Configuring Initial Bulk Match and Load Tool Logging" on page 18; otherwise, skip to "Performing a Match Analysis" on page 24.**

Configuring Initial Bulk Match and Load Tool Logging

Logging for the IBML Tool is configured in `logger.properties`, which is located in the `conf` subdirectory in the directory where you extracted the IBML Tool files.

▼ To Configure IBML Tool Logging

1 **Complete "Configuring the Initial Bulk Match and Load Tool Processing" on page 17.**

2 **Navigate to the location where you extracted the IBML Tool.**

3 **Open `conf/logger.properties`.**

4 **Modify the properties defined in "Initial Bulk Match and Load Tool Logging Properties" on page 23.**

5 **Save and close the file.**

6 **Continue to "Performing a Match Analysis" on page 24.**

Initial Bulk Match and Load Tool Configuration Properties

Note – This topic describes the properties that are available in Java CAPS 6 Update 1.

The configuration file for the IBML Tool, `loader-config.xml`, defines several aspects of the match process, including the blocking query, match string, EUID generator, FTP server, cluster synchronizer database, and SQL*Loader properties.

The configuration file is divided into the sections listed below. In addition to these sections, you can set the match and duplicate thresholds at the beginning of the file. Use these settings to help analyze the matching logic.

Initial Bulk Match and Load Tool Field Validation Configuration

The default field validation for the master index is parsed by the standard XSD and then copied into the IBML Tool configuration file. The default field validator checks the local ID and system fields to verify that the system code is valid, the local ID format is correct, the local ID is the correct length, and neither field is null. You should not need to modify this section unless you defined custom field validations.

Initial Bulk Match and Load Tool Blocking Query Configuration

When you generate the IBML Tool, the configuration for the blocking query defined for matching in the master index application is parsed by an IBML parser and then copied into the IBML Tool configuration file.

Caution – If you defined a custom parser configuration for the blocking query, the query configuration might not be copied correctly. To ensure the correct configuration if you defined a custom parser, copy the blocking query definition directly from `query.xml` to `loader-config.xml`. You also need to create a custom block generator using the `com.sun.mdm.index.loader.blocker.BlockIdGenerator` interface, and add the name of the custom generator to the field elements in the block ID (for example, `<field>Enterprise.SystemSBR.Person.FirstName+CustomBlockIdGenerator</field>`).

This section is included in the configuration file so you can modify the blocking query during the analysis phase to help you fine–tune the query configuration for the best match results. You can quickly change the query configuration and analyze the results of the changes without needing to update the master index application and regenerate the IBML Tool each time. The query configuration is only used by the master IBML Tool, which uses the blocks defined for the query to divide the bulk data into block buckets to be distributed to the other processors that will process the data concurrently.

The blocking query might include fields that are not in your original input data, such as phonetic and normalized fields. If the input to the Bulk Matcher is the file that was generated by the Data Cleanser, the file includes all required fields, including phonetic and standardized fields. If you are using a different source for the input, the IBML Tool can standardize the data for you. For consistent matching results between the initial data set and future transactions, the query configuration used for the match and load processes should be as close as possible to that in `query.xml` in the master index Project.

Initial Bulk Match and Load Tool Match String Configuration

When you generate the IBML Tool, the configuration of the match string is copied from the master index Project to the IBML Tool configuration file. The match string is used by all IBML Tools processing the data, so this section should be identical for all IBML Tools. This section is provided so you can modify the match string during the analysis phase in order to fine–tune the match logic to achieve the best match results. As with the query configuration above, you can quickly change the match string configuration and analyze the results without needing to modify the master index application and regenerate the IBML Tool.

Ideally, the match string defined in `loader-config.xml` is as close as possible to the match string defined in `mefa.xml` in the master index Project. This assures consistent matching results between the initial data set and future transactions.

Initial Bulk Match and Load Tool Processing Configuration

The processing properties described in the following table configure how the IBML Tool processes data. In these properties, you define a name for each IBML Tool, the location of the working directories, polling properties, and so on. Some of these properties only apply to specific phases of the match and load process, and some apply to either the master or slave processors.

TABLE 1 IBML Tool Processing Properties

Property Name	Description
loaderName	A unique name for the IBML Tool residing on the current processor. This name should be unique to each IBML Tool in the distributed environment. It does not need to be modified if you are using a single processor.
isMasterLoader	An indicator of whether the IBML Tool being configured is the master IBML Tool. Specify **`true`** if it is the master or only IBML Tool; otherwise specify **`false`**.
matchAnalyzerMode	An indicator of whether to process the data in match analysis mode, which only generates analysis reports, or to perform the complete match process and generate the master index image files. Specify **`true`** to perform an analysis only; specify **`false`** to perform the actual blocking and matching process and generate the master index image files.
BulkLoad	An indicator of whether the current run will load the matched data into the database using SQL*Loader once the match process is complete. Specify **`true`** to load the data. To run a match analysis or just the matching process, specify **`false`**. If you just run the match process, you can verify the process and then load the output of the Bulk Matcher at a later time.
standardizationMode	An indicator of whether to standardize the input data. Leave the value of the this property set to **`true`**.

TABLE 1 IBML Tool Processing Properties *(Continued)*

Property Name	Description
deleteIntermediateDirs	An indicator of whether the working directories are deleted when each process is complete. Specify **`true`** to delete the directories; specify **`false`** to retain the directories.
optimizeDuplicates	An indicator of whether to automatically merge records in the input data if they have the same system and local ID. Specify **`true`** to automatically merge the duplicate records; otherwise specify **`false`**. The default is **`true`**.
rmiPort	This is not currently used.
workingDir	The absolute path to the directory in which the IBML Tools create the working files as they progress through the processing stages. The master IBML Tool also creates the master index image files here. If the path you specify does not exist, create it before running the IBML Tool.
ftp.workingDir	The absolute path to the directory on the master processor where files are placed for distribution to the remaining IBML Tools. You only need to define this property for the master IBML Tool and only if you are running multiple IBML Tools. All other tools ignore this property.
numBlockBuckets	The number of block buckets to create for the initial distribution of data blocks. Each IBML Tool works on one bucket at a time so multiple buckets are processed at once. The number of block buckets you specify depends on the number of records to process and how specific the data blocks are in the blocking query.
numThreads	The number of threads to run in parallel during processing.
numEUIDBuckets	The number of buckets the EUID assigner should place the processed records into after they have been matched and assigned an EUID.
totalNoOfRecords	The total number of records being processed. This does not need to be an exact value, but needs to be greater than or equal to the exact number of records.
pollInterval	The number of milliseconds the IBML Tools should wait before polling the master IBML Tool for their next task.
maxWaitTime	The maximum time for an IBML Tool to wait for the next task before giving up.

