

**i2b2**

**Software Documentation**

# **i2b2 Software Architecture**

## **Ontology Management (ONT) Cell**

# TABLE OF CONTENTS

- DOCUMENT MANAGEMENT ..... 4**
- ABSTRACT ..... 5**
- 1 OVERVIEW ..... 6**
  - 1.1 ONT DEFINITIONS, ACRONYMS AND ABBREVIATIONS ..... 7
    - 1.1.1 *Vocabulary Data Object (VDO)* ..... 7
    - 1.1.2 *Scheme* ..... 7
  - 1.2 ROLES ..... 8
  - 1.3 SECURITY ..... 9
  - 1.4 SCOPE OF THE SYSTEM ..... 9
  - 1.5 ASSUMPTIONS / CONSTRAINTS ..... 9
  - 1.6 TECHNICAL PLATFORM ..... 10
    - 1.6.1 *Transaction* ..... 10
    - 1.6.2 *Security* ..... 10
    - 1.6.3 *Persistence* ..... 11
    - 1.6.4 *Reliability / Availability* ..... 11
    - 1.6.5 *Performance* ..... 11
- 2 USE CASE ..... 12**
  - 2.1 OPERATIONS ..... 12
- 3 ARCHITECTURE DESCRIPTION ..... 14**
  - 3.1 COMPONENTS AND CONNECTOR VIEW ..... 14
    - 3.1.1 *Client-Server Style* ..... 14
      - 3.1.1.1 Primary Presentation ..... 14
      - 3.1.1.2 Element Catalog ..... 15
      - 3.1.1.3 Design Rationale, Constraints ..... 16
    - 3.2 MODULE VIEW TYPE ..... 16
      - 3.2.1 *Decomposition Style* ..... 16
        - 3.2.1.1 Primary Presentation ..... 16
        - 3.2.1.2 Element Catalog ..... 17
        - 3.2.1.3 Context Diagram ..... 17
      - 3.2.2 *Uses Style* ..... 17
        - 3.2.2.1 Primary Presentation ..... 17
        - 3.2.2.2 Element Catalog ..... 18
        - 3.2.2.3 Context Diagram ..... 18
        - 3.2.2.4 Sequence Diagram ..... 19
  - 3.3 MAPPINGS OF STYLES ..... 20
- 4 DATA VIEW ..... 21**
  - 4.1 SELECTING THE DATA SOURCE ..... 21
  - 4.2 SCHEMAS WITHIN THE METADATA DATA SOURCE ..... 23
    - 4.2.1 *TABLE\_ACCESS Table* ..... 23
    - 4.2.2 *Metadata Tables* ..... 26
    - 4.2.3 *Schemes Table* ..... 27
    - 4.2.4 *Process Status Table* ..... 28
- 5 DEPLOYMENT VIEW ..... 30**
  - 5.1 GLOBAL OVERVIEW ..... 30

5.2 DETAILED DEPLOYMENT MODEL ..... 31

**REFERENCES..... 32**

# DOCUMENT MANAGEMENT

Revision Number	Date	Author	Description of change
1.7.1	09/12/12	Janice Donahoe	Created 1.7 version of document.
1.7.2	10/23/13	Mike Mendis	Updated Jboss and Axis2
1.7.00-003	06/08/2015	Janice Donahoe	Fixed Document Version information that appears on the cover page. This was never updated to 1.7.2. It has now been updated to 1.7.00-003. Also fixed some minor spelling issues.
1.7.00-004	04/27/2016	Janice Donahoe	Fixed title page and some minor formatting. Content did not change.

