Oracle® Java CAPS Master Data Management Suite Primer

Part No: 821–2653
March 2011

ORACLE

120126@25097

Contents

Oracle Java CAPS Master Data Management Suite Primer

The topics listed here provide information about the complete Java CAPS MDM Suite and its individual components.

About Master Data Management

In today's business environment, it is becoming increasingly difficult to access current, accurate, and complete information about the people or entities for which information is stored across an organization. As organizations merge and grow, information about the same entity is dispersed across multiple disparate systems and databases, and there might be several different versions of the information of varying quality. Information becomes fragmented, duplicated, unreliable, and hard to locate. A single source of authoritative, reliable, and sustainable data is needed. As soon as data about the same entities begins to be stored in multiple departments, locations, and applications, the need for this single source becomes apparent.

Master Data Management (MDM) creates and maintains a source of enterprise data that identifies and stores the single best information about each entity across an organization in a secure environment. MDM is the framework of processes and technologies used to cleanse records of inconsistent data, analyze the state of the data, remove data duplication, call into question potential duplication, and maintain a system of continuous cleansing. Implementing an MDM initiative produces a complete and consolidated view of the entities about which information is stored, such as customers, patients, vendors, inventory, and so on. The MDM solution produces a single best view of the data. The single best view is referred to as reference data.

Core features of an MDM solution include data profiling, stewardship, standardization, matching, and deduplication. This combination cleanses data from the very beginning, identifying and rectifying data anomalies from the start and providing a system of continuous cleansing as new data is added.

About the Oracle Java CAPS Master Data Management Suite

The Oracle Java CAPS Master Data Management Suite(MDM Suite) is a unified product offering that provides a comprehensive MDM solution. The MDM Suite creates an integrated and consistent view of master data. It addresses the full MDM lifecycle, from extracting, cleansing, and matching data in source systems to loading the modified data into the master index database, and finally to managing and maintaining the reference data. The final step includes deduplication, merging, unmerging, auditing, and so on.

The Java CAPS MDM Suite is based on the solid foundation of the Oracle Java Composite Application Platform Suite (Java CAPS), which provides an integration platform for building and managing composite applications based on a service oriented architecture (SOA). The Java CAPS MDM Suite includes products for creating a cross-reference of records stored throughout an organization and for extracting, transforming, and loading bulk data. It also includes data quality tools, a portal and presentation layer, and a security and identity management framework. Java CAPS MDM Suite operations can be exposed as web services for complete integration with external systems.

The Java CAPS MDM Suite leverages existing applications and systems and consolidates existing information to provide a single best view of the information and improve the quality, accuracy, and availability of data across an organization. The single best view is the source of consistent, reliable, accurate, and complete data for the entire organization and, in some cases, business partners. The suite is able to create the singe best view by using advanced standardization and matching methodologies along with configurable business logic to uniquely identify common records and determine whether two records represent the same entity.

The following topics provide additional information about the Java CAPS MDM Suite.

- "Java CAPS MDM Suite Features" on page 6
- "Java CAPS MDM Suite Architecture" on page 7
- "Java CAPS Master Data Management Process" on page 12
- "About the Standardization and Matching Process" on page 15

Java CAPS MDM Suite Features

The Java CAPS MDM Suite establishes the data model, data storage, and data quality for an MDM solution, and also includes the ongoing management lifecycle to maintain the most accurate and current data and make that information available to diverse customers. Features of the Java CAPS MDM Suite include the following:

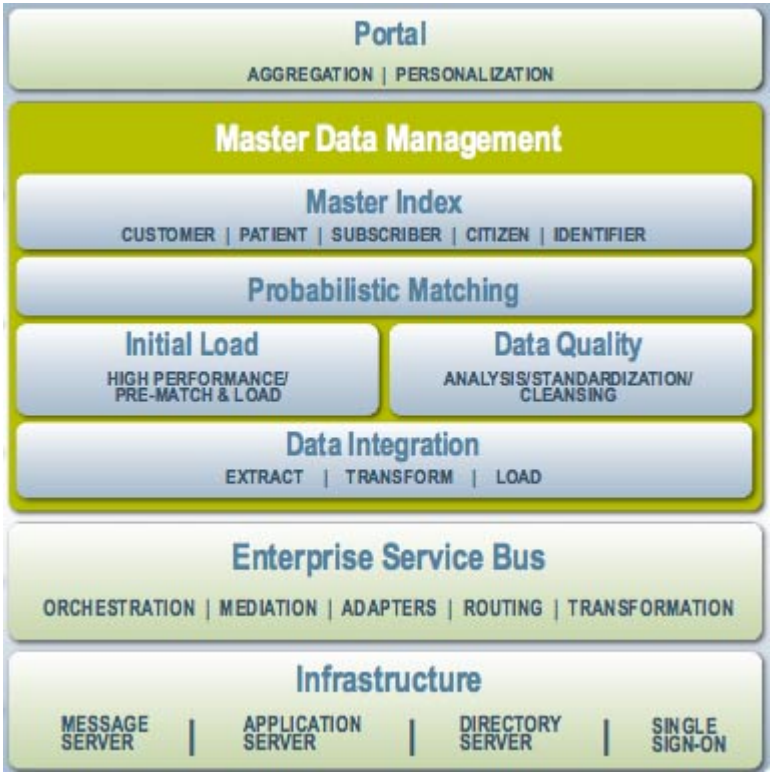
- Consolidates, cleanses, deduplicates, matches, publishes, and protects the reference data integrated from fragmented data sets.
- Improves the accuracy, visibility, and availability of an organization's data.
- Allows rapid development of new functionality and extensions to existing functionality with a flexible and extensible framework that can handle future applications and protocols.
- Creates an integrated and consistent view of master data.
- Includes a rich and intuitive web–based user interface for data stewards to review and manage master data.
- Synchronizes with external source systems, leveraging a rich integration platform.
- Provides a unified development and monitoring environment.
- Extends the single best view to partners using federated identity.
- Allows for layered levels of access for privacy and security.
- Leverages current applications, data, and systems. Changes to existing systems are minimal.
- Delivers real-time access to master data based on defined restrictions.

Java CAPS MDM Suite Architecture

The Java CAPS MDM Suite is a subset of the Java Composite Application Platform Suite (Java CAPS). It includes an infrastructure layer, Enterprise Service Bus (ESB) layer, MDM layer, and portal layer. The infrastructure layer is the foundation for deploying the MDM applications, access and security, and database connectivity. The ESB layer provides connectivity to external through adapters and business process orchestration. It also performs data transformation, mapping, and routing.

The MDM layer includes the core MDM products. Java CAPS Master Index defines the data structure for the reference data, and stores and maintains the reference data on an ongoing basis. Java CAPS Data Integrator extracts legacy data from existing systems, transforms that data if necessary, and loads it into a master index database. The Java CAPS Data Quality and Load Tools profile, cleanse, standardize, match, and load the reference data. The load tool uses Java CAPS Data Integrator for its high-performance loading capabilities.

The Portal layer defines personalized content delivery for MDM data, providing access to the reference data based on the specific needs of each user or group of users.



Master Data Management Components

Certain components of the Java CAPS MDM Suite are geared specifically to the needs of an MDM solution. These include Java CAPS Master Index, Java CAPS Data Integrator, and the Data Quality tools. These components provide data cleansing, profiling, loading, standardization, matching, deduplication, and stewardship to the MDM Suite.

Java CAPS Master Index

Java CAPS Master Index provides a flexible framework for you to design and create custom single-view applications, or *master indexes*. A master index cleanses, matches, and cross–references business objects across an enterprise. A master index that contains the most current and accurate data about each business object is at the center of the MDM solution. Java CAPS Master Index provides a wizard that takes you through all the steps of creating a master index application. Using the wizard, you can define a custom master index with a data structure, processing logic, and matching and standardization logic that are completely geared to the type of data you are indexing. Java CAPS Master Index also provides a graphical editor so you can further customize the business logic, including matching, standardization, queries, match weight thresholds, and so on.

Java CAPS Master Index addresses the issues of dispersed data and poor quality data by uniquely identifying common records, using data cleansing and matching technology to automatically build a cross-index of the many different local identifiers that an entity might have. Applications can then use the information stored by the master index to obtain a comprehensive and current view of an entity since master index operations can be exposed as services. Java CAPS Master Index also provides the ability to monitor and maintain reference data through a customized web-based user interface called the Master Index Data Manager (MIDM).

Java CAPS Data Integrator

Java CAPS Data Integrator is an extract, transform, and load (ETL) tool designed for high-performance ETL processing of bulk data between files and databases. It manages and orchestrates high-volume data transfer and transformation between a wide range of diverse data sources, including relational and non-relational data sources. Java CAPS Data Integrator is designed to process very large data sets, making it the ideal tool to use to load data from multiple systems across an organization into the master index database.

Java CAPS Data Integrator provides a wizard to guide you through the steps of creating basic and advanced ETL mappings and collaborations. It also provides options for generating a staging database and bulk loader for the legacy data that will be loaded into a master index database. These options are based on the object structure defined for the master index. The ETL Collaboration Editor allows you to easily and quickly customize the required mappings and transformations, and supports a comprehensive set of data operators. Java CAPS Data Integrator works within the MDM Suite to dramatically shorten the length of time it takes to match and load large data sets into the master index database.

Java CAPS Data Quality and Load Tools

By default, Java CAPS Master Index uses the Master Index Match Engine and Master Index Standardization Engine to standardize and match incoming data. Additional tools are generated directly from the master index application and use the object structure defined for the master index. These tools include the Data Profiler, Data Cleanser, and the Initial Bulk Match and Load (IBML) tool.