FTP Server Configuration

The processing properties described in the following table configure the connection information for the FTP server. They only need to be defined if the IBML Tools are run on multiple processors, and they only need to be defined for the slave processors.

TABLE 2 FTP Server Properties

Property Name	Description
ftp.server	The name of the FTP server on the master processor.

TABLE 2 FTP Server Properties *(Continued)*

Property Name	Description
ftp.username	The user ID to log in to the FTP server.
ftp.password	The password to log in to the FTP server.

Cluster Synchronizer Database Configuration

The cluster synchronizer database is used to coordinate the activities of all IBML Tools processing data. The configuration of this section must be identical for all processors.

TABLE 3 Cluster Synchronizer Database Properties

Property Name	Description
cluster.database	The database platform on which the cluster synchronizer database is installed. Possible values are **`Oracle`**, **`MSSQL`**, **`MySQL`**, or **`derby`**.
cluster.database.url	The URL for the cluster synchronizer database. The format for the URL varies by database platform. ■ For Oracle, the format is `jdbc:oracle:thin:@`*hostname*`:`*port*`:`*database_name*. ■ For SQL Server, the format is `jdbc:sqlserver://`*hostname*`:`*port*`;databaseName=`*database_name*. ■ For Derby, the format is `jdbc:derby://`*hostname*`:`*port*`/`*database_name*. ■ For MySQL, the format is `jdbc:mysql://`*server*`:`*port*`:`*database_name*.
cluster.database.user	The user ID to log in to the cluster synchronizer database.
cluster.database.password	The password to log in to the cluster synchronizer database.
cluster.database.jdbc.driver	The name of the database driver class.

SQL*Loader Configuration

The SQL*Loader properties are only used if you use SQL*Loader to load the bulk data into the master index database after it has gone through match processing and EUID assignment. If you use the Data Integrator Bulk Loader, you do not need to modify these properties. SQL*Loader can only be used with an Oracle database.

TABLE 4 SQL*Loader Property

Property Name	Description
sqlldr.userid	The connection descriptor for the master index database. For Oracle, this is in the format *user*`/`*password*`@`*service_name*; for example, `midm/midm@MIDBS.sun.com`.

Data Reader Configuration

This section defines the data reader to use, the location of the input file, and any parameters. You can define a custom data reader to read data into the Bulk Matcher if you are not using the data file output by the Data Cleanser or if your data is formatted differently than the default data reader requires. For more information about the default data reader's requirements, see "Required Format for Flat Data Files" on page 13. You define the custom reader in Java, and then specify the class and configure the reader using the elements and attributes in the following table. You only need to configure the data reader for the master IBML Tool.

TABLE 5 Custom Data Reader Configuration Elements and Attributes

Element	Attribute	Description
bean		A definition for one data reader. This element includes the following elements and attributes.
	id	A unique name for the data reader.
	class	The Java class to implement for the data reader. The default data reader is `com.sun.mdm.index.dataobject.DataObjectFileReader`. This reader can access flat file data in the format described in "Required Format for Flat Data Files" on page 13.
	singleton	An indicator of whether to use the singleton pattern for the data reader class. Specify **`true`** to use a singleton pattern; otherwise specify **`false`**.
constructor-arg		A parameter for the data reader. The default reader accepts two parameters. The first parameter has a type of `java.lang.String` and specifies the location of the input file. The second parameter has a type of boolean and indicates whether the input file contains delimiters. Specify **`true`** if the file contains delimiters.
	type	The data type of the parameter value.
	value	The value for the parameter.

Initial Bulk Match and Load Tool Logging Properties

The configuration file for the logging properties, `logging.properties`, defines how much information is logged while the IBML Tools are running. By default, logging is set to display INFO level messages and above on both the console and in a log file.

The following table lists and describes the default properties for the configuration file, but you can add new properties based on the log handlers you use. For more information about log handlers and the properties for each, see the Javadocs for the `java.util.logging` package.

TABLE 6 IBML Tool Logging Properties

Property Name	Description
handlers	A list of log handler classes to use, such as `java.util.logging.FileHandler` and `java.util.logging.ConsoleHandler`. Each handler you define needs to be configured according to the properties defined for the class.
level	The logging level to use. By default, this is set to INFO level logging.
java.util.logging.FileHandler.pattern	The name and location of the log files that are generated. By default, the files are located in the directory where you extracted the IBML Tools in the logs subdirectory. The files are named `loader#.log`, where # is an integer that is incremented each time a new log file is created. The log file with the lowest number is the most recent.
java.util.logging.FileHandler.limit	The maximum number of bytes to write to any one log file.
java.util.logging.FileHandler.count	The number of output files to cycle through.
java.util.logging.FileHandler.formatter	The name of the Java class used to format the log file. The IBML Tool provides a formatting class, `com.sun.mdm.index.loader.log.LogFormatter`, but you can define your own.
java.util.logging.ConsoleHandler.level	The level at which information is logged on the console.
java.util.logging.ConsoleHandler.formatter	The name of the Java class used to format the log information on the console. By default, the IBML Tool uses `java.util.logging.SimpleFormatter`, but you can define your own.

Performing a Match Analysis

Before you perform the actual data matching, you can perform match analyses on a subset of the data to be loaded to determine whether the various components of the match process are configured correctly. This analysis can show whether the data blocks defined for the blocking query are returning too many or too few records and whether certain fields in the match string are inaccurately skewing the composite match weight. You can also use this analysis to determine whether the duplicate and match threshold are correct.

This is an iterative process, and you might need to run through the analysis several times before you are satisfied that the match and query configuration is optimized for your data set.

Perform the following steps to analyze the data for matching:

Running the Bulk Matcher in Analysis Mode

Note – This procedure includes steps that were updated for Java CAPS Release 6 Update 1. The variable JDBC_JAR_PATH was previously ORACLE_JDBC_JAR, and wasn't present in all files.

When you run the Bulk Matcher in analysis mode, use a representative sample of the actual data you are loading into the master index database. You do not need to run the entire set of input records through the analysis.

Caution – If you are *rerunning* the Bulk Matcher in analysis mode, make sure to truncate the cluster synchronizer database tables first. Otherwise, unique constraint errors occur and the run fails. To truncate the tables, run `cluster-truncate.sql` against the cluster synchronizer database.

