



Multi-Tiered Object Model Functional Requirements Specification In Master Data Management

Version 1.0

Who	When	What
Charles Ye	June-24-2008	Created.
Charles Ye	June-30-2008	Modified data model relationship schema definition based on Raymond Tam's suggestion.
Charles Ye	June-30-2008	Updated based on the meeting: Suhanto, Swaranjit, Kevin, Raymond, Andrea.
Charles Ye	August-20-2008	Updated.

1 Purpose of this Document

This feature definition document provides

- 1 The **requirements** of the function in enough detail to base a design and implementation on.
- 2 A **high level architecture design** briefs design and implementation approach choices.
- 3 A **discussion** of different approaches to satisfy the requirements with a decision which approach to take.
- 4 A **plan of attack** that shows steps or milestones in delivering the functionality to satisfy the requirements.
- 5 **User stories** that show how the new functionality works.

Intended audience

- 1 MDM team members
- 2 QE team members
- 3 Documentation team members

2 Functional Requirements Specification

2.1 Introduction

The data object model is the model that describes how data is represented and accessed and is the fundamental of the master index data management. The multi-tiered data object model, a.k.a. multi-level data object model, or the depth of data model in the industry terminology, defines how deep the customer data model can be constructed. It is a hierarchy data model. Ideally the depth should be unlimited. In the hierarchical data model, data is organized into a tree-like structure which allows repeating information using parent/child relationships, each parent can have many children and each child can have many children, each child could have many parents. All attributes of a specific entity node are listed under an entry type node. Therefore, with the support of the unlimited multi-tiered object model, the master data management makes the customer data modeling more flexible and more efficiently supports hierarchical relationships.

The current data object model in the mural master data management only supports two-tiered. More specifically data object node only processes two layers: parent object node and children object nodes. Each parent object node can have many children object nodes, but each child object node can not have any child object node, each child object node only can have one parent object node. With this two-layer data structures capability the coverage of data modeling of the master data management is limited.

2.2 Motivation and Goals

The depth of data model and the data model flexibility are two major Forrester evaluation criteria. Competitors already support multi-tiered object model. The goal is to extend the mural master index to support multi-tiered object model with unlimited levels in order to extend to cover the hierarchical data model and relationship.

2.3 Multi-tiered Object Model Definition Specification

2.3.1 The multi-tiered object model is a hierarchical data model in which data is organized into a tree-like structure. It is a hierarchy of parent and child data segments. Each parent can have many children and each child can have many children. A leaf has no children. The parent-child relationship is one-to-many. This restricts a child to having only one parent. Many-to-many parent-child data structure or a child has many parents is not supported in the current plan and it can be done if the market requires. This object model constraint does not apply to the entity relationship model. one-to-one, one-to-many, many-to-many and many-to-one relationships can be supported.

2.3.2 The depth of the data object model is unlimited. There is no limit how deep data model can be defined and processed.

2.3.3 Currently only one primary object is supported in a specific domain. The question is: Are multiple primary objects required to support in a specific domain? A discussion is about competitor's group concept which can allow multiple primary objects grouped under one domain, the use cases would be service and account, telecommunication subscriber model. A debate on grouping is that a grouping is a special type of relationships. And should not category to the data modeling. More details on grouping concept and use cases of multiple primary objects and market requirements need to be investigated and clarified. Currently we favor support only one primary object unless we have a clear market requirement.

2.3.4 Category and taxonomic hierarchy relationships support.

2.3.5 Enterprise object definition constraints: in the current two-layer enterprise object definition mode, both sub object name and primary object name must be unique in the object tree. Any sub object name can not be the same as the primary object name and any sub object name can not be duplicated. In the multi-tiered object model definition:

- The primary object name must be unique.
- The sub object names should be unique in the same layer of the same parent.
- The sub object can not have the same name as its parent.
- The sub object names are in different sub data object tree, can they be the same? There are two options as addressed in 2.3.6.

2.3.6 Are the sub object names required to be unique if they are in the different sub object trees? There are two solutions.

Solution 1: Make the names of sub objects, or entity types, tables in different sub object trees be unique. This is an acceptable solution for the first cut. But the question is: do we require the common

tables, like SBYN_TRANSACTION table, be unique? One use case is that allowing the same account for multiple domains applications due to connection licensing. This issue has been addressed a while ago already.