Master Index Standardization Engine
The standardization engine is built on a highly configurable and extensible framework to enable standardization of multiple types of data originating in various languages and counties. It performs parsing, normalization, and phonetic encoding of the data being sent to the master index or being loaded in bulk to the master index database. Parsing is the process of separating a field into individual components, such as separating a street address into a street name, house number, street type, and street direction. Normalization changes a field value to its common form, such as changing a nickname like Bob to its standard version, Robert. Phonetic encoding allows queries to account for spelling and input errors. The standardization process cleanses the data prior to matching, providing data to the match engine in a common form to help provide a more accurate match weight.

Master Index Match Engine
The match engine provides the basis for deduplication with its record matching capabilities. The match engine compares the match fields in two records and calculates a match weight for each match field. It then totals the weights for all match fields to provide a composite match weight between records. This weight indicates how likely it is that two records represent the same entity. The Master Index Match Engine is a high-performance engine, using proven algorithms and methodologies based on research at the U.S. Census Bureau. The engine is built on an extensible and configurable framework, allowing you to customize existing comparison functions and to create and plug in custom functions.

Data Profiler
When gathering data from various sources, the quality of the data sets is unknown. You need a tool to analyze, or profile, legacy data in order to determine how it needs to be cleansed prior to being loaded into the master index database. It uses a subset of the Data Cleanser rules to analyze the frequency of data values and patterns in bulk data. The Data Profiler performs a variety of frequency analyses. You can profile data prior to cleansing in order to determine how to define cleansing rules, and you can profile data after cleansing in order to fine-tune query blocking definitions, standardization rules, and matching rules.

Data Cleanser
Once you know the quality of the data to be loaded to the master index database, you can clean up data anomalies and errors as well as standardize and validate the data. The Data Cleanser validates, standardizes, and transforms bulk data prior to loading the initial data set into a master index database. The rules for the cleansing process are highly customizable and can easily be configured for specific data requirements. Any records that fail validation or are rejected can be fixed and put through the cleanser again. The output of the Data Cleanser is a file that can be used by the Data Profiler for analysis and by the IBML tool. Standardizing data using the Data Cleanser aids the matching process.

Initial Bulk Match and Load Tool
Before your MDM solution can begin to cleanse data in real time, you need to seed the master index database with the data that currently exists in the systems that will share information with the master index. The IBML tool can match bulk data outside of the master index environment and then load the matched data into the master index database, greatly reducing the amount of time it would normally take to match and load bulk data. This tool is highly scalable and can handle very large volumes of data when used in a distributed computing environment. The IBML loads a complete image of processed data, including potential duplicate flags, assumed matches, and transaction information.

Java CAPS MDM Integration and Infrastructure Components

The Java CAPS MDM Suite is built on a platform of integration and infrastructure applications that provide connectivity, define the flow of data, handle access and security, and route and transform data.

The Java CAPS MDM Suite can be used with the Oracle applications listed below:

Oracle Java CAPS Enterprise Service Bus

The Enterprise Service Bus is an integration platform based on Java technology and web services. It is a pluggable platform that incorporates the Java Business Integration (JBI) standard to allow loosely coupled components to communicate with each other through standards-based messaging. It provides core integration capabilities to the MDM Suite, including comprehensive application connectivity, guaranteed messaging, and transformation capabilities along with a unified environment for integration development, deployment, monitoring, and management.

Oracle Java CAPS Business Process Manager

The Business Process Manager enables long-lived, process-driven integration. It allows you to model, test, implement, monitor, manage, and optimize business processes that orchestrate the flow of activities across any number of web services, systems, people, and partners. It delivers an open, graphical modeling environment for the industry-standard business process execution language (BPEL). Java CAPS Master Index services can be called from a business process in order to share data with external systems.

Oracle Java System Access Manager

Oracle Java System Access Manager is based on open standards and delivers authentication and policy-based authorization within a single, unified framework to support composite application integration. It secures the delivery of essential identity and application information on top of the Oracle Directory Server Enterprise Edition and scales with growing business needs by offering single sign-on as well as enabling federation across trusted networks of partners, suppliers, and customers.

Oracle Directory Server Enterprise Edition

The Oracle Directory Server Enterprise Edition (DSEE) builds a solid foundation for identity management by providing a central repository for storing and managing identity profiles, access privileges, and application and network resource information. The Oracle DSEE enables enterprise applications and large-scale extranet applications to access consistent, accurate, and reliable identity data.

Oracle Java System Portal Server

The Oracle Java System Portal Server provides a user portal for collaboration with business processes and composite applications layered on top of legacy and packaged applications that are integrated using business integration components within the Java CAPS MDM Suite.

Java CAPS Adapters

Java CAPS Adapters provide extensive support for integration with legacy applications, packaged applications, and data stores through a combination of traditional adapter technology and modern JBI and Java Connector Architecture (JCA) standards-based approach.

GlassFish Enterprise Server

The GlassFish Enterprise Server is an application server that is compatible with Java Platform, Enterprise Edition (Java EE), for developing and delivering server-side applications. Once you create and configure the MDM applications, they can be deployed to the GlassFish server.

NetBeans Integrated Development Environment (IDE)

NetBeans provides a unified interface for building, testing, and deploying reusable, secure web services. The Java CAPS MDM Suite applications are created within the NetBeans project structure and include wizards and editors that are fully integrated into the NetBeans IDE.

Java CAPS Master Data Management Process

MDM Lifecycle

An effective MDM implementation involves more than just creating and running the required applications. Once the applications are in place, the MDM Suite continues to cleanse and deduplicate data and makes the updated information available to external sources. The Oracle Java CAPS MDM Suite organizes the MDM lifecycle into three phases: Creation, Synchronization, and Syndication.

- **Creation** - This phase begins with analyzing the structure of the reference data and then designing and building the master index application based on that analysis. Once the master index application is configured, the data quality tools can be generated in order to profile, cleanse, match, and load the legacy data from external systems that are part of the MDM system. This phase is iterative; the results of the profiling and match analysis steps provide you with key information to fine-tune the query, blocking, standardization, and match logic for the application. This phase also includes creating the components that will integrate the

flow of data between the MDM applications and external systems. When this step is complete, the master index application is running and its operations can be exposed as web services.

- **Synchronization** - The MDM application can propagate any reference data updates to external systems that are configured to accept such information. There are a number of methods to make this information available to external systems, including web services, Java clients, JMS Topics, business processes, and so on. Once MDM services are implemented as either passive or active services, the project can be configured to actively deliver MDM services to external systems. Synchronization keeps data in all systems current, and is an ongoing process.
- **Syndication** - Once the MDM application is running, you can create and manage virtual views on the reference data, defining who in your organization can see what information and how that information is presented. All access to information is available as services implemented by the MDM Suite in different views. For example, your accounting department might need a different set of data than the sale department requires. Syndication removes the complexity of obtaining information from multiple sources and provides a single point of access.

In addition to the above three phases of the MDM lifecycle, the Java CAPS MDM Suite applies three operational layers to control and monitor each phase: Governance, Federation, and Analytics.

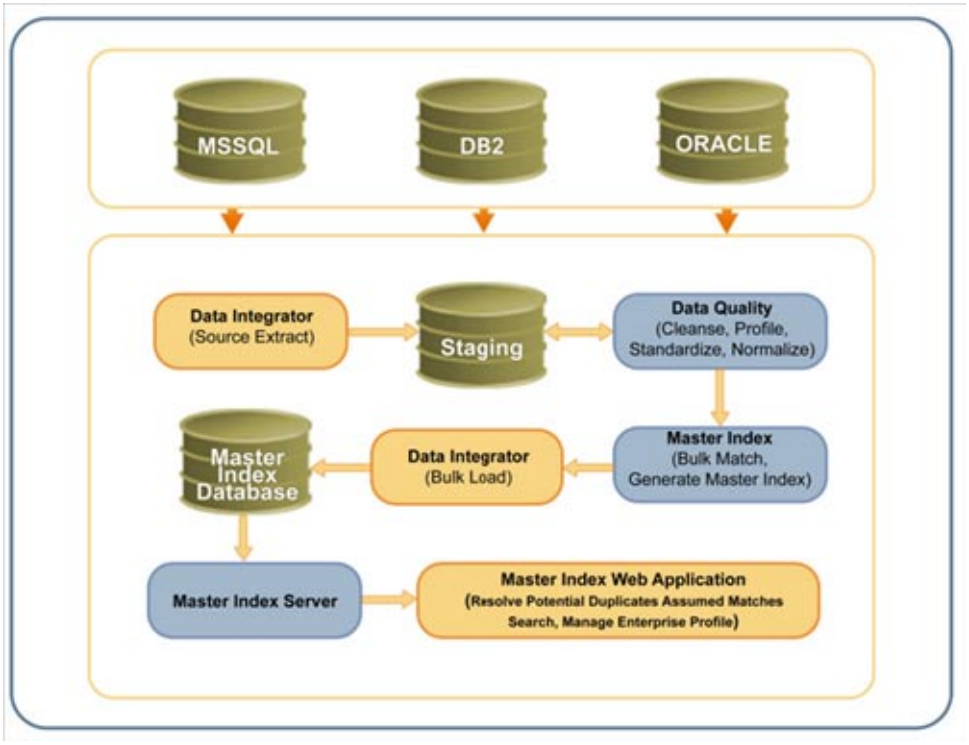
- **Governance** - This layer provides policy enforcement, reporting, and compliance to all phases of the MDM lifecycle. The standardization and matching operations form the basis of a compliance strategy, ensuring that the reference data has been strictly verified. During runtime, the MDM application controls, executes, and audits the notifications and repair of incomplete information, revealing problems at their sources. The MDM application can also govern access to master data, and you can govern the use of MDM services at a business level rather than governing technical services.
- **Federation** - This layer provides provisioning, authentication, and authorization to all phases of the MDM lifecycle. You can allow trusted business partners to view certain portions of your reference data using secure standards. This is done in secure and compliant manner with federated identity and access management.
- **Analytics** - The Java CAPS MDM Suite offers reporting, alerts, and analysis tools at all three phases to provide information about business data, including sources of quality issues, histories of deduplication, audit logs, searches, and statistics about the number and types of master data errors encountered. This is particularly important in the Creation phase, where identifying problems early can help ensure that quality issues are addressed.