▼ To Run the Bulk Matcher in Analysis Mode

1. **Complete the steps under "Configuring the Initial Bulk Match and Load Tool" on page 16.**
2. **For each IBML Tool, open `loader-config.xml` (located in the IBML Tool home directory in the `conf` subdirectory).**
3. **Set the `matchAnalyzerMode` property to `true`, and verify the remaining property settings.**
4. **Save and close the file.**
5. **To configure and run the match analysis, do one of the following.**
   - **If the master loader is running on Windows:**
     a. **Navigate to the master IBML Tool home directory and open `run-loader.bat` for editing.**
     b. **Change the value of the JDBC_JAR_PATH variable in the first line to the location and name of the database driver for the master index database platform; for example, `set JDBC_JAR_PATH=C:\oracle\jdbc\lib\ojdbc14.jar`.**

c. **Close and save the file.**

d. **Double-click** `run-loader.bat` **or type** `run-loader` **from a command line.**

- **If the master loader is running on UNIX:**

a. **Navigate to the master IBM Tool home directory and open** `run-loader.sh` **for editing.**

b. **Change the value of the JDBC_JAR_PATH variable in the first line to the location and name of the database driver for the master index database platform; for example,** `export JDBC_JAR_PATH=${oracle_home}/jdbc/lib/ojdbc14.jar`.

c. **Close and save the file.**

d. **Type** `sh run-loader.sh` **at the command line.**

6 **Examine the log files to be sure no errors occurred during the analysis.**

7 **Continue to** **"Reviewing the Match Analysis Results" on page 26****.**

Reviewing the Match Analysis Results

The output of the Bulk Matcher when run in analysis mode is a PDF file with a list of records that were automatically matched to each other (assumed matches) from the data set you analyzed. The report displays the matching weight given to each field, so you can analyze the value and accuracy of each field for matching as well as the agreement and disagreement weights (or u-probabilities and m-probabilities) defined in the matching configuration file of the master index application.

The following figure shows two entries from the match analysis report. The name of each match field is listed in the left column, the values for those fields in the two assumed match records are listed in the next two columns, and the composite match weight and the weight for each field are listed in the final column.

systemcode	ORACLE	ORACLE	38.75
localid	55469812	84213574	38.75
Person.FirstName_Std	JOSEPH	JOSEPH	10.0
Person.LastName_Std	WARDELL	WARDELL	10.0
Person.SSN	888447477	888447477	5.0
Person.Gender	M	M	5.0
Person.MotherMN_Std	FLEMING	null	0.0
Person.DOB	12/07/1958	12/06/1958	8.75
Person.Address[*].AddressLine1_StName	OCEAN	null	0.0
Person.Address[*].AddressLine1_HouseNo	1704	null	0.0

systemcode	ORACLE	SAP	38.72472226619
localid	213006500	123456789	38.72472226619
Person.FirstName_Std	ELIZABETH	ELIZABETH	10.0
Person.LastName_Std	WARREN	WARREN	10.0
Person.SSN	null	123124242	0.0
Person.Gender	F	F	5.0
Person.MotherMN_Std	null	null	0.0
Person.DOB	05/14/1964	05/14/1964	10.0
Person.Address[*].AddressLine1_StName	SHORELINE	WAYFIELD	1.222222208976
Person.Address[*].AddressLine1_HouseNo	12500	1400	2.502500057220

FIGURE 3 Match Analysis Report Excerpt

After you perform the steps under "Running the Bulk Matcher in Analysis Mode" on page 25, complete the analysis by using the information in the match analysis report to do the following:

- Look for records that are assumed matches but should not be. This might indicate that the match threshold is set too low for the number of match fields or that one or more fields are given too much weighting relevance.
- Verify that fields that uniquely identify records, such as a social security number, are given a higher weight when they match.
- Verify that null fields are being handled correctly.
- If the assumed match records with the lowest composite match weights are definite matches of one another, the match threshold might be set too high. You might want to experiment with setting the match threshold lower and running another match analysis.

After you complete your analysis, you can reconfigure the matching logic as described in"Reviewing the Match Analysis Results" on page 26 and then rerun the analysis. If your analysis shows that the matching configuration is correct and does not require any more changes, continue to "Performing the Bulk Match" on page 28. If the matching configuration is correct, make sure to update the master index application to match the new configuration.

Reconfiguring the Matching Logic

If the results of the match analysis show that you need to modify the query, thresholds, or match string, you can make the changes to the IBML Tool configuration file and run the Bulk Matcher again to analyze the new settings. Once you are satisfied with the new settings, you need to update the master index application configuration accordingly.

▼ To Reconfigure the Matching Logic

1. **Complete the match analysis, as describe under "Reviewing the Match Analysis Results" on page 26.**
2. **In the directory where the IBML Tool is located, open `conf/loader-config.xml`.**
3. **To modify the match and duplicate thresholds for match analysis, enter new values for the `duplicateThreshold` and `matchThreshold` elements.**
4. **To modify the blocking query for match analysis, modify the query builder section (described in "Initial Bulk Match and Load Tool Blocking Query Configuration" on page 19).**
5. **To modify the match string for match analysis, modify the MatchingConfig section (described in "Initial Bulk Match and Load Tool Match String Configuration" on page 20).**
6. **Run the match analysis again, as described in "Running the Bulk Matcher in Analysis Mode" on page 25.**
7. **After you run the analysis for the final time, continue to "Performing the Bulk Match" on page 28.**

Caution – When you complete the analysis and have made the final modifications to the blocking query, matching string, and match thresholds, be sure to modify the master index application so the processing is identical. The match string is defined in `mefa.xml`, the thresholds are defined in `master.xml`, and the blocking query is defined in `query.xml`. You can copy the configuration from `loader-config.xml` directly into these files.

Performing the Bulk Match

Note – This procedure includes steps that were updated for Java CAPS Release 6 Update 1. The variable JDBC_JAR_PATH was previously ORACLE_JDBC_JAR, and wasn't present in all files.

After you perform the match analysis and are satisfied with the matching logic configuration, you are ready to match the full set of data to be loaded into the master index database.

If you are using SQL*Loader to load the matched data, you can run the Bulk Matcher and the Bulk Loader all in the same step. For instructions on how to do this, see "Running the Bulk Match and Bulk Load in One Step (SQL*Loader Only)" on page 31.