## ABSTRACT

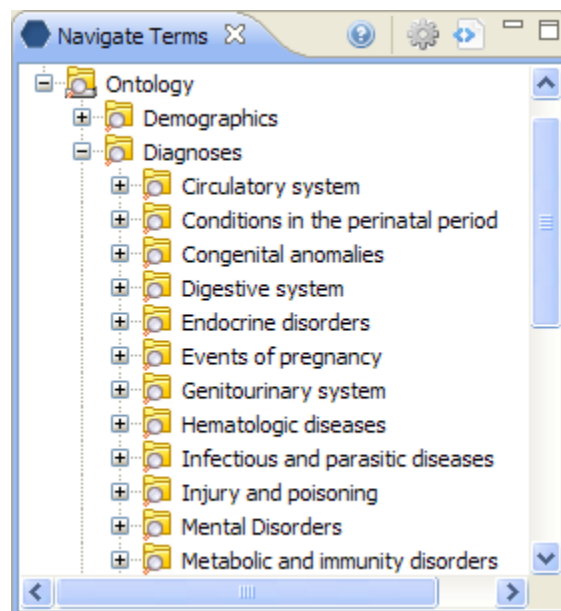
This is a software architecture document for the **Ontology Management (ONT) cell**. It identifies and explains the important architectural elements. This document will serve the needs of stake holders to understand the system concepts and give a brief summary of the use of the ONT message format.

# 1 OVERVIEW

The Ontology Management cell (ONT) is an i2b2 Hive core cell. This cell manages i2b2 vocabulary definitions and contains concepts and information about relationships between concepts for the entire hive. It is accessed by other cells to give semantic meaning to data.

Vocabularies in the ONT cell are organized in hierarchical structures that represent the relationship between terms. The top levels in the hierarchy are called the “**parents**” or “**roots**” with the lower levels being their “**children**”. Elements occurring on the same level are known as “**siblings**”. A level in a hierarchy is sometimes referred to as a “**node**” and a group of related data is called a “**category**”.

A category is defined as a set of data for which there is a common rule or rules for querying against the *Clinical Research Chart (CRC)*. A category is usually represented visually as a table of terms. An example of a category is the “Diagnoses” category show in the diagram below, which consists of a table of diagnostic terms and uses a single rule to build all diagnostic queries.

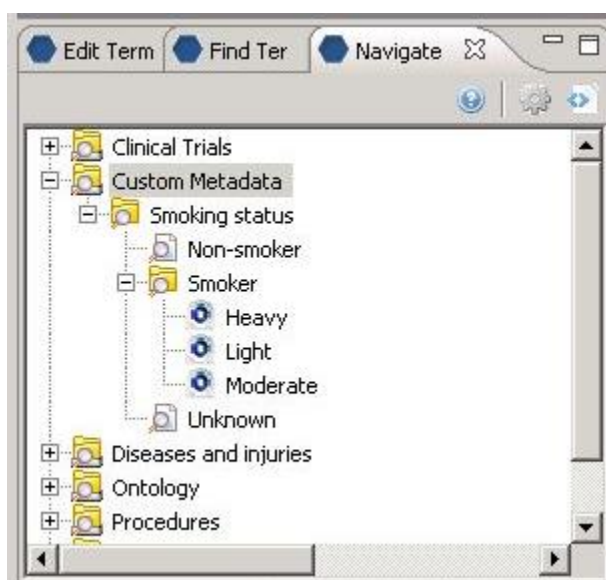


Vocabularies in the ONT cell may originate as code from different sources. The ONT cell distinguishes these codes from one another by pre-pending a unique prefix to each code. Each distinct vocabulary and their associated codes are called a **scheme**.

In release 1.6.00, we added **modifiers** to the ONT cell vocabulary. A modifier may be used to extend the meaning of a concept. As example of modifiers is shown below:

The concept “Smoker” has three modifiers that may be applied to it:

1. Heavy
2. Light
3. Moderate



## 1.1 ONT Definitions, Acronyms and Abbreviations

### 1.1.1 Vocabulary Data Object (VDO)

This object holds vocabulary definitions and information about the relationships between concepts.

### 1.1.2 Scheme

Each distinct vocabulary and their associated codes are called a scheme. A distinction is made between codes from different sources by pre-pending a unique prefix to each code.

## 1.2 Roles

When and how data is presented to a user is based on their user roles, which are specified in the PM Cell. Each user will have at least two roles per user\_id and product\_id combination. These two roles can be further defined as a *Data Protection role* and a *Hive Management role*.


The data protection role establishes the detail of data the user can see while the hive management role defines the level of functionality the user has in a project. The following tables summarize the roles in a hierarchical order of least to most access.