MDM Workflow

The following steps describe the general workflow for implementing the Oracle Java CAPS MDM Suite solution once you create the master index application and generate the data quality tools. These steps correspond to the diagram below.

- Extract data from existing systems (Data Integrator).
- Configure standardization, cleansing, and analysis rules, and then cleanse and profile the extracted data (Data Quality).
- Match and load standardized data (Master Index and Data Integrator).
- Deploy the MDM application to perform ongoing cleansing and deduplication (Master Index Server).
- Monitor and maintain data using the data stewardship application (Master Index Web Application).

FIGURE 1 MDM Workflow Diagram



Below is a more detailed outline of the development steps required to create an MDM solution using the MDM Suite.

1. Perform a preliminary analysis of the data you plan to store in the master index application to determine the fields to include in the object structure and their attributes.
2. Create and configure the Oracle Java CAPS Master Index application, defining the object structure, standardization and match logic, queries, runtime characteristics, and any custom processing logic.
3. Create the database that will store the reference data.
4. Define security for the MIDM and any web services you will expose.
5. Generate the profiling, cleansing, and bulk match and load tools.
6. Extract the data from external systems that will be profiled, cleansed, and loaded into the master index database.
7. Analyze and cleanse the extracted data. Adjust the application configuration based on the results.
8. Perform a match analysis using the IBML tool. Adjust the matching logic based on the results.
9. Load the matched records to the master index database.
10. Build and deploy the MDM project.
11. Define connectivity to external systems using a combination of adapters, business processes, web services, Java, and JMS Topics.
12. Create any necessary presentation layer views.

About the Standardization and Matching Process

The foundation of the Java CAPS MDM Suite lies in data standardization and matching capabilities. During runtime, both matching and standardization occur when two records are analyzed for the probability of a match. In an MDM application, the standardization and matching process includes the following steps:

1. The master index application receives an incoming record.
2. The Master Index Standardization Engine standardizes the fields specified for parsing, normalization, and phonetic encoding based on customizable rules.
3. The master index application queries the database for a candidate selection pool (records that are possible matches) using the customizable blocking query.
4. For each possible match, the master index application creates a match string based on the fields specified for matching. It sends the string to the Master Index Match Engine.
5. The Master Index Match Engine checks the incoming record against each possible match, producing a matching weight for each. Matching is performed using the weighting rules defined in the match configuration file.

6. The master index application determines how to handle the incoming record based on the match weight, matching parameters, and configurable business logic. One of the following occurs:
   - A new record is added with no potential duplicates.
   - A new record is added, but is flagged as a potential duplicate of other records.
   - An existing record is updated.

Java CAPS Master Index

Java CAPS Master Index provides a flexible framework that allows you to create matching and indexing applications, known as enterprise-wide master indexes. A master index uniquely identifies and cross-references the business objects stored in your system databases using data cleansing and matching technology to create a single view of all like objects. Business objects can be any type of entity about which you store information, such as customers, members, vendors, businesses, inventory items, and so on.

The following topics provide information about Java CAPS Master Index and its components.

- "Java CAPS Master Index Overview" on page 16
- "Master Index Design and Development Phase" on page 20
- "Master Index Runtime Phase" on page 23

Java CAPS Master Index Overview

With Java CAPS Master Index, you can create and configure an enterprise-wide master index for any type of data. The Master Index Wizard guides you through the initial setup steps and special editors are provided so you can further customize the configuration, processing rules, and database structure of the master index. The wizard automatically generates the components you need to implement a master index.

Java CAPS Master Index is highly configurable, allowing you to define the data structure of the information to be indexed and to define the logic that determines how data is updated, standardized, weighted, and matched in the master index database. The master indexes created by Java CAPS Master Index provide accurate identification of objects throughout your organization and cross-reference an object's local identifiers using an enterprise-wide unique identification number (EUID) assigned by the master index. The master index also ensures accuracy by identifying potential duplicate records and providing the ability to merge or resolve duplicate records. All information is centralized in one shared index. Maintaining a centralized database for multiple systems enables the indexing application to integrate data throughout the enterprise while allowing local systems to continue operating independently.

Java CAPS Master Index Features

Java CAPS Master Index provides your business with a powerful assortment of design-time features that you can use to create and configure master index applications. The runtime features allow you to manage the master index system and to perform continuous data cleansing in real time.

Design-Time Features

The Java CAPS Master Index tools provide your business with flexibility in designing and creating indexing applications. This flexibility allows you to perform the following tasks.

- Rapidly develop a master index for any type of business entity using a wizard to create the framework and using a graphical editor to configure the attributes of the index.
- Automatically create the primary components of the master index.
- Customize the strategies that determine the field values to populate into the single best record (SBR).
- Configure the matching algorithm and logic by specifying the fields to standardize, the fields to use for matching, and the matching logic to use for each field.
- Incorporate a Java API that is customized to the object structure you define. You can call the operations in this API in the Collaboration Definitions or Business Processes of different Projects.

Runtime Features

The components of the master index application are designed to uniquely identify, match, and maintain information throughout a business enterprise. These components are highly configurable, allowing you to create a custom master index suited to your specific data processing needs. Java CAPS Master Index applications provide the following features.

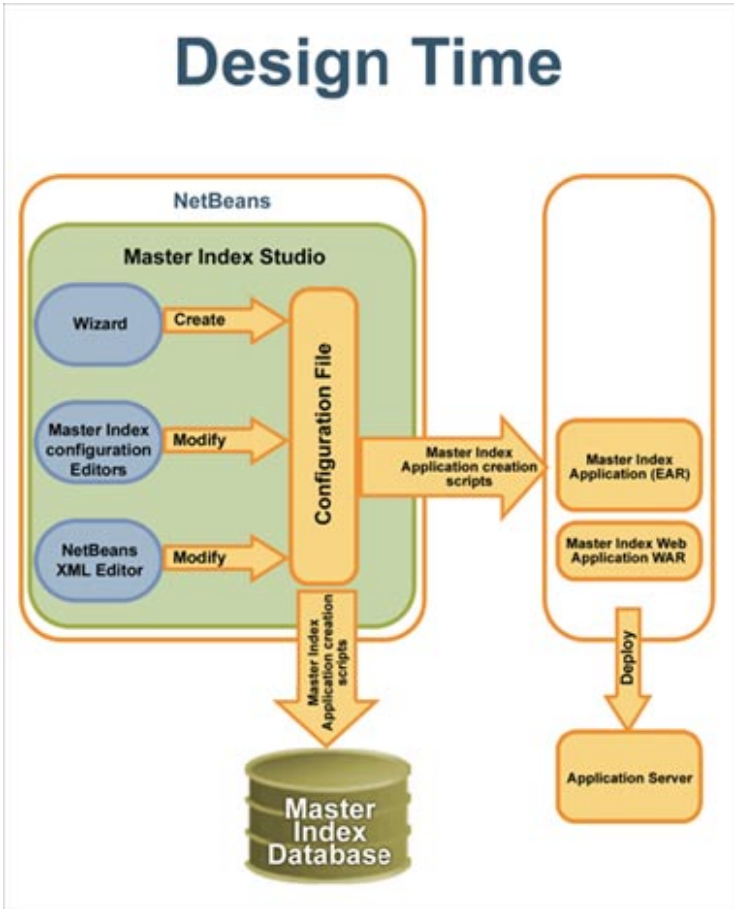
- **Centralized Information** - The master index maintains a centralized database, enabling the integration of data records throughout the enterprise while allowing local systems to continue operating independently. The index stores copies of local system records and of SBRs, which represent the most accurate and complete data for each object.
- **Configurability** - Before deploying the master index, you define the components and processing capabilities of the system to suit your organization's processing requirements. You can configure the object structure, matching and standardization rules, survivorship rules, data filters, queries, Master Index Data Manager (MIDM) appearance, and field validation rules.
- **Cross-referencing** - The master index is a global cross-indexing application that automates record matching across disparate source systems, simplifying the process of sharing data between systems. The master index uses the local identifiers assigned by your existing systems as a reference for cross-indexing, allowing you to maintain your current systems and practices.

- **Data Cleansing** - The master index uses configurable matching algorithm logic to uniquely identify object records and to identify duplicate and potential duplicate records. The index provides the functionality to easily merge or resolve duplicates. The index can be configured to automatically merge records that are found to be duplicates of one another.
- **Data Updates** - The master index provides the ability to add, update, deactivate, and delete data in the database tables through messages received from external systems. Records received from external systems are checked for potential duplicates during processing. Updates can be performed in real time or as batch processes.
- **Identification** - The master index employs configurable probabilistic matching technology, using a matching algorithm to formulate an effective statistical measure of how closely records match. Using a state-of-the-art algorithm in real time and establishing a common method of locating records, the index consistently and precisely identifies objects within an enterprise.
- **Matching Algorithm** - The matching algorithm and logic used by the master index application is highly configurable. The Master Index Match Engine and Master Index Standardization Engine are used for standardization and matching, but a master index can also be implemented with other match engines that provide a compatible API. The matching and standardization logic is customizable with a framework that allows you to plug in customized logic.
- **Unique Identifier** - Records from various systems are cross-referenced using an enterprise-wide unique identifier (EUID) that the master index assigns to each object record. The index uses the EUID to cross-reference the local IDs assigned to each object by the various computer systems throughout the enterprise.