▼ To Perform the Bulk Match

1. **Complete the steps under "Performing a Match Analysis" on page 24.**

2. **From the master IBML Tool home directory, run `cluster-truncate.sql` against the cluster synchronizer database.**

3. **For each IBML Tool, open `loader-config.xml` (located in the IBML Tool home directory in the `conf` subdirectory).**

4. **Set the `matchAnalyzerMode` property to `false`.**

5. **Verify that the rest of the properties are configured correctly, and then save and close the file.**

   For information about the configurable properties, see "Initial Bulk Match and Load Tool Configuration Properties" on page 18.

6. **To configure and run the match process, do one of the following.**

   - **If the master loader is running on Windows:**

     a. **Navigate to the master IBML Tool home directory and open `run-loader.bat` for editing.**

     b. **Change the value of the JDBC_JAR_PATH variable in the first line to the location and name of the database driver for the master index database platform; for example, `set JDBC_JAR_PATH=C:\oracle\jdbc\lib\ojdbc14.jar`.**

     c. **Close and save the file.**

     d. **Double-click `run-loader.bat` or type `run-loader` from a command line.**

   - **If the master loader is running on UNIX:**

     a. **Navigate to the master IBML Tool home directory and open `run-loader.sh` for editing.**

     b. **Change the value of the JDBC_JAR_PATH variable in the first line to the location and name of the database driver for the master index database platform; for example, `export JDBC_JAR_PATH=${oracle_home}/jdbc/lib/ojdbc14.jar`.**

     c. **Close and save the file.**

     d. **Type `sh run-loader.sh` at the command line.**

7 **Examine the log files in the `logs` directory of the IBML Tool home directory to verify that no errors occurred during the match process.**

8 **Examine the files located in the `masterindex` folder of your working directory to be sure all tables were populated.**

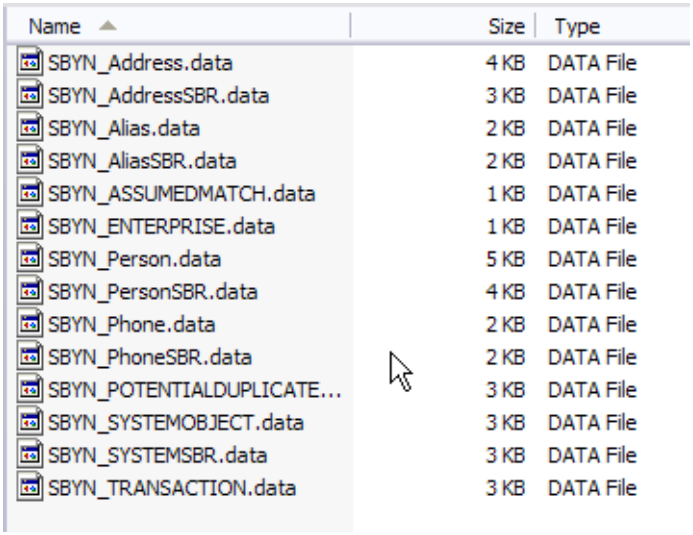


FIGURE 4 Master Index Image Files

9 **If you configured the IBML Tool to keep the temporary folders in the working directory, you can examine the blocking, EUID, and match files for additional verification.**

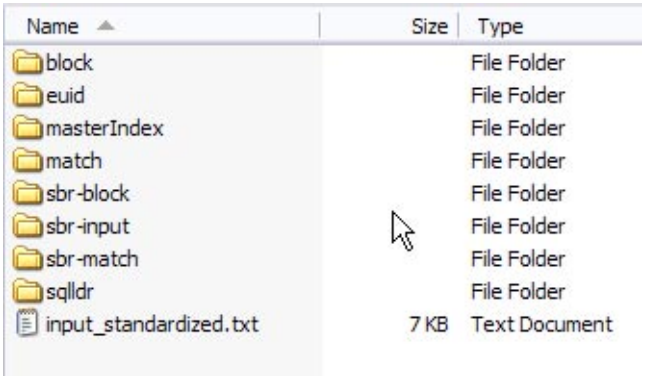


FIGURE 5 IBML Tool Working Directory

10 **To load the data into the master index database, follow one of the procedures under "Loading the Matched Data Into the Master Index Database" on page 33.**

Running the Bulk Match and Bulk Load in One Step (SQL*Loader Only)

Note – This procedure includes steps that were updated for Java CAPS Release 6 Update 1. The variable JDBC_JAR_PATH was previously ORACLE_JDBC_JAR, and wasn't present in all files.

After you perform the match analysis and are satisfied with the matching logic configuration, you can either run the match process alone, or you can run the match and load processes together if you are using SQL*Loader to load the data. SQL*Loader can only be used on an Oracle master index database. To run the match process separately, see "Performing the Bulk Match" on page 28.

▼ To Run the Bulk Match and Bulk Load in One Step

1 **Complete the steps under "Performing a Match Analysis" on page 24.**

2 **From the master IBML Tool home directory, run `cluster-truncate.sql` against the cluster synchronizer database.**

3 **For each IBML Tool, open `loader-config.xml` (located in the IBML Tool home directory in the `conf` subdirectory).**

4 **Set the `matchAnalyzerMode` property to `false` and the `BulkLoad` property to `true`.**

5 **Define the SQL*Loader property described in "SQL*Loader Configuration" on page 22.**

6 **Verify that the rest of the properties are configured correctly, and then save and close the file.**

7 **To generate the loader, do one of the following.**

- **If the master loader is running on Windows:**

  a. **Navigate to the master IBML Tool home directory and open `generate-sql-loader.bat` for editing.**

b. **Change the value of the JDBC_JAR_PATH variable in the first line to the location and name of the database driver for the master index database platform; for example,** `set JDBC_JAR_PATH=C:\oracle\jdbc\lib\ojdbc14.jar`.

c. **Close and save the file.**

d. **Double-click** `generate-sql-loader.bat` **or type** `generate-sql-loader` **from a command line.**

- **If the master loader is running on UNIX:**

a. **Navigate to the master IBML Tool home directory and open** `sh generate-sql-loader.sh` **for editing.**

b. **Change the value of the JDBC_JAR_PATH variable in the first line to the location and name of the database driver for the master index database platform; for example,** `export JDBC_JAR_PATH=${oracle_home}/jdbc/lib/ojdbc14.jar`.

c. **Close and save the file.**

d. **Type** `sh generate-sql-loader.sh` **at the command line.**

A new directory named `sqlldr` is created in the working directory.