Data Protection Track	
Role	Access Description
DATA_OBFSC	<p>OBFSC = Obfuscated</p> <ul style="list-style-type: none"> <li>The user can see aggregated results that are obfuscated (example: patient count).</li> <li>The user is limited on the number of times they can run the same query within a specified time period. If the user exceeds the maximum number of times then their account will be locked and only the Admin user can unlock it.</li> </ul>
DATA_AGG	<p>AGG = Aggregated</p> <ul style="list-style-type: none"> <li>The user can see aggregated results like the patient count.</li> <li>The results are <u>not</u> obfuscated and the user is <u>not</u> limited to the number of times they can run the same query.</li> </ul>
DATA_LDS	<p>LDS = Limited Data Set</p> <ul style="list-style-type: none"> <li>The user can see all fields except for those that are encrypted.</li> <li>An example of an encrypted field is the <i>blob fields</i> in the <i>fact</i> and <i>dimension tables</i>.</li> </ul>
DATA_DEID	<p>DEID = De-identified Data</p> <ul style="list-style-type: none"> <li>The user can see all fields including those that are encrypted.</li> <li>An example of an encrypted field is the <i>blob fields</i> in the <i>fact</i> and <i>dimension tables</i>.</li> </ul>
DATA_PROT	<p>PROT = Protected</p> <ul style="list-style-type: none"> <li>The user can see all data, including the identified data that resides in the Identity Management Cell.</li> </ul>

Hive Management Track	
Role	Access Description
USER	Can create queries and access them if he / she is the owner of the query




MANAGER	Can create queries as well as access queries created by different users within the project
---------	--------------------------------------------------------------------------------------------

 **Additional Resources**

Further details regarding roles can be found in the *Project\_Management\_Design* document.

## 1.3 Security

Users can access the ONT with user-id and password combination, which is authenticated through the Project Management Cell. The implementation detail of the Project Management Cell is considered out-of-scope to this system context.

 **Additional Resources**

Further details about the implementation of the Project Management Cell can be found in the following documents:

- Project\_Management\_Architecture
- Project\_Management\_Design
- Project\_Management\_Messaging
- [i2b2 Installation Guide](#)

## 1.4 Scope of the system

Some other participants, currently outside the scope of the ONT are:

- Project Management Cell

## 1.5 Assumptions / Constraints

The ontology metadata database shall not contain protected health information.

## 1.6 Technical Platform

The product is designed to run on the following platform:

- Java 2 Standard Edition 7.0
- Oracle Server 10g/11g database
- SQL Server 2005/2008
- Xerces2 XML parser
- Jboss Application server version 7.1.1
- Spring Web Framework 2.0
- Axis2 1.6.2 web service (SOAP / REST)

### 1.6.1 Transaction

The ONT system is transactional, leveraging the transaction management model of the J2EE platform.

### 1.6.2 Security

The application must implement basic security behaviors:

Category	Behavior
Authentication	Authenticate using at least a user name and a password.
Authorization	User may only access categories that they are allowed to by role.
Confidentiality	Sensitive data must be encrypted.
Data Integrity	Data sent across the network cannot be modified by a tier.
Auditing	In the later releases we may implement logging of sensitive actions.

### **1.6.3 Persistence**

This application utilizes JDBC calls to retrieve persisted data.

### **1.6.4 Reliability / Availability**

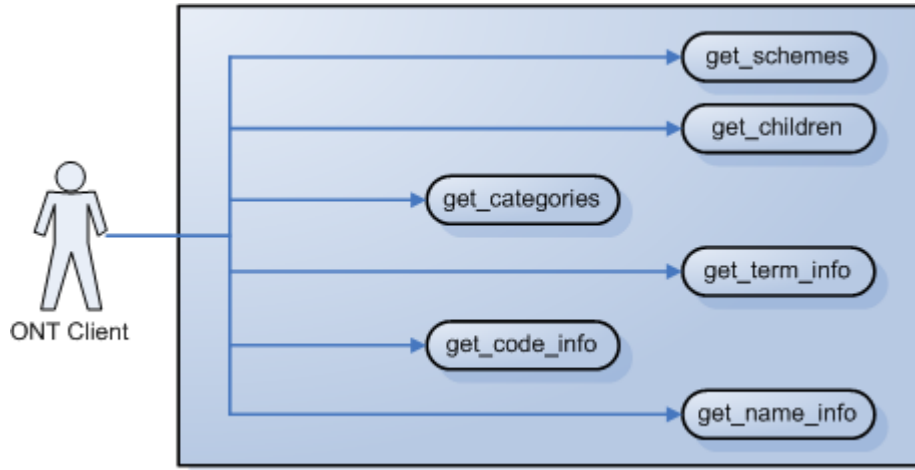
- The reliability / availability will be addressed through the J2EE platform
- Targeted availability is 16 / 7: 16 hours a day, 7 days a week
- The remaining time (8 hours) is reserved for any maintenance activities