Solution 2: Allow the names of sub objects, or entity types, tables in different sub object trees duplicated during design time. Adding a qualified path as a prefix into the table name makes the table name be unique. The maximum length of the table name could be limited for various databases: that of SQL server 2000 is 128 characters and of oracle 9i is 30 characters. The length of table name could be larger than what is allowed. The hash-code could be used to solve the problem. This solution does not change any object model definition schema which is currently used.

We favor the solution 1 in the first cut based on the current market requirement.

2.5.7 Backward compatibility. Any existing MDM project with two-layer data object model project should work (can be opened and built) with the MDM V2.1 which supports multi-tiered object model. And any two-layer data object model project created by the MDM V2.1 does not necessarily work (can be opened and built) with the MDM R6. We could break that compatibility if we see any specific reason to do so.

2.5.8 The Object data model definition schema remain the same. The object model is still defined in object.xml using <nodes> element. The design view mode is defined in midm.xml using node type:

```
<xs:element name="node">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="field" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

and the object mode is defined in object.xml using nodes type:

```
<xs:element name="nodes">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tag"/>
      <xs:element ref="fields" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.5.9 The relationship schema element is not necessary to add in midm.xml and will be removed and will be only defined in object.xml. The relationship schema element will be changed to a recursive hierarchy schema element definition in object.xml as:

```
<xs:element name="hierarchy">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="node" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```

    </xs:complexType>
  </xs:element>

```

```

<xs:element name="node">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element ref="node" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

note: maxOccurs="1" indicates one root node (primary object). To support multiple primary objects, extend maxOccurs to be "unbounded". The old relationship schema:

```

<xs:element name="relationships">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="children" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

will still be supported in a case that the MDM V2.1 works with the previous two-layer data object projects.

2.5.10 Two multi-tiered object model templates solutions will be offered: telecommunication subscriber model and patient model.

2.5.11 The web-based MIDM application has to be enhanced to support multi-tiered object model. The tree-view visual presentation of multi-tiered object model will be introduced. The look and feel should remain the same as that in R6.

2.5.12 A visual Editor is an impressive tool which provides graphical presentation of object model. The current tree editor will be replaced by the visual editor which allows to create multi-tiered object model.

2.5.13 Hibernate is a powerful object/relational mapping, persistence and query service. The Hibernate lets you develop persistent classes following common Java idiom - including association, inheritance, polymorphism, composition and the Java collections framework. Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities and can significantly reduce development time otherwise spent with manual data handling in SQL and JDBC. OPS layer could be replaced by Hibernate. The problem is whether Hibernate degrades performance and whether the MDM gains other benefits.

2.4 Components Enhancement Requirements Overview

There are many components in the mural master data management suite. What this document focuses is on the master index functionality. From the development point of view, each package should be investigated and ensured to support multi-tiered data object structures. The functional components which are discussed here may already support multi-tiered object model or require limited changes, all will be detailed during development.