8 **For UNIX only, modify the permissions for the shell scripts in the new** `sqlldr` **directory by running the following command:**

```
chmod u+x *.sh
```

9 **In the master IBML Tool home directory, run** `cluster-truncate.sql` **against the master index database to clear the cluster synchronizer tables.**

10 **In the** `sqlldr` **folder in the working directory, run** `drop.sql` **against the master index database to drop constraints and indexes.**

11 **To configure and run the load process, do one of the following.**

- **If the master loader is running on Windows:**

a. **Navigate to the master IBML Tool home directory and open** `run-loader.bat` **for editing.**

b. **Change the value of the JDBC_JAR_PATH variable in the first line to the location and name of the database driver for the master index database platform; for example,** `set JDBC_JAR_PATH=C:\oracle\jdbc\lib\ojdbc14.jar`.

c. **Close and save the file.**

d. **Double-click** `run-loader.bat` **or type** `run-loader` **from a command line.**

- **If the master loader is running on UNIX:**

  a. **Navigate to the master IBML Tool home directory and open** `run-loader.sh` **for editing.**

  b. **Change the value of the JDBC_JAR_PATH variable in the first line to the location and name of the database driver for the master index database platform; for example,** `export JDBC_JAR_PATH=${oracle_home}/jdbc/lib/ojdbc14.jar`**.**

  c. **Close and save the file.**

  d. **Type** `sh run-loader.sh` **at the command line.**

12 **Examine the log files to be sure no errors occurred during the match process, and check the files located in the** `masterIndex` **folder of your working directory to be sure all tables were populated.**

Tip – If for any reason you need to stop and restart this process, run `truncate.sql` in the `sqlldr` directory against the master index database before restarting the process.

---

13 **From the** `sqlldr` **directory, run** `create.sql` **against the master index database to reinstate the dropped indexes and constraints.**

Loading the Matched Data Into the Master Index Database

The IBML Tool provides three methods to load the master data images generated by the Bulk Matcher. A Data Integrator command line tool is provided to generate and then run the ETL collaborations that load the data. Data Integrator also provides a wizard that guides you through the creation of the ETL collaborations. You can then generate a command-line program to run the collaborations. Finally, you can use SQL*Loader to load the data if the master index database is running on Oracle.

Perform one of the following procedures to load the matched data into your master index database:

- "Loading Matched Data Using SQL*Loader" on page 34
- "Loading Matched Data Using the Data Integrator Wizard Bulk Loader" on page 36
- "Loading Matched Data Using the Data Integrator Command-Line Bulk Loader" on page 43

Loading Matched Data Using SQL*Loader

Note – This procedure includes steps that were updated for Java CAPS Release 6 Update 1. The variable JDBC_JAR_PATH was previously ORACLE_JDBC_JAR, and wasn't present in all files.

If the master index database runs on an Oracle platform, you can use either SQL*Loader or the Data Integrator Bulk Loader to load the matched data into the database. SQL*Loader cannot be used for a SQL Server or MySQL database.

▼ To Load Matched Data Using SQL*Loader

1. **Complete the steps under "Performing the Bulk Match" on page 28.**
2. **From the master IBML Tool home directory, run `cluster-truncate.sql` against the cluster synchronizer database.**
3. **For each IBML Tool, open `loader-config.xml` (located in the IBML Tool home directory in the `conf` subdirectory).**
   a. **Define the SQL*Loader property as described in "SQL*Loader Configuration" on page 22.**
   b. **Change the value of the `BulkLoad` property to true.**
   c. **Save and close the file.**
4. **To generate the loader, do one of the following.**
   - **If the master loader is running on Windows:**
     a. **Navigate to the master IBML Tool home directory and open `generate-sql-loader.bat` for editing.**
     b. **Change the value of the JDBC_JAR_PATH variable in the first line to the location and name of the database driver for the master index database platform; for example, `set JDBC_JAR_PATH=C:\oracle\jdbc\lib\ojdbc14.jar`.**
     c. **Close and save the file.**
     d. **Double-click `generate-sql-loader.bat` or type `generate-sql-loader` from a command line.**

- **If the master loader is running on UNIX:**

  a. **Navigate to the master IBML Tool home directory and open** `sh generate-sql-loader.sh` **for editing.**

  b. **Change the value of the JDBC_JAR_PATH variable in the first line to the location and name of the database driver for the master index database platform; for example,** `export JDBC_JAR_PATH=${oracle_home}/jdbc/lib/ojdbc14.jar`**.**

  c. **Close and save the file.**

  d. **Type** `sh generate-sql-loader.sh` **at the command line.**

  A new directory named `sqlldr` is created in the working directory.

5 **In the master IBML Tool home directory, run** `cluster-truncate.sql` **against the master index database to clear the cluster synchronizer tables.**

6 **In the** `sqlldr` **folder in the working directory, run** `drop.sql` **against the master index database to drop constraints and indexes.**

7 **In the** `sqlldr` **directory, do one of the following:**

- **On Windows, double-click** `bulk-loader.bat` **or type** `bulk-loader.bat` **from a command line.**
- **On UNIX, type** `sh bulk-loader.sh` **at the command line.**

8 **After the data is loaded, close any command prompts that were left open by the process and examine the SQL*Loader log files located in the** `sqlldr/log` **directory to ensure there were no errors during processing.**

Note – Any records that contained bad data and were not inserted into the master index database are written to the `sqlldr/bad` directory. Any records that contained bad data and were discarded are written to the `sqlldr/discard` directory.

---

9 **In the** `sqlldr` **directory, run** `create.sql` **against the master index database to reinstate the dropped indexes and constraints.**

Loading Matched Data Using the Data Integrator Wizard Bulk Loader

Note – This process is new for Java CAPS 6 Update 1 and is unavailable in Release 6.

You can use the Data Integrator Wizard to generate the Bulk Loader for a master index application. The wizard generates the collaborations that define the load process. Once you build the Data Integrator project, you can generate a command-line tool that runs the collaborations in the correct order.