### **1.6.5 Performance**

The user authentication with the project management cell must be under 1 second.

## 2 USE CASE

The diagram below depicts common use cases a user may perform with the ONT cell.



### 2.1 Operations

The ONT service is designed as a collection of operations or use cases:

Service Operation	Description
getCategories	Returns a list of categories available for a given user. These categories are displayed in a tree format. The top level of the tree consists of all the categories a particular user has permission to see.
getChildren	Expands any level of a vocabulary category, providing information about its children, for a given user.
getSchemes	Returns a list of schemes available in the system. This operation basically provides information about the different kinds of coding systems that exist.
getNameInfo	Returns information needed about all nodes related to a given search keyword or name.
getCodeInfo	Returns information about a code, such as the name associated with a particular code.
getTermInfo	Returns information about a particular node.
addChild	Adds a new ontology term to the tree under the selected parent node.

Service Operation	Description
deleteChild	Deletes a selected ontology term.
modifyChild	Modifies content within an existing ontology term.
updateCRCConcept	Notifies the ontology cell to synchronize metadataterms with concept_dimension table.
getProcessStatus	Returns status information about the concept_dimension synchronization process.
getDirtyState	Returns the state information about the need to synchronize with the concept_dimension table.
updateConceptTotalNum	Notifies the ontology cell to get the patient count from the CRC for this concept and then update the totalnum for this concept in the metadata table.
getModifiers	Returns a list of modifiers for a concept (if they exist).
getModifierChildren	Expands any level of a modifier folder, providing information about its children.
getModifierInfo	Returns information about a particular modifier.
getModifierNameInfo	Returns modifiers associated with a concept that meets the name search criteria.
getModifierCodeInfo	Returns modifiers associated with a concept that meets the code search criteria.
addModifier	Adds a new ontology modifier to the tree under the selected parent node or modifier.
excludeModifier	Excludes an existing modifier from a concept lower in the hierarchy than the modifier's specified hierarchy level.

## 3 ARCHITECTURE DESCRIPTION

This section provides a description of the architecture as multiple views. Each view conveys the different attributes of the architecture.