2.4.1 Master Index

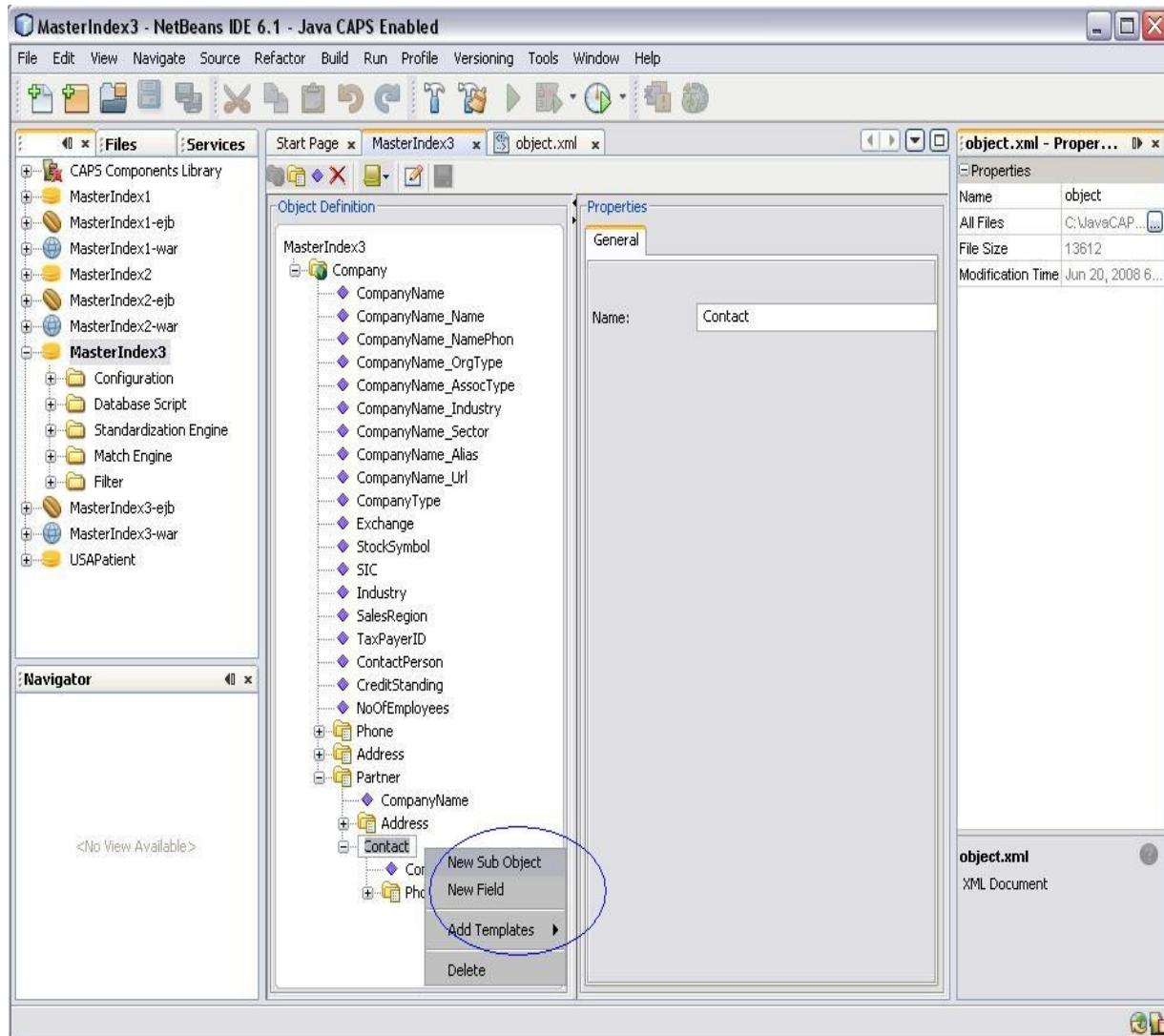
2.4.1.1 Design Time

The master index studio is a NetBeans-based design-time environment which needs to be enhanced.

- Multi-tiered object definition wizard

The current object definition wizard is only allow a user to create two-layer object model: parent and children. From the children layer, a user can not go down one more layer to create its child. This wizard is enhanced to allow to add any new sub-object from any sub-object layer. The figure 3.1 shows a sample of multi-tiered object definition wizard. The Contact is a sub-object of Partner which is a sub-object of Company. The context menu of sub-object Contact is the same as that of primary object Company. A user can create a new sub-object, a new field and add a sub-object from templates.

Figure 3.1 Multi-tiered Object Definition Wizard



- Extend the master index object schema(object.xsd) to support multi-tiered object model. The current object model schema already supports multi-tiered object model. The work will be limited.
- Extend the master index studio to generate multi-tiered object definition metadata(object.xml, midm.xml) once it is created.
- Extend the master index studio to configure the multi-tiered object master index application.

- Extend to configure the multi-tiered object matching properties.
- Extend to generate the multi-tiered-object-supported matching configuration metadata (mefa.xsd and mefa.xml).
- Extend to configure the multi-tiered object queries.
- Extend to generate the multi-tiered-object-supported queries configuration metadata(query.xsd, query.xml).
- Extend configure the multi-tiered object fields for standardization.
- Extend to generate the multi-tiered-object-supported standardization configuration metadata (mefa.xsd and mefa.xml).
- Extend to configure the multi-tiered object fields for normalization.
- Extend to generate the multi-tiered-object-supported normalization configuration metadata (mefa.xsd and mefa.xml).
- Extend to configure the multi-tiered object fields for phonetic encoding.
- Extend to generate the multi-tiered-object-supported phonetic encoding configuration metadata (mefa.xsd and mefa.xml).
- The update schema(update.xsd, update.xml) needs to be extended to support multi-tiered object model.
- Add multi-tiered object model domain template. The telecommunications subscriber model is suggested. And other domains could be account, product information management, etc.
- The master index studio should generate and package the multi-tiered-object-model-support master index application.