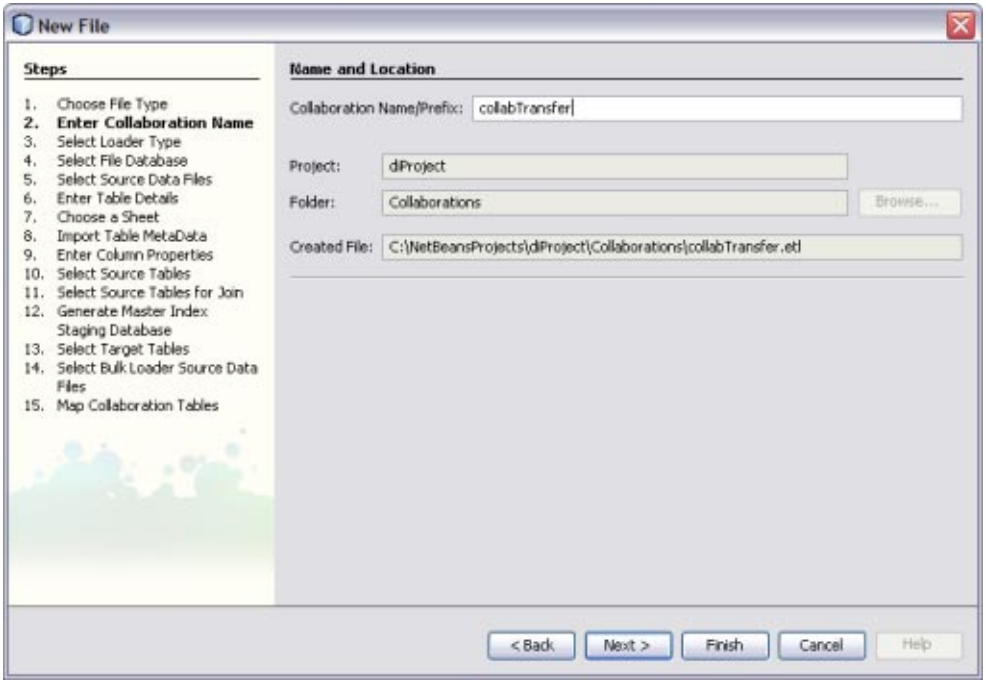
▼ To Load Matched Data Using the Data Integrator Wizard Bulk Loader

Before You Begin

- Make sure the master index database is running, and that your NetBeans IDE is connected to the master index database. To connect to the database, right-click Drivers under Databases on the Services window, select Connect Using, and enter the database connection information. One the connection is created, right-click the new database entry under Databases and select Connect.
- Start the application server.
- Start the ETL Service Engine (located on the Services window of Net Beans under the application server > JBI > Service Engines).

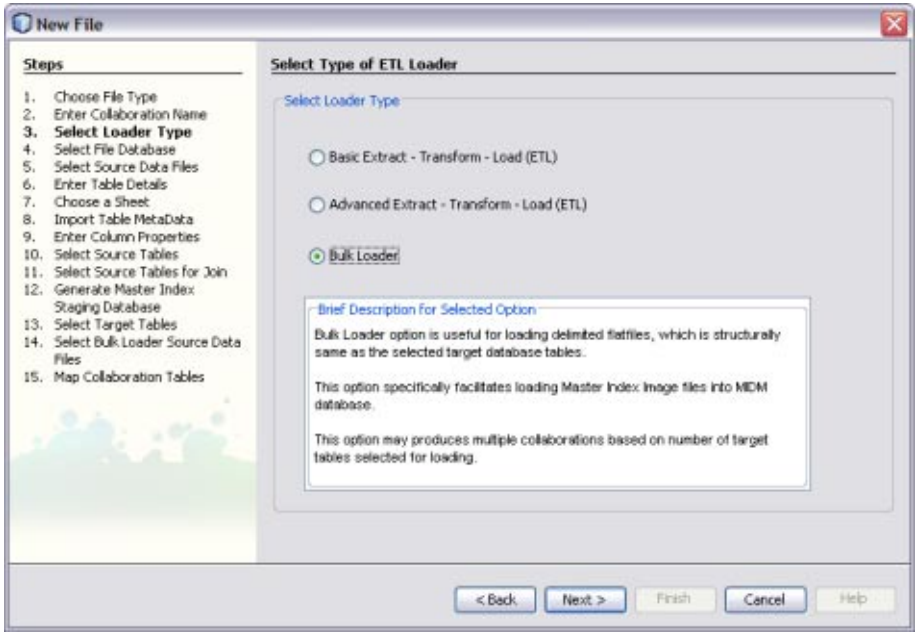
1. **On the NetBeans Projects window, expand the new Data Integrator project and right-click Collaborations.**
2. **Point to New, and then select ETL.**

   The Data Integrator Wizard appears with the Name and Location window displayed.
3. **Enter name for the collaboration.**



4 **Click Next.**

5 **On the Select Type of ETL Loader window, select Bulk Loader.**



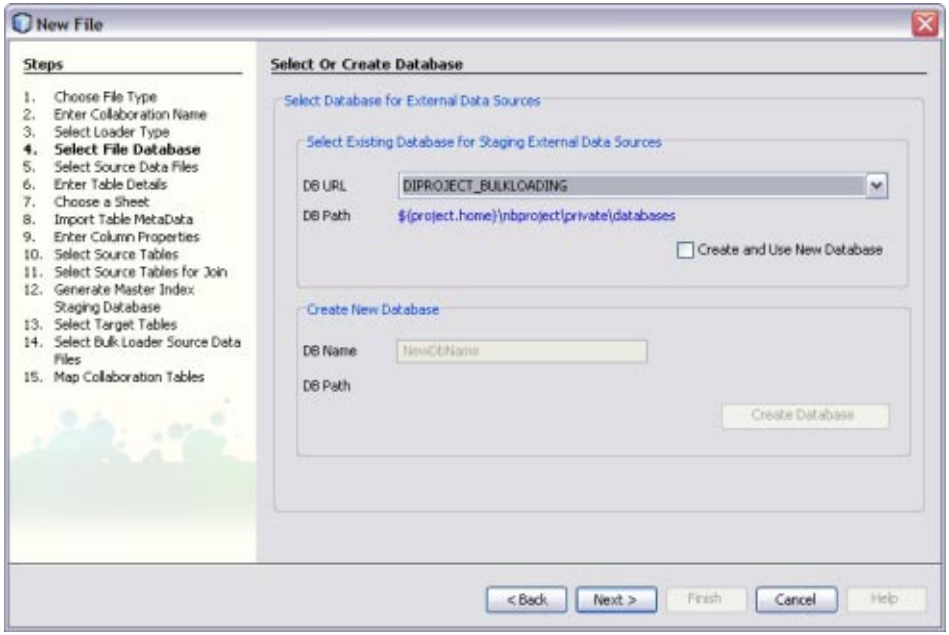
6 Click Next.

The Select or Create Database window appears.

7 To specify a staging database to use for external data sources (for this project only), do one of the following:

a. Select an existing database to use from the DB URL field.

b. Select Create and Use New Database, enter a name for a new database in the DB Name field, and then click Create Database. Select the new database in the DB URL field.