Java CAPS Master Index Architecture

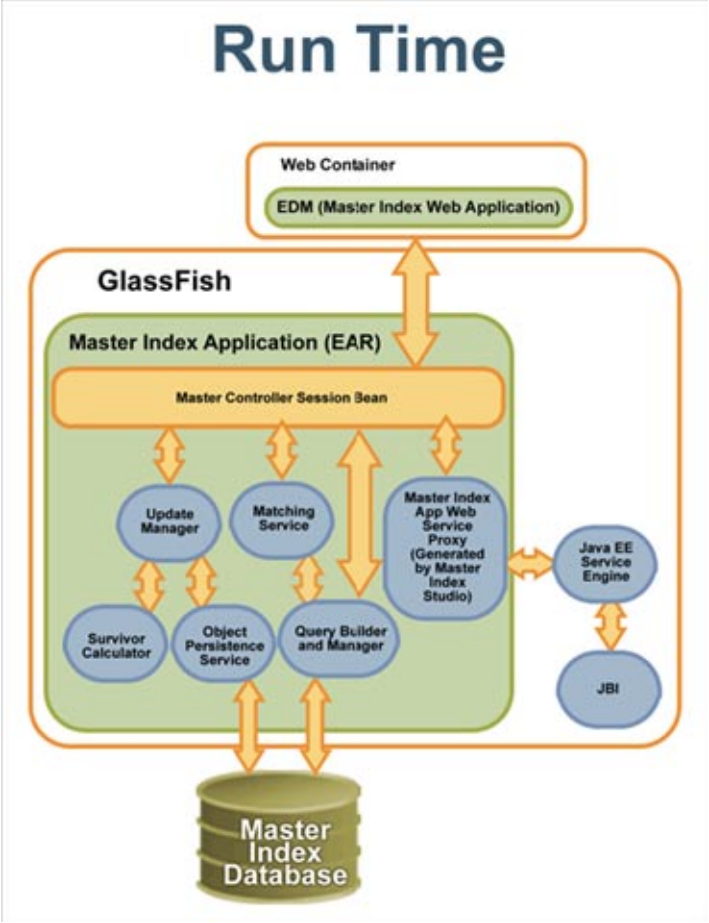
The Java CAPS Master Index design-time components allow you to define the data structure of the business objects to be stored and cross-referenced and to define the logic that determines how data is processed in the master index application. As shown in the following diagram, the design-time components include a wizard, editors, configuration files, and database scripts. When the master index project is built, a master index application is created that can be deployed to the application server.

FIGURE 2 Java CAPS Master Index Design-Time Components



Building and deploying the master index application creates the runtime components of Java CAPS Master Index, including components that process and persist data, master index services, and the Master Index Data Manager (a web-based GUI to monitor and maintain master index data). Runtime components also include the master index database. The following diagram illustrates the runtime components of a master index application.

FIGURE 3 Java CAPS Master Index Runtime Components



Master Index Design and Development Phase

The development phase consists of standard tasks for creating an indexing application and advanced tasks for further customizing the applications you create.

Analysis and Design Tasks

The process of creating a master index begins with a thorough analysis of the structure and characteristics of the data you plan to store in the master index database and to share among external systems. The results of this analysis define the structure of the information stored in the master index database and provide information to help you customize the processing and matching logic for the master index.

From this analysis you can design the object structure, matching and standardization logic, any required custom processing, and the connectivity components for the indexing system. Once you have created the master index framework, you can generate custom tools to perform a more in-depth analysis and cleansing of the actual data to be stored in the master index database. For more information, see "Data Cleanser and Data Profiler" on page 37.

Standard Development Tasks

The following steps outline the basic procedure for developing a master index using Java CAPS Master Index.

1. Create a Master Index Application project in NetBeans.
2. Using the Master Index Wizard, define the data and message structures, the operating environment, and external systems sharing data with the master index application.
3. Using the Configuration Editor, customize the application.
4. Generate the application.
5. Customize the database scripts, and then create the database.
6. Define the database connection pools and JDBC resources.
7. Define security.
8. Build and deploy the fully configured master index application.

Advanced Development Tasks

You can perform additional tasks during the development phase to customize your indexing application further.

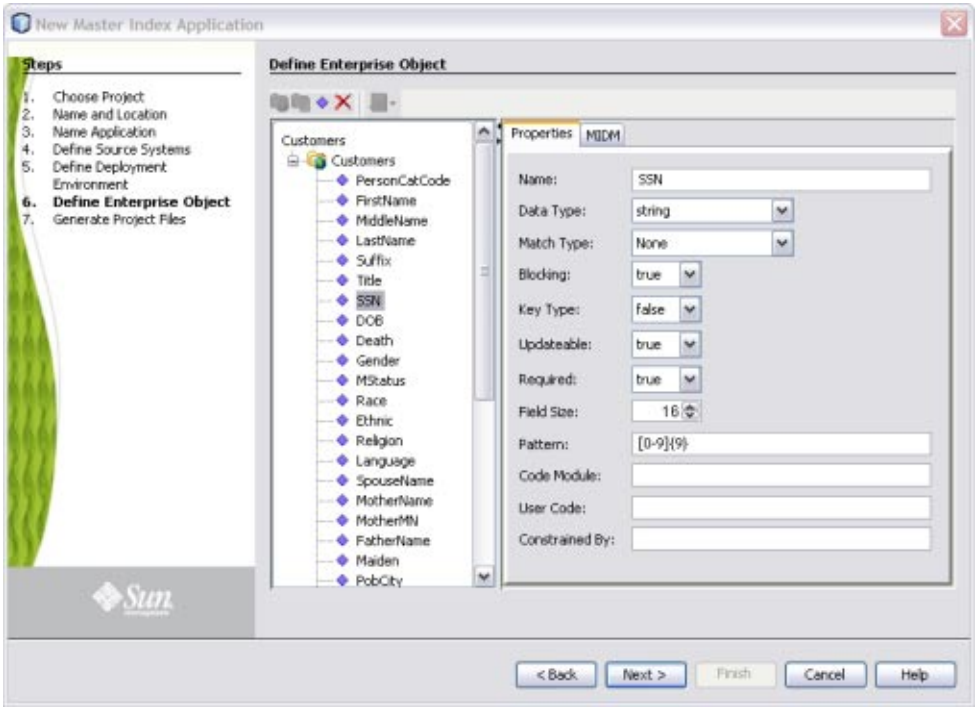
- **Data Analysis and Cleansing** – Generate tools from the master index project to help you analyze and cleanse the initial set of data to be loaded into the master index database. The analysis and cleansing steps are iterative, and each iteration will help you to fine-tune the standardization, matching, and filter logic.
- **Bulk Loading** – Generate the Initial Bulk Match and Load tools to rapidly match, deduplicate, and load the initial data set into the master index database. For more information, see "Initial Bulk Match and Load Tool" on page 39.
- **Custom Plug-ins** - Create Java classes to perform custom processing during the matching process and once the matching process is complete (such as performing additional operations before finalizing a transaction or validating certain field values).
- **Database Distribution** - Before running the predefined scripts against the database, create additional tablespaces to distribute the tables of the master index.
- **Match Engine Configuration** - Customize how weighting is performed by modifying the match engine configuration files included in the master index project.

- **External System Integration** - When you generate the master index project, a set of operations are created that are specifically tailored to the object structure you defined. Use these operations to integrate the master index application using BPEL processes, web services, or Java clients.

Master Index Wizard

The Master Index Wizard takes you through each step of the master index setup process and, based on the information you specify, creates the XML files that define the configuration of the application.

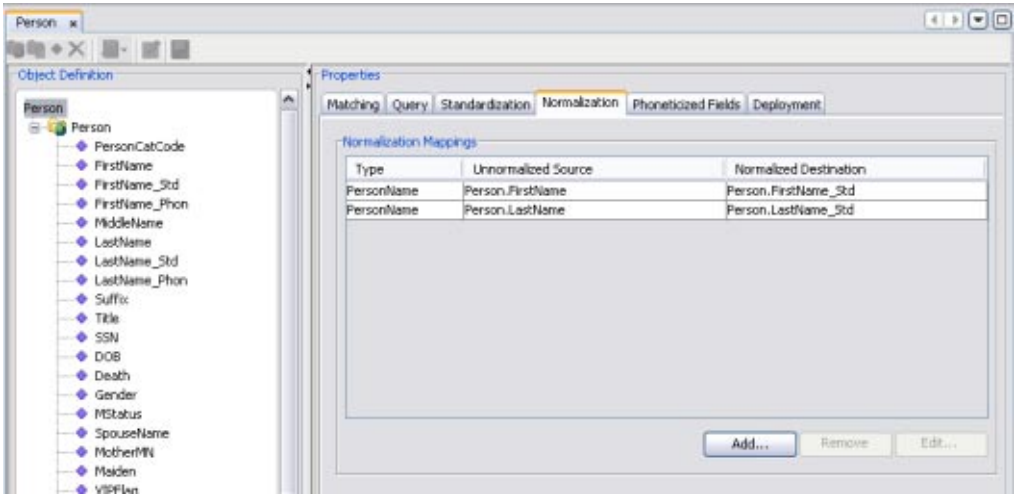
FIGURE 4 Field Properties on the Master Index Wizard



Configuration Editor

Once you create the Project files using the wizard, you can further customize the configuration of the master index application using the Configuration Editor.

FIGURE 5 Normalization Page of the Master Index Configuration Editor



With the Configuration Editor, you can customize the following:

- Object structure
- Queries
- Standardization rules, including field parsing, normalization, and phonetic encoding
- Transaction mode (XA or non-XA)
- Matching rules and thresholds

Master Index Runtime Phase

Once all of the analysis, design, and development tasks are complete and the system is running, you can perform any of these maintenance tasks.