1. Components and Connector View
  - a. Client-Server Style
  
2. Module View
  - a. Decomposition Style
  - b. Uses Style
  
3. Data View
  
4. Deployment View

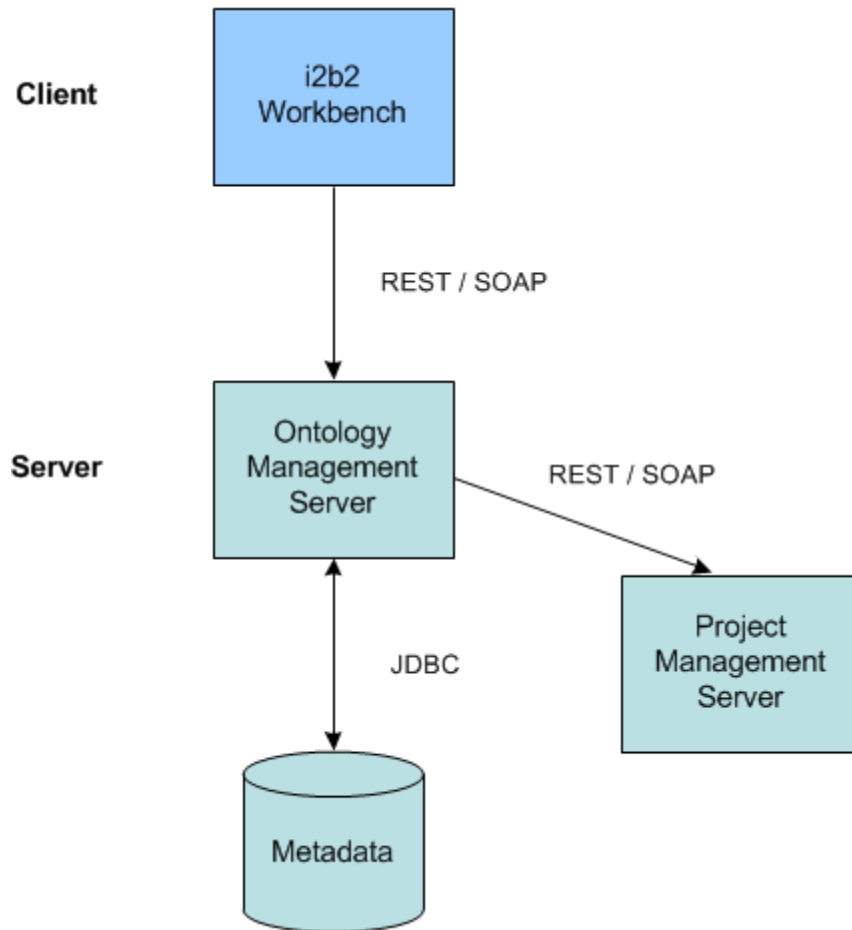
### 3.1 Components and Connector View

A **Component and Connector view** (C&C) represents the runtime instances and the protocols of connection between the instances. The connectors represent the properties such as concurrency, protocols and information flows. The diagram shown in the *Primary Presentation* section represents the Component and Connector view for the multi-user installation. As seen in the diagram, component instances are shown in more detail with specific connectors drawn in different notations.

#### 3.1.1 Client-Server Style

The ONT system is represented using the components and connector client-server view.

##### 3.1.1.1 Primary Presentation



### 3.1.1.2 Element Catalog

Element Name	Type	Description
i2b2 Workbench	Client Component	Webservice client submits the request to the ONT Server components and renders a response XML.
Ontology Management Server	Server Component	Provides a Web Service Interface for the ONT system. It supports the REST or SOAP protocol. It directs the user to the correct data source associated with the project. It uses the Project Management server to handle user authentication.
Project Management Server	Server Component	ONT cell uses the Project Management cell to authenticate the user. ONT cell constructs the PM request message and makes a web service call to the Project Management cell.

Element Name	Type	Description
Metadata	Data Repository Component	This repository is a database for the i2b2 metadata.
JDBC	Query Connector	SQL query used as a connector between the ONT System and the Metadata database.
Web Service	Request Connector	REST protocol used to communicate with the external system.

### 3.1.1.3 Design Rationale, Constraints

#### N-tier Architecture

The client-server style depicts an n-tier architecture that separates the presentation layer from the business logic and data access layer.

## 3.2 Module View Type

The module view shows how the system is decomposed into implementation units and how the functionality is allocated to these units. The layers show how modules are encapsulated and structured. The layers represent the “*allowed-to-use*” relation.

The following sections describe the module view using **Decomposition** and **Uses Styles**.

### 3.2.1 Decomposition Style

The Decomposition style presents system functionality in terms of manageable work pieces. It identifies modules and breaks them down into sub-modules and so on, until a desired level of granularity is achieved.

#### 3.2.1.1 Primary Presentation

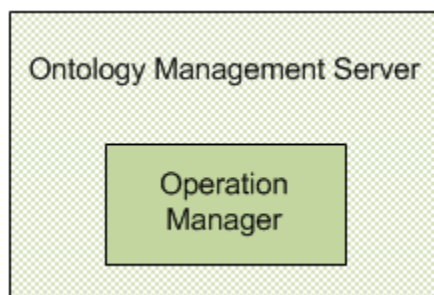
System	Segment
Ontology Management Server	Operation Manager



### 3.2.1.2 Element Catalog

Element Name	Type	Description
Operation Manager	Subsystem	This subsystem manages queries for ontology operations.