2.4.1.2 Run Time

Extend the master index runtime components to support multi-tiered object model.


- Matching service
The matching service provides the logic for standardization, data parsing and normalization, phonetic encoding and matching and along with the configuration. This service needs to be extended to support multi-tiered object model.
- Query builder
The query builder defines all queries available to the master index. It includes automatic queries performed by the master index when searching for possible matches to an incoming record, and manual queries through the master index data manager(MIDM). This component needs to be extended to support multi-tiered object model.
- Query manager
The query manager is a service that performs queries against the master index database and returns a list of objects that match or closing match the query criteria. This component needs to be extended to support multi-tiered object model.
- Update manager or Transaction Manager
The update manager manipulates how updates are made to an entity's single best record (SBR) by defining a survivor strategy for each field. This component needs to be extended to support multi-tiered object model.
- Object persistence manager
The object persistence manager is a database service mapping from the data object model

into the database and from the database to the data object model. This component needs to be extended to support multi-tiered object model.

- Master index application web service
Extend to support multi-tiered object model.

master index
data manager

edimuser Sign Out



DashboardDuplicate RecordsRecord DetailsAssumed MatchesTransactionsReportsSource RecordAudit Log

ViewEditAddMerge

< View Record List

Status:activeName Of The Source:SystemALocal ID:000000001

Person

Address

Alias

Phone

Person_Fields:

Person Cat CodeSun

First NameJONATHAN

Middle NameG

Last NameMCNEALY

Suffixcode description

Titlecode description

SSN000000001

Date of Birth08/08/1982

Death

GenderMale

Marital StatusMarried

RaceWhite

Ethniccode description

Religioncode description

Languagecode description

Spouse NameEmily

Mother NameEmma

Mother Maiden NameA

Father NameWilliam

MaidenEmma

Pob CityMENLO PARK

Pob StateCalifornia

Pob CountryUSA

VIP FlagVIP

Veteran StatusTenure

StatusRetired

Driver LicenseA00000001

Driver License StValidated

Date of Death

Death CertificateNone

Nationalitycode description

Citizenshipcode description

Address_Fields:

Address Typecode description

Address Line1MON01

Address Line2800 ROYAL OAKS

CityMONROVIA

State CodeCA

Postal Code91016

Postal Code Ext6347

CountyUSA

Country Code100

Alias_Fields:

First NameJOHN

Middle NameA

Last NameMC

Phone_Fields:

Phone TypeHome


Phone6264700000

Phone Ext.0001

EditView EUID

Figure 3.2

Figure 3.3

master index edmuser Sign Out 

Master Index Data Manager

Dashboard Duplicate Records Record Details Assumed Matches Transactions Reports **Source Record** Audit Log

View/Edit Add Merge

< View Record List **Status:** active **Name Of The Source:** SystemA **Local ID:** 000000001

Person

- Address
- Alias
- Phone

Person_Fields:

Person Cat Code	Sun
First Name	JONATHAN
Middle Name	G
Last Name	MCNEALY
Suffix	code description
Title	code description
SSN	000000001
Date of Birth	08/08/1982
Death	
Gender	Male
Marital Status	Married
Race	White
Ethnic	code description
Religion	code description
Language	code description
Spouse Name	Emily
Mother Name	Emma
Mother Maiden Name	A
Father Name	William
Maiden	Emma
Pob City	MENLO PARK
Pob State	California
Pob Country	USA
VIP Flag	VIP
Veteran Status	Tenure
Status	Retired
Driver License	A0000001
Driver License St.	Validated
Date of Death	
Death Certificate	None
Nationality	code description
Citizenship	code description

Edit **View EUID**

Figure 3.4

master index data manager edmmuser Sign Out 

Dashboard Duplicate Records Record Details Assumed Matches Transactions Reports **Source Record** Audit Log

View/Edit Add Merge

< View Record List Status: active Name Of The Source: SystemA Local ID: 0000000001