- Transform and route data between external systems and the master index application (where the matching process occurs)
- Monitor and manage activities and alerts in the application server logs
- Monitor and maintain the indexed records in the master index database using the MIDM

Manager Service

The Manager Service provides a session bean to all components of the master index, such as the MIDM, Query Builder, Update Manager, and so on. During the runtime phase, the Manager Service performs the following tasks:

- Manages connectivity to the master index database

- Specifies the query to use for the match process and the system parameters that control the match process
- Coordinates the activities of the various components of the master index, including queries, updates, object persistence, system parameters, and so on

Master Index Database

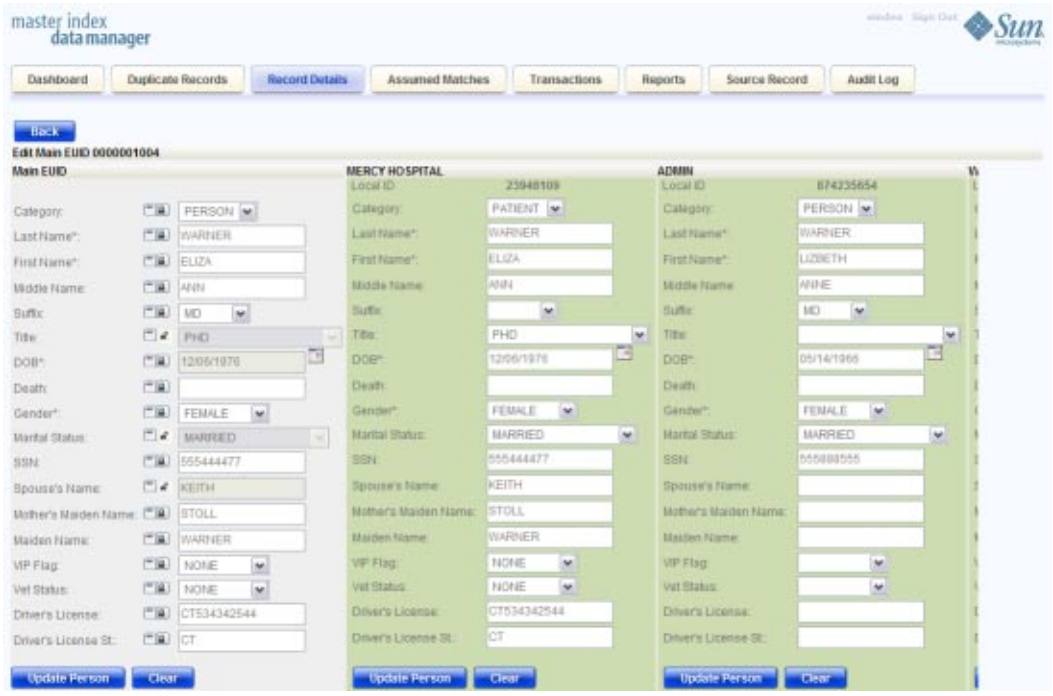
The components of a master index connect to the database to provide the following features:

- **Persistence** - The Object Persistence service writes instance data to database tables to ensure that data is able to persist in the system.
- **Recoverability** - The master index database allows you to recover data from the last state of consistency.
- **Transaction History** - The database stores a description of the changes that occur for each transaction. This allows you to view a complete history of changes to each record in the database.

Data Monitoring and Maintenance

The Master Index Data Manager (MIDM) is a web-based interface that allows you to monitor and maintain the data in your master index database. The appearance and search capabilities of the MIDM are highly configurable to allow you to view and search for information in the way that best suits your business needs. The following figure shows a sample page on the MIDM.

FIGURE 6 Master Index Data Manager



The MIDM allows you to perform these primary functions to monitor and maintain the data in a master index database.

- **Transaction History** - You can view a complete history of each object for both the local system records and the single best record.
- **Data Maintenance** - You can add new records; view, update, deactivate, or reactivate existing records; and compare records for similarities and differences.
- **Search** - You can perform searches against the database for a specific object or a set of objects. For certain searches, the results are assigned a matching weight indicating the probability of a match.
- **Potential Duplicate Detection and Handling** - Using matching algorithm logic, the master index identifies potential duplicate records and provides the functionality to correct the duplication.
- **Merge and Unmerge** - You can merge records you find to be actual duplicates of one another at either the enterprise-wide unique identifier (EUID) level or the system record level. Merges made in error can easily be unmerged.

Java CAPS Data Integrator

Extraction, transform, and load (ETL) is a data integration method that extracts data from various data sources, transforms the data into a common format, and then loads the data format into one or more target data sources. ETL processes bring together and combine data from multiple source systems into a data warehouse, enabling all users to work off a single, integrated set of data — a single version of the truth.

The following topics provide information about Java CAPS Data Integrator and its components.

Java CAPS Data Integrator Overview

Java CAPS Data Integrator manages and orchestrates high-volume, high-performance data transformation from within the SOA tier. It is optimized for extracting, transforming, and loading bulk data between files and databases and provides connectivity to a vast range of heterogeneous and diversified data sources, including non-relational data sources. It is optimized for handling very large record sets.

You can use Java CAPS Data Integrator for many purposes. You might need to acquire a temporary subset of data for reports or other purposes, or you might need to acquire a more permanent data set in order to populate a data warehouse. You can also use Java CAPS Data Integrator for database type conversions or to migrate data from one database or platform to another.

Java CAPS Data Integrator applies the following ETL methodology:

- **Extract** – The data is read from specified source databases or flat files and a specific subset of data is extracted. With Java CAPS Data Integrator, the data can be filtered and joined from multiple, heterogeneous sources.
- **Transform** – The extracted data is converted from its previous form into the proper form to be placed into a target database. Transformation occurs by using rules or lookup tables or by combining data from multiple sources.Java CAPS Data Integrator applies the operators specified for the process to transform and cleanse data to the desired state.
- **Load** – The transformed data is loaded into one or more target databases or data warehouses.

Java CAPS Data Integrator Features

Java CAPS Data Integrator provides your business with a powerful assortment of design-time features that you can use to create and configure ETL processes. The runtime features allow you to monitor the ETL processes and to review any data errors.

Java CAPS Data Integrator provides the following features:

- Stores all data transformation logic in one place and enables users, managers, and architects to understand, review, and modify the various interfaces.
- Generates a schema based on the master index object structure in order to extract data from legacy systems and load it into a staging database for cleansing and analysis.
- Can integrate with a wide variety of source data types, including HTML, XML and RSS.
- Simplifies and standardizes ETL processes, requiring little database expertise to build high performance ETL processes.
- Automatically discovers metadata, enabling you to design ETL processes faster.
- Loads data warehouses faster by taking advantage of database bulk, no-logging tuning where applicable.
- Supports creating automatic joins based on primary key and foreign key relationship, and create the code to ensure data integrity.
- Takes advantage of the database engine by pushing much of the workload on to the target and source databases.
- Supports extensive non-relational data formats.
- Provides transform, filter, and sort features at the data source where appropriate.
- Provides data cleansing operators to ensure data quality and a dictionary-driven system for complete parsing of names and addresses of individuals, organizations, products, and locations.
- Provides the ability to normalize and denormalize data.
- Converts data into a consistent, standardized form to enable loading to conformed target databases.
- Provides build-in data integrity checks.
- Allows you to define customized transformation rules, data type conversion rules, and null value handling.
- Provides a robust error handler to ensure data quality, and a comprehensive system for reporting and responding to all ETL error events. Java CAPS Data Integrator also provides automatic notification of significant failures.
- Supports concurrent and parallel processing of multiple source data streams.
- Supports full refresh and incremental extraction.
- Supports data federation that enables you to use SQL as the scripting language to define ETL processes.
- Supports near real-time click-stream data warehousing (in conjunction with the JDBC Binding Component (BC)).
- Supports ERP/CRM data sources (in conjunction with various components from Java CAPS).

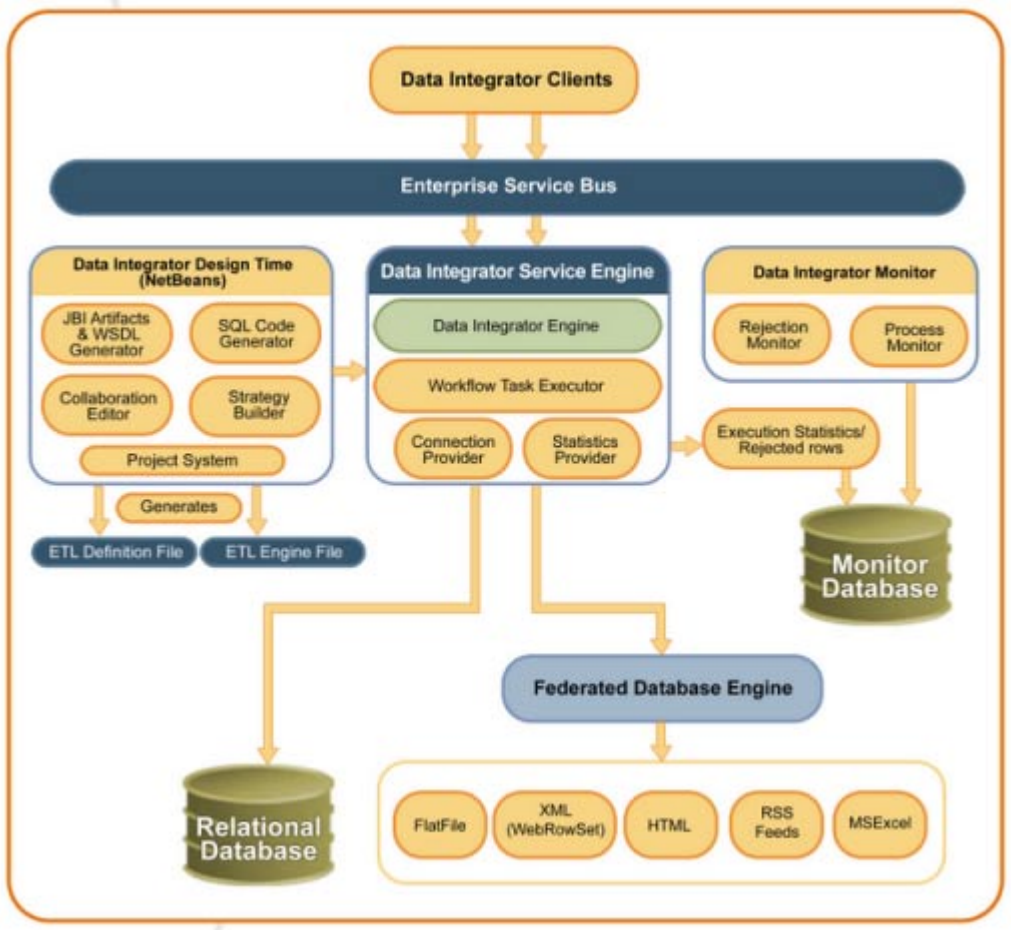
- Is platform independent, and can be scaled to enterprise data warehousing applications.
- Provides built-in transformation objects so you can easily specify complex transformations.
- Supports scheduling of ETL session based on time or on the occurrence of a specific event.
- Participates as a partner with BPEL business processes by exposing the ETL process as a web service.
- Is able to extract data from outside a firewall in conjunction with the FTP BC and the HTTP BC.
- Provides analysis of transformations that failed or were rejected and then allows for resubmitting them after the data is corrected.