Note – This database is required and is used for internal processing only.

8 Click Next.

The Select JDBC Target Tables window appears.

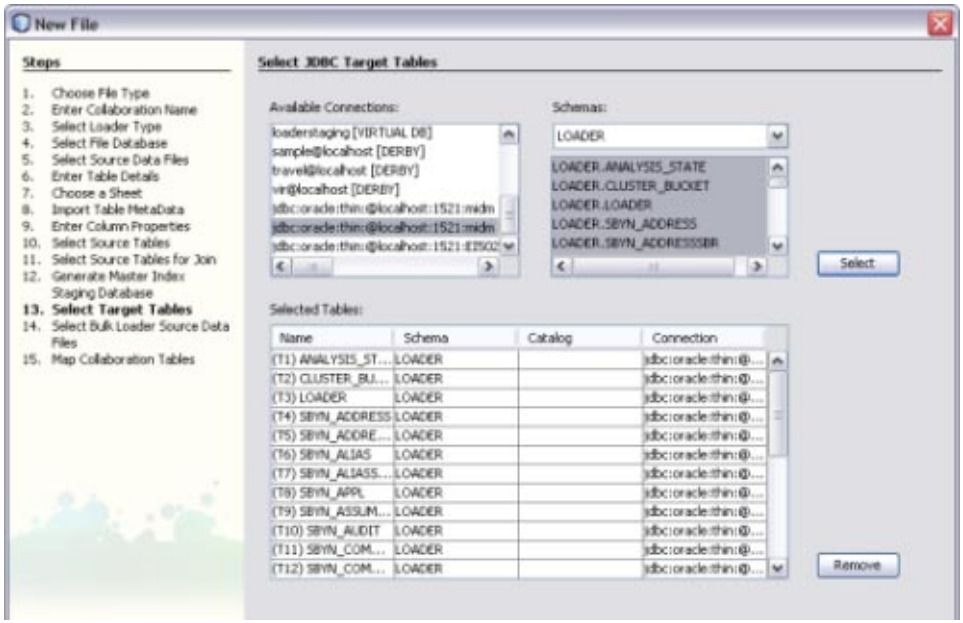
9 To choose the target tables to load the extracted data into, do the following:

a. Under Available Connections, select the master index database.

b. Under Schemas, select the schema that contains the tables to load the data into.

c. Under Schema, select only the tables that correspond to the data files produced by the Bulk Matcher, and then click Select.

Tip – You can use the Shift and Control keys to select multiple tables at once. If you select target tables that do not correspond to the Bulk Matcher files, collaborations without source table are generated and the project fails to build.



d. Click Next.

The Choose Bulk Loader Data Source window appears.

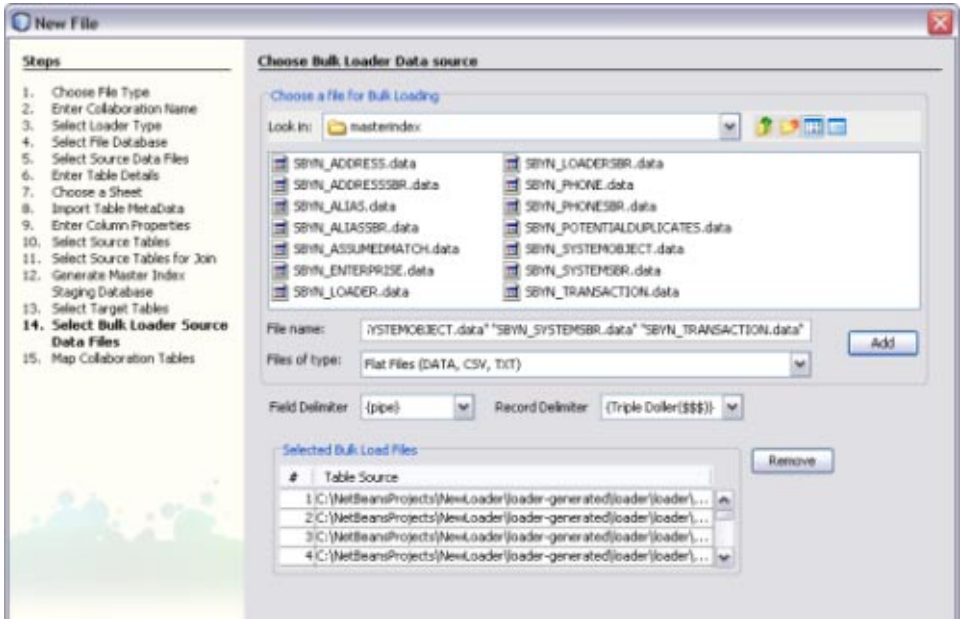
10 To specify the source data for the Bulk Loader, do the following:

a. In the upper portion of the window, browse to the location of the of the output files from the Bulk Matcher.

Note – These files are located in *NetBeansProjects_Home*/*Project_Name*/`loader-generated/loader/`*work*`/masterindex`, where *work* is the location you specified for the working directory in `loader-config.xml`.

---

b. Select all of the data files in the `masterindex` directory, and then click Add.



c. Click Next.

The Map Selected Collaboration Tables window appears.

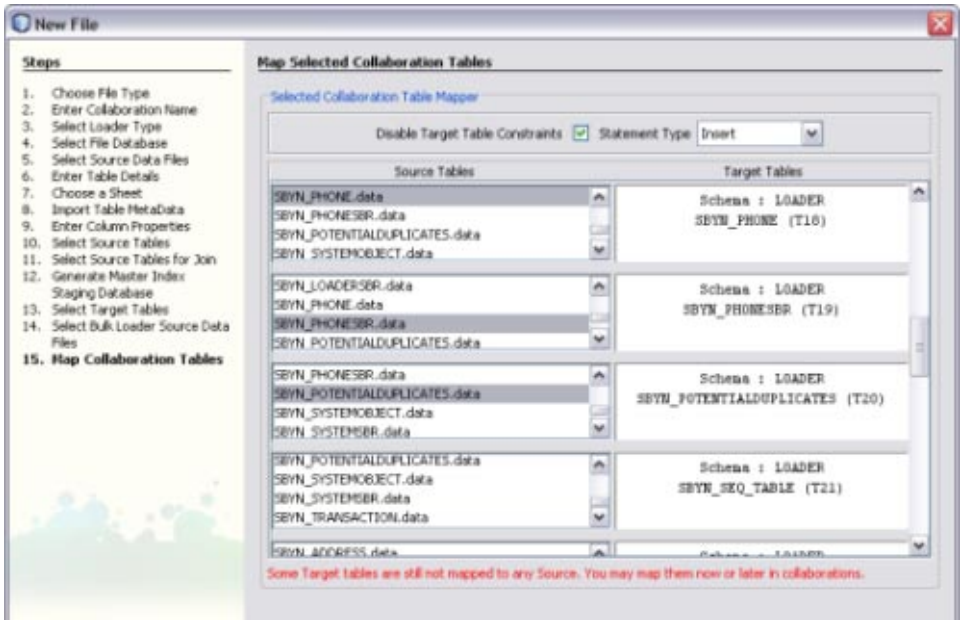
11 To map source and target data, do the following:

a. To disable constraints on the target tables, select Disable Target Table Constraints.