 Person

-  Address
-  Alias
-  Phone

Address_Fields:

Address Type	code description
Address Line1	MON01
Address Line2	800 ROYAL OAKS
City	MONROVIA
State Code	CA
Postal Code	91016
Postal Code Ext.	6347
County	USA
Country Code	100

Edit View EUID

master index data manager edmmuser Sign Out 

Dashboard Duplicate Records Record Details Assumed Matches Transactions Reports **Source Record** Audit Log

View/Edit Add Merge

< View Record List Status: active Name Of The Source: SystemA Local ID: 0000000001

 Person

-  Address
-  Alias
-  Phone

Alias_Fields:

First Name	JOHN
Middle Name	A
Last Name	MC

Edit View EUID

master index data manager edmmuser Sign Out 

Dashboard Duplicate Records Record Details Assumed Matches Transactions Reports **Source Record** Audit Log

View/Edit Add Merge

< View Record List Status: active Name Of The Source: SystemA Local ID: 0000000001

 Person

-  Address
-  Alias
-  Phone

Phone_Fields:

Phone Type	Home
Phone	6264700000
Phone Ext.	0001

Edit View EUID

- **Record Details Page:** need to enhance research results pages for three search types:
Advanced Lookup(Phonetic)
Advanced Lookup(Alpha)
Simple Lookup

Need to redesign the pages by using tree structures with both vertical and horizontal scroll bars.

- **Duplicate Records Page:** need to enhance the search results pages for two types search:
Basic Search
Advanced Search

Need to redesign the pages by using tree structures with both vertical and horizontal scroll bars.

- **Assumed Matches Result Page.**

2.4.1.4 Initial Bulk Match And Load Tool

- Design time: extend the master index studio to generate the IBML zip package based on multi-tiered object definition.
- Runtime: extend the IBML runtime components to support multi-tiered object model.

2.4.1.5 Enhanced Components

Each components need to be investigated and ensure to support multi-tiered data structures and processing and the corresponding Junit tests need to be added.

- Index-core (open-dm-mi)
 - configurator, master controller ejb, filter, search, matching, epath, validation, ops, outbound, page, parser, query, querybuilder, report, survivor, update, webservice.
- index-gui (open-dm-mi)
 - project generator, ui application editor, wizards
- index-services (open-dm-mi)
 - configuration, control, common
- loader (open-dm-mi)
 - dataobject, epath, eindex, loader
- index-webapp (open-dm-mi)
 - presentation, jsp pages
- report (open-dm-mi)
 - report client

2.4.2 Data Quality

- Extend the mural standardization engine to support multi-tiered object model.
- Extend the mural match engine to support multi-tiered object model.
- Extend the mural data cleanser to support multi-tiered object model.
- Extend the mural data profiler to support multi-tiered object model.
- Each components need to be investigated and ensure to support multi-tiered data structures and processing.
 - cleanser (open-dm-dq)
 - data-analysis-binding-objects (open-dm-dq)

- data-analysis-processor (open-dm-dq)
- init (open-dm-dq)
- matcher (open-dm-dq)
- profiler (open-dm-dq)
- standardizer (open-dm-dq)

2.4.3 Data Integrator

- Extend the mural data integrator design time components to support multi-tiered object model.
- Extend the mural data integrator runtime components to support multi-tiered object model.
- TBD in a separate document by an assigned developer.

2.4.4 Data Mashup

- Extend the mural open data mashup services to support multi-tiered object model.
- TBD in a separate document by an assigned developer.

2.4.5 Data Migrator

- Extend the mural match engine to support multi-tiered object model.
- TBD in a separate document by an assigned developer.

2.4.6 Mural Solutions Integration

- The telecommunication subscriber solution based on multi-tiered object model.
- UK(USA, AUS) Patient solution

3 Deliverable Items

3.1 MDM V 2.1 Suite

3.2 Multi-Tiered Object Model Telecommunication Subscriber Solution

3.3 Multi-tiered Object Model Patient Solutions.

3.4 Updated MDM Documentation and Solution Documentations

4 User Stories and Plan of Attack

4.1 Junit Tests for Components

4.2 Jemmy/Jelly Tests for NetBeans

4.4 Telecommunication Subscriber Model Test

4.5 Patient Model Test

5 Issues and Discussions