Java CAPS Data Integrator Architecture

The Java CAPS Data Integrator design-time components allow you to specify the data source and target databases, map source fields and columns to target fields and columns, define custom processing, and test and validate the ETL collaboration. Design-time components include the NetBeans project system, a wizard to guide you through creating and configuring an ETL process, and a mapping editor where you can map source and target data and customize the transformation.

The runtime components include monitors to view the status of ETL processes and any rejected data. The Data Integrator Engine execute the ETL process. The following diagram shows the Java CAPS Data Integrator components and their relationship to one another. Data Integrator clients could include technologies such as web services, Java EE or .Net applications, reporting tools, or MDM applications, such as Java CAPS Master Index.

FIGURE 7 Data Integrator Architecture



Java CAPS Data Integrator Development Phase

The development phase consists of standard tasks for specifying source and target databases and advanced tasks for further customizing the data transformation logic.

Standard Development Tasks

The following steps outline the basic procedure for developing an ETL process using Java CAPS Data Integrator.

1. Connect to the source and target databases from the Services window in NetBeans.
2. Create a new Data Integrator Module project in NetBeans.

3. Using the Data Integrator Wizard, specify the source database and tables and the target database and tables.
4. Using either the Data Integrator Wizard or the ETL Collaboration Editor, specify join conditions and map the source fields or columns to the target fields or columns.
5. Specify the execution strategy.
6. Add the ETL service to a composite application.
7. Build and deploy the composite application.

Advanced Development Tasks

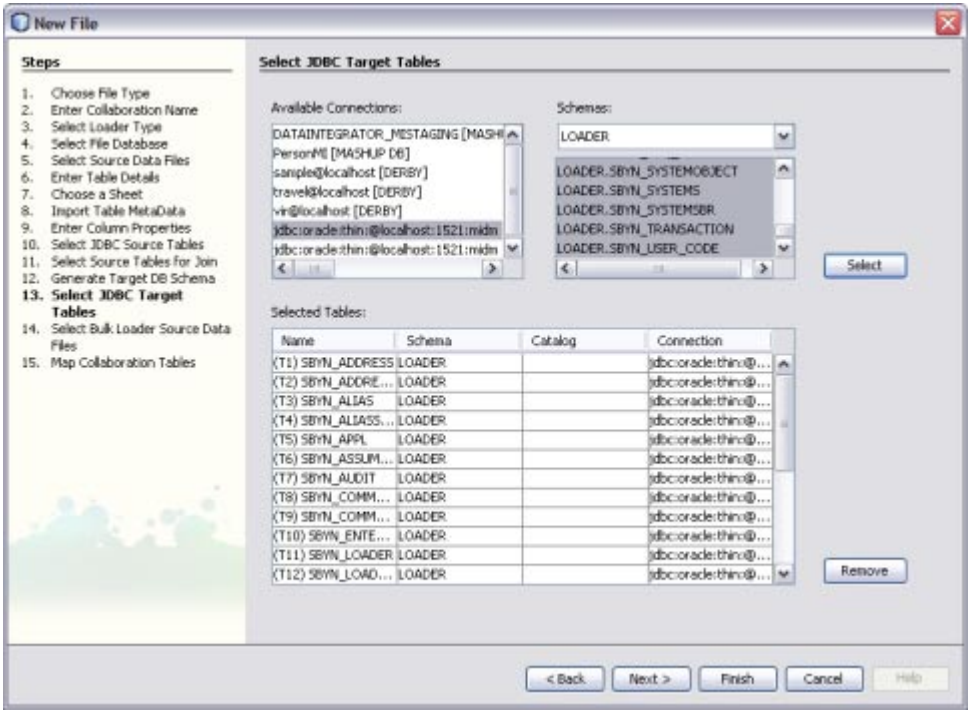
You can perform additional tasks during the development phase to customize your ETL application further.

- **Customized Data Transformation** – Use data transformation operators to define advanced standardization and cleansing rules for the source data.
- **Master Index Staging** – Create a staging database populated with the initial bulk data that will be loaded into a new master index application. The staging database is used by the master index Data Cleanser and Data Profiler prior to loading the data.
- **ETL Process Integration** - Call an ETL collaboration from a BPEL business process, web service, Java client, or other application.
- **Extraction Scheduling** - Configure a time or event that triggers a data extraction from the data source. You can extract data in batch mode or continuously based on database triggers.
- **Parallel Processing** - Configure the ETL process to run on multiple threads for better performance and faster execution.

Data Integrator Wizard

The Data Integrator Wizard takes you through each step of the ETL setup process and, based on the information you specify, creates a collaboration that defines the configuration of the ETL process.

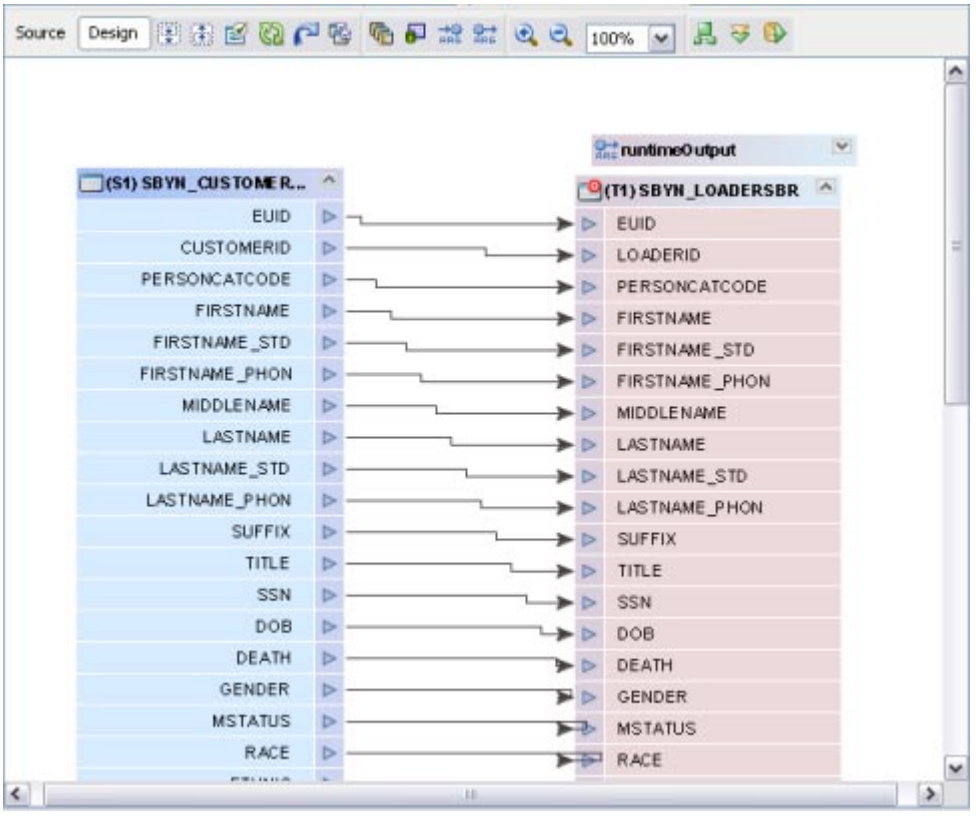
FIGURE 8 Select Target Table on the Data Integrator Wizard



ETL Collaboration Editor

Once you define the data integration framework using the wizard, you use the ETL Collaboration Editor to further customize its configuration.

FIGURE 9 ETL Collaboration Editor



Java CAPS Data Integrator Runtime Phase

Once all of the development tasks are complete and the system is running, you can perform any of these maintenance tasks.