b. Select the SQL statement type to use for the transfer. You can select insert, update, or both.

c. The wizard automatically maps the source and target tables for you. Review the mapping to verify its accuracy.

Note – Not every table on the left will be mapped. For example, system tables such as SBYN_COMMON_HEADER, SBYN_COMMON_DETAIL, SBYN_APPL, and SBYN_SYSTEMS do not need to be mapped.



d. Click Finish.

An ETL collaboration is created for each target table. This might take a few minutes to generate.

12 Verify that all the required collaborations were created, and then right-click the Data Integrator project and select Clean and Build.

13 Right-click the Data Integrator project again, and select Generate Command Line ETL.

14 To configure the load process, do the following:

a. On the Files window, expand the Data Integrator project and then expand ETLLoader.

b. Open `startLoad.bat` or `startLoad.sh` for editing.

c. In the JAVA_HOME variable, enter the path to your JRE installation; for example, `C:\Java\jre1.5.0_12`.

d. In the DATABASE_DRIVERS variable, enter the paths to the master index database driver and the Axion database driver.

Tip – The master index driver should be located in *app_server*`/lib` and the Axion database driver can be found in `ETLLoader/lib`.

e. **Save and close the file.**

15 **On your computer, navigate to your NetBeans projects home directory and then to** *bulk_loader_project*`/ETLLoader`**.**

16 **To perform the load, do one of the following:**

- **For Windows, double-click** `startLoad.bat` **or type** `startLoad.bat` **from a command line.**
- **For UNIX, type** `sh startLoad.sh` **at the command prompt.**

17 **If your master index application is deployed, you can use the Master Index Data Manager to verify that the data was loaded successfully. Also review the log files under** `ETLLoader/logs`**.**

Loading Matched Data Using the Data Integrator Command-Line Bulk Loader

You can use the Data Integrator command-line Bulk Loader to load data into an Oracle, MySQL, or SQL Server database. Using the command-line tool does not require the use of NetBeans, but it does require that NetBeans be installed on the master loader machine.

▼ To Load Matched Data Using the Data Integrator Bulk Loader

1 **Complete the steps under "Performing the Bulk Match" on page 28.**

2 **In the master IBML Tool home directory, open** `genCollab.bat` **(or** `genCollab.sh` **for UNIX) and configure the properties described in "Data Integrator Bulk Loader Properties" on page 44.**

3 **Save and close the file.**

4 **In the master IBML Tool home directory, do one of the following:**

- **On Windows, double-click** `genCollab.bat` **or type** `genCollab.bat` **from a command line.**
- **On UNIX, type** `sh genCollab.sh` **at the command line.**
  This generates a zip file in the IBML Tool home directory.

5 **Extract the contents of `etl-loader.zip` to the current directory.**

This generates an ETL collaboration and creates a new directory, `ETLloader`, in the IBML Tool home directory.

6 **In the master IBML Tool home directory, run `cluster-truncate.sql` against the master index database to clear the cluster synchronizer tables.**

7 **In the `ETLloader/config` directory, open `logger.properties` and modify any logging properties if needed.**

8 **In the `ETLloader` directory, do one of the following:**

- **On Windows, double-click `startLoad.bat` or type `startLoad.bat` from a command line.**
- **On UNIX, type `sh startLoad.sh` at the command line.**

9 **After the data is loaded, check the log files to ensure there were no errors during processing.**

Data Integrator Bulk Loader Properties

The Data Integrator collaboration is generated by a file that includes configurable properties you need to define. The file is named `genCollab.bat` for Windows and `genCollab.sh` for UNIX. It is located in the directory where you extracted the IBML Tool files on the master processor. The following table lists and describes the default properties for the file.

Tip – If you get a usage error when running the Bulk Loader after configuring the properties below, remove the double-quotes from around the paths and filenames (but not from the delimiters).

TABLE 7 Data Integrator Bulk Loader Properties

Property Name	Description
NetBeans and Java Properties	
NB_HOME	The absolute path to the NetBeans home directory on the master processor.
JAVAPATH	The absolute path to the `bin` directory in the Java installation; for example, `C:\\Java\jre1.5.0_11\bin`.
DB_DRIVER_PATH	The absolute path to the database driver for the database platform of the master index database.

TABLE 7 Data Integrator Bulk Loader Properties *(Continued)*

Property Name	Description
DB_DRIVER_NAME	The name of the database driver in the path specified above; for example, `ojdbc14.jar`.
Source Data Properties	
SOURCE_LOC	The absolute path to the data files to be loaded into the master index database. These are located in the `masterindex` folder in the working directory you created for the Bulk Matcher.
FIELD_DELIMITER	The character that separates the fields in the master data image files. By default, fields are delimited by a pipe character (|).
RECORD_DELIMITER	The characters that separate the records in the master data image files. By default, the records are delimited by three dollar signs ($$$).
Target Database Properties	
TARGET_DB_TYPE	The database platform used for the master index database. Specify 1 for Oracle, 2 for MySQL, or 3 for SQL Server.
TARGET_LOC	The name or IP address of the server on which the master index database resides.
TARGET_PORT	The port number on which the master index database is listening. The default port is 1521 for Oracle, 1433 for SQL Server, and 3306 for MySQL.
TARGET_ID	The SID or database name of the master index database.
TARGET_SCHEMA	The name of the database schema that defines the tables, fields, and relationships for the master index database. The default schema for SQL Server databases is “dbo”; for Oracle, the default is the same as the SID name of the database.
TARGET_CATALOG	The name of the database catalog containing the master index database metadata. This property can be left empty.
TARGET_LOGIN	The login ID of the user with administrator abilities for the master index database.
TARGET_PW	The password for the above login ID.