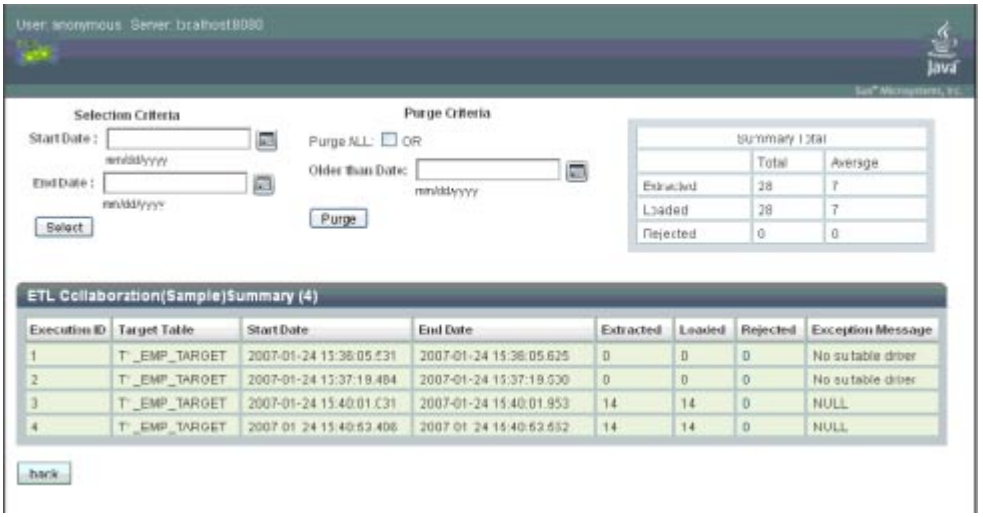
- Automatically connect to the predefined data sources and execute the ETL collaboration at specified times or when specific event occurs.
- Monitor and manage activities and alerts in the application server logs
- Modify the configuration of the ETL collaboration.

Monitoring and Maintenance

You can monitor ETL collaborations using the ETL Monitor, which is deployed on the application server Admin Console. The monitor allows you to specify a date range of events to monitor and also provides a purge function so you can remove outdated or obsolete events. For each event, the monitor displays the target table, start and end dates, the number of records

extracted and loaded, the number of rejected records, and any exception messages. You can also view a summary, and drill down into the details of rejected records.

FIGURE 10 ETL Monitor on the Admin Console



Java CAPS Data Quality and Load Tools

The Java CAPS MDM Suite includes several data quality tools that provide capabilities to analyze, cleanse, standardize, and match data from multiple sources. The cleansed data can then be loaded into the central database using the high-performance Bulk Loader. Used with the master index application, these tools help ensure that the legacy data that is loaded into a master index database at the start of the MDM implementation is cleansed, deduplicated, and in a standard format. Once the MDM solution is in production, the tools provide continuous cleansing, deduplication, and standardization so your reference data always provides the single best view.

- "Master Index Standardization Engine" on page 33
- "Master Index Match Engine" on page 35
- "Data Cleanser and Data Profiler" on page 37
- "Initial Bulk Match and Load Tool" on page 39

Master Index Standardization Engine

The Master Index Standardization Engine parses, normalizes, and phonetically encodes data for external applications, such as master index applications. Before records can be compared to evaluate the possibility of a match, the data must be normalized and in certain cases parsed or

phonetically encoded. Once the data is conditioned, the match engine can determine a match weight for the records. The standardization engine is built on a flexible framework that allows you to customize the standardization process and extend standardization rules.

Standardization Concepts

Data standardization transforms input data into common representations of values to give you a single, consistent view of the data stored in and across organizations. This common representation allows you to easily and accurately compare data between systems.

Data standardization applies three transformations against the data: parsing into individual components, normalization, and phonetic encoding. These actions help cleanse data to prepare it for matching and searching. Some fields might require all three steps, some just normalization and phonetic conversion, and other data might only need phonetic encoding. Typically data is first parsed, then normalized, and then phonetically encoded, though some cleansing might be needed prior to parsing.

A common use of normalization is for first names. Nicknames need to be converted to their common names in order to make an accurate match; for example, converting "Beth" and "Liz" to "Elizabeth". An example of data that needs to be parsed into its individual components before matching is street addresses. For example, the string "800 W. Royal Oaks Boulevard" would be parsed as follows:

- Street Number: 800
- Street Name: Royal Oaks
- Street Type: Boulevard
- Street Direction: W.

Once parsed the data can then be normalized so it is in a common form. For example, "W." might be converted to "West" and "Boulevard" to Blvd" so these values are similar for all addresses.

Phonetic encoding allows for typos and input errors when searching for data. Several different phonetic encoders are supported, but the two most commonly used are NYSIIS and Soundex.

Master Index Standardization Engine Configuration

The Master Index Standardization Engine uses two frameworks to define standardization logic. One framework is based on a finite state machine (FSM) model and the other is based on rules programmed in Java. In the current implementation, the person names and telephone numbers are processed using the FSM framework, and addresses and business names are processed using the rules-based framework. Both frameworks can be customized as needed.

A finite state machine (FSM) is composed of one or more states and the transitions between those states. In this case, a state is a value within a text field, such as a street address, that needs to be parsed from the text. The Master Index Standardization Engine FSM framework is designed to be highly configurable and can be easily extended. Standardization is defined using a simple markup language and no Java coding is required.

The Master Index Standardization Engine rules-based framework defines the standardization process for addresses and business names in Java classes. This framework can be extended by creating additional Java packages to define processing.

Both frameworks rely on sets of text files that help identify field values and to determine how to parse and normalize the values.

Master Index Standardization Engine Features

The Master Index Standardization Engine provides proven and extensive standardization capabilities to the Java CAPS MDM Suite, and includes the following features:

- Works with Java CAPS Master Index applications and can also be called from other applications, such as Data Integrator, web services, web applications, and so on.
- Uses standardization algorithms based on research at the U.S. Census Bureau, Statistical Research Division (SRD).
- Is highly configurable and can be used to standardize various types of data.
- Supports data sets specific to Australia, France, Great Britain, and the United States by default, and can be extended to support additional locales.
- Processes data using one of the defined locales or using multiple locales.
- Supports a variety of data types, including addresses, person names, businesses, and telephone numbers. Additional data types can be easily added.
- Provides comprehensive person name normalization tables for the four default locales.
- Uses a probability-based mechanism to resolve ambiguity during processing.
- Allows you to apply cleansing rules prior to the standardization processing.
- Performs preprocessing, matching, and postprocessing during the parsing process based on customizable rules.
- Is highly configurable and the standardization and matching logic can be adapted to specific needs. New data types or variants can be created for even more customized processing.
- Supports pluggable standardization sets, so you can define custom standardization processing for most types of data.

Master Index Match Engine

The Master Index Match Engine provides record matching capabilities for external applications, including Java CAPS Master Index applications. It works best along with the Master Index Standardization Engine, which provides the preprocessing of data that is required for accurate matching. The match engine compares records containing similar data types by calculating how closely certain fields in the records match. A weight is generated for each field and the sum of those weights (the composite weight) is either a positive or negative numeric value that represents the degree to which the two sets of data are similar. The composite weight

could also be a function of the match field weights. The match engine relies on probabilistic algorithms to compare data using a comparison function specific to the type of data being compared. The composite weight indicates how closely two records match.

Master Index Match Engine Matching Weight Formulation

The Master Index Match Engine determines the matching weight between two records by comparing the match string fields between the two records using the defined rules and taking into account the matching logic specified for each field. The match engine can use either matching (m) and unmatching (u) conditional probabilities or agreement and disagreement weight ranges to fine-tune the match process.

M-probabilities and u-probabilities use logarithmic functions to determine the maximum agreement and minimum disagreement weights for a field. When agreement and disagreement weights are used instead, the maximum agreement and minimum disagreement weights are specified directly as integers.

The Master Index Match Engine uses a proven algorithm to arrive at a match weight for each match string field. It offers a comprehensive array of comparison functions that can be configured and used on different types of data. Additional comparison functions can be defined and incorporated into the match engine. The master engine allows you to incorporate weight curves, validations, class dependencies, and even external data files into the configuration of a comparison function.

Master Index Match Engine Features

The Master Index Match Engine provides comprehensive record matching capabilities to the Java CAPS MDM Suite using trusted and proven methodologies. It includes the following features:

- Uses proven matching methodologies based on research at the U.S. Census Bureau, SRD, and customized for stable, reliable, and high-speed matching.
- Uses a combination of probabilistic and deterministic matching.
- Is very flexible and generic, allowing you to customize existing matching rules and to define custom rules as needed.
- Provides a comprehensive array of customizable comparators for matching on various types of fields, such as numbers, dates, single characters, and so on. Includes more specialized comparison functions for searching on specific types of data, such as person names, address fields, social security numbers, genders.
- Supports the creation of custom comparison functions to enable matching against any type of data. This is possible because the match engine is built on a pluggable architecture.
- Supports a variety of configurations for comparison functions, including validations, weighting curves, class dependencies, and the ability to access a data file that provides additional information on how to match data.

- Allows you to create new validation and configuration rules.
- Provides a simple and clear method to incorporate customizations into Java CAPS Master Index applications.
- Matches on free-form fields by using the features of the Master Index Standardization Engine prior to matching.
- Outputs the composite match weight as well as the individual, field-level match weights.

Data Cleanser and Data Profiler

Data analysis and cleansing are essential first steps towards managing the quality of data in a master index system. Performing these processes early in the MDM project helps ensure the success of the project and can eliminate surprises down the road. The Data Cleanser and Data Profiler tools, which are generated directly from the Java CAPS Master Index application, provide the ability to analyze, profile, cleanse, and standardize legacy data before loading it into a master index database. The tools use the object definition and configuration of the master index application to validate, transform, and standardize data.

The Data Profiler examines existing data and gives statistics and information about the data, providing metrics on the quality of your data to help determine the risks and challenges of data integration. The Data Cleanser detects and corrects invalid or inaccurate records based on rules you define, and also standardizes the data to provide a clean and consistent data set. Together, these tools help ensure that the data you load into the master index database is standard across all records, that it does not contain invalid values, and that it is formatted correctly.

About the Data Profiler

The Data Profiler analyzes the frequency of data values and patterns in your existing data based on rules you specify. Rules are defined using a rules definition language (RDL) in XML format. The RDL provides a flexible and extensible framework that makes defining rules an easy and straightforward process. A comprehensive set of rules is provided, and you can define custom rules to use as well.

You can use the Data Profiler to perform an initial analysis of existing data to determine which fields contain invalid values or format; the results will analysis highlight values that need to be validated or modified during the cleansing process. For example, if you find you have several dates in the incorrect format, you can reformat the dates during cleansing.

After the data is cleansed using the Data Cleanser, you can perform a final data analysis to verify the blocking definitions for the blocking query used in the master index match process. This analysis indicates whether the data blocks defined for the query are too wide or narrow in scope, which would result in an unreliable set of records being returned for a match search. This analysis can also show how reliably a specific field indicates a match between two records, which affects how much relative weight should be given to each field in the match string.

The Data Profiler performs three types of analysis and outputs a report for each field being profiled. A *simple frequency analysis* provides a count of each value in the specified fields. A *constrained frequency analysis* provides a count of each value in the specified fields based on the validation rules you define. A *pattern frequency analysis* provides a count of the patterns found in the specified fields.

About the Data Cleanser

The Data Cleanser validates and modifies data based on rules you specify. Rules are defined using the same RDL as the Data Profiler, which provides a flexible framework for defining and creating cleansing rules. A comprehensive rule set is provided for validating, transforming, and cleansing data, and also includes conditional rules and operators. The Data Cleanser not only validates and transforms data, but also parses, normalizes, and phonetically encodes data using the standardization configuration of the master index project.

The output of the Data Cleanser is two flat files; one file contains the records that passed all validations and was successfully transformed and standardized, and the other contains all records that failed validation or could not be transformed correctly. The bad data file also provides the reason each record failed so you can easily determine how to fix the data. This is an iterative process, and you might run the Data Cleanser several times to make sure all data is processed correctly.

The final run of the Data Cleanser should produce only a good data file that conforms to the master index object definition and that can then be fed to the Initial Bulk Match and Load tool. The data file no longer contains invalid or default values. The fields are formatted correctly and any fields that are defined for standardization in the master index application are standardized in the file.

Data Cleanser and Data Profiler Features

The Data Cleanser and Data Profiler give you the ability to analyze, validate, cleanse, and standardize legacy data before it is loaded into the central database. They provide the following features:

- Provide valuable insight into the current state and quality of legacy data.
- Provide an extensive and flexible set of validation, transformation, and cleansing rules for both analysis and cleansing.
- Allow you to easily specify and configure data processing rules using a simple rules markup language.
- Allow you to define custom processing rules in Java that can be incorporated into the rules engine.
- Provide information about field values to help determine a reliable a field is for matching purposes.
- Output a data file that has been cleansed of errors, invalid data, and invalid formats, and that conforms to the object structure of the reference data.

Initial Bulk Match and Load Tool

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 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. The IBML Tool provides a scalable solution that can be run on multiple processors for better performance.

The IBML Tool is generated from the master index application, and 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.

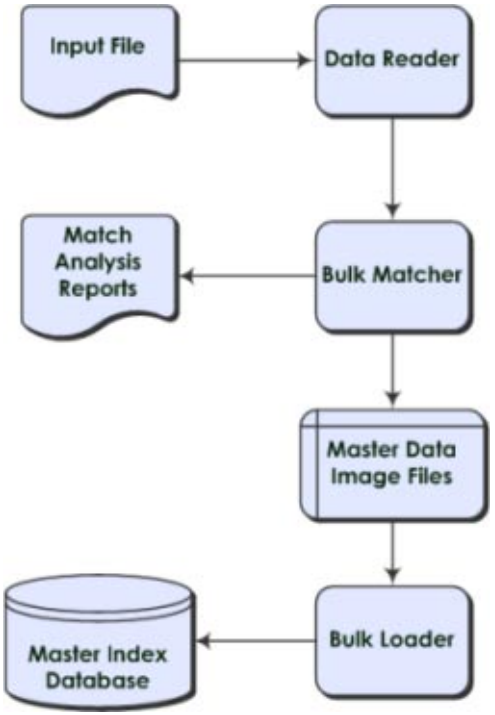
Initial Bulk Match and Load Process Overview

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 and the blocking query for the master index application. Analyzing the data in this way is an iterative process, and the Bulk Matcher provides a configuration file that you can modify to test and retest the settings before you perform the final configuration of the master index application.

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 11 Initial Bulk Match and Load Tool Process Flow

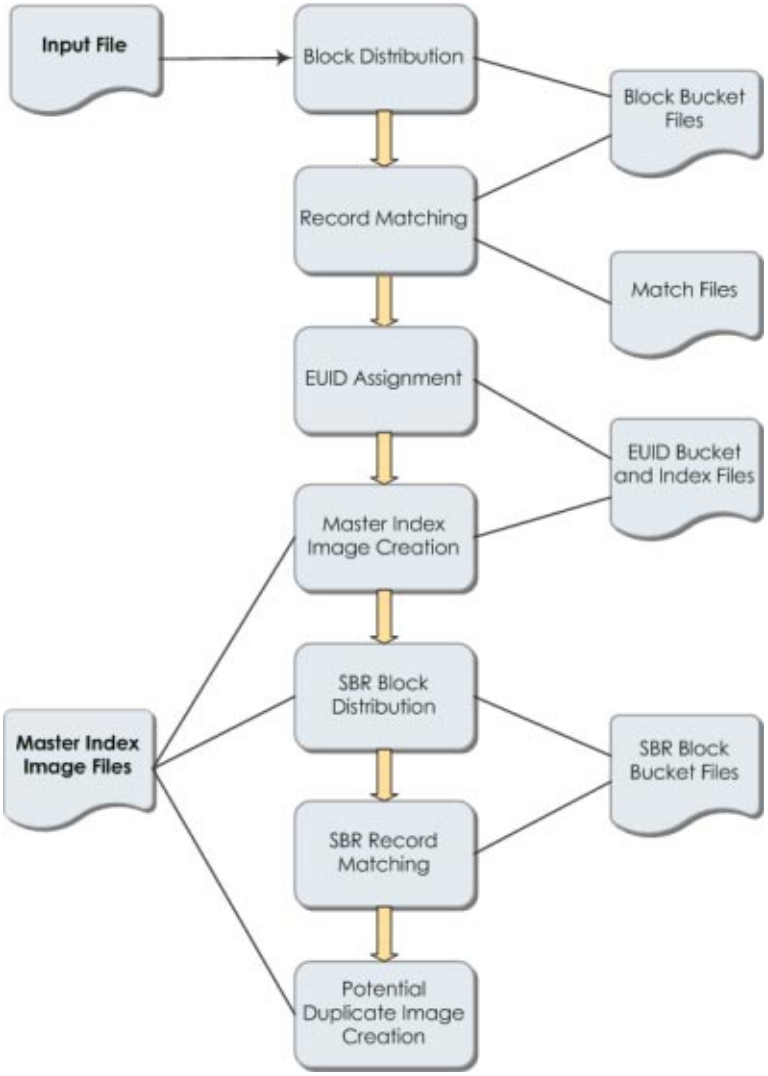


About the Bulk Match Process

The Bulk Matcher performs a sequence of tasks to prepare the master image that will be loaded into the master index database. The first phase groups the records into buckets that can then be distributed to each matcher to process. Records are grouped based on the blocking query. The second phase matches the records in each bucket to one another and assign a match weight. The third phase merges all matched records into a master match file and assigns EUIDs (EUIDs are the unique identifiers used by the master index to link all matched system records). The fourth phase creates the master image of the data to be loaded into the master index database. The master image includes complete enterprise records with SBRs, system records, and child objects, as well as assumed matches and transactional information. The final phase generates any potential duplicate linkages and generate the master image for the potential duplicate table.

The following diagram illustrates each step in more detail along with the artifacts created along the way.

FIGURE 12 Bulk Matcher Internal Process



About the Bulk Load Process

After the matching process is complete, you can load the data using either the Data Integrator Bulk Loader or a SQL*Loader 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. Data Integrator provides a wizard to help create the ETL collaboration that defines the logic used to load the master images.

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.

IBML Tool Features

The IBML Tool provides high-performance, scalable matching and loading of bulk data to the Java CAPS MDM Suite. It provides the following features:

- Includes a match analysis tool that can be used to test and analyze the values of the match threshold and duplicate threshold. (Depending on certain matching parameters, records with a match weight above the match threshold are automatically matched, and records with a match weight between the match threshold and the duplicate threshold are considered potential duplicates.)
- Quickly and accurately performs the matching required for a high volume of legacy data that will become the MDM reference data.
- Provides a highly scalable and powerful loading mechanism that dramatically reduces the length of time required to load bulk data.
- Uses a cluster-based architecture to distribute the processing over multiple servers, so all activities are performed concurrently by all servers.
- Reduces the time and resources required to perform a bulk match and load by first grouping records into blocks and then matching within each block rather than matching each record in sequence.
- Synchronizes activities between all match and load processes, with a cluster of processors executing the same activity at any point. A cluster synchronizer coordinates activities across all components and processors.
- Uses a sequential file I/O to read and write intermediate data.
- Performs load balancing across all servers dynamically by having each server process one block of data at a time. Once a server completes a block, it picks up the next one to process.
- Provides a default data reader that reads a flat file in the format output by the Data Cleanser, but also allows you to define a custom data reader for other formats.
- Uses the existing configuration of the master index project for blocking and matching, and generates the master images based on the object structure of the master index.
- Data Integrator provides a convenient wizard to help you generate the ETL collaboration that defines the load process. You can also use a command-line utility instead.