### 3.2.1.3 Context Diagram



## 3.2.2 Uses Style

The Uses style shows the relationship between modules and sub-modules. This view is very helpful for implementing, integrating and testing the system.

### 3.2.2.1 Primary Presentation

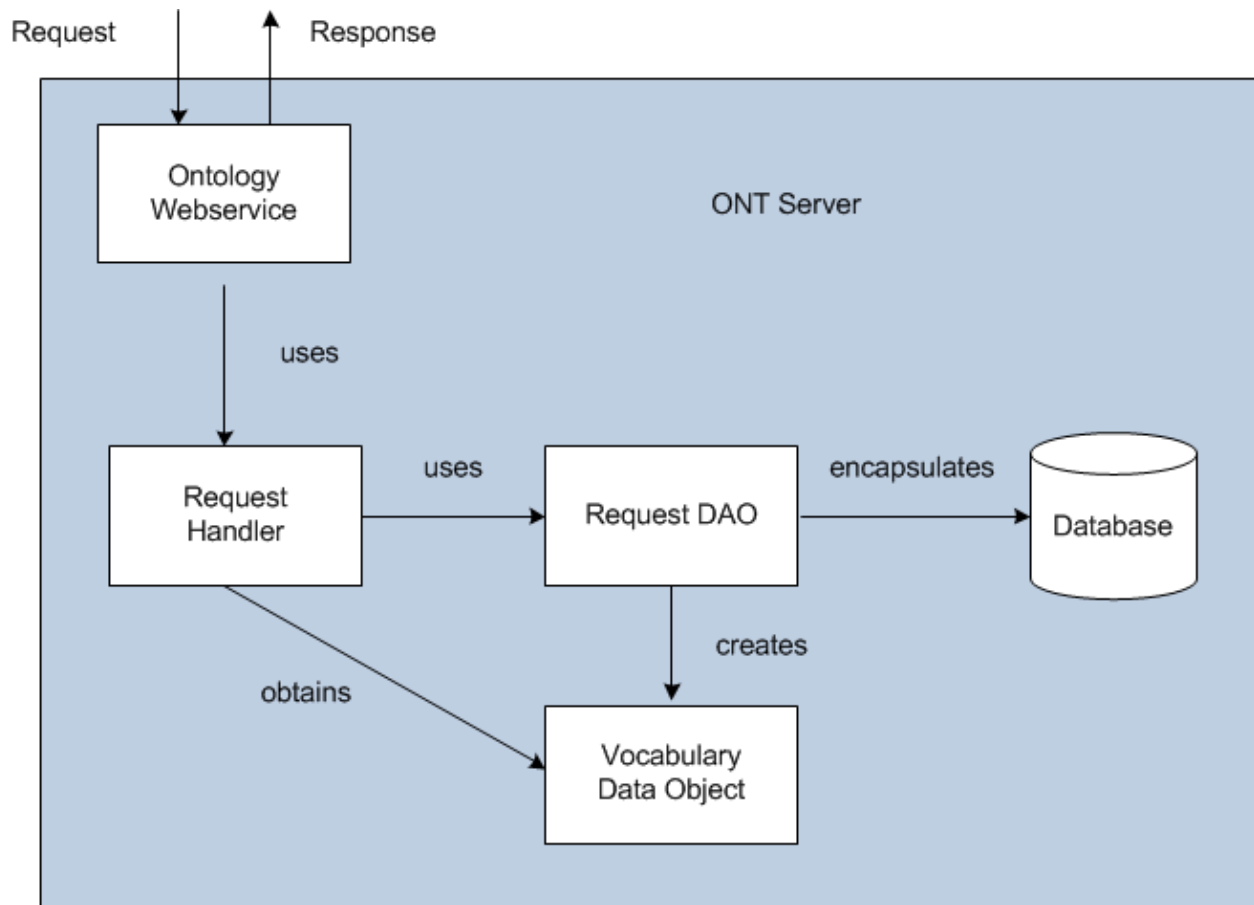
System	Segment
Ontology Management Server	ONT Module
Operation Manager Subsystem	Ontology Webservice Request Handler Request DAO

System	Segment
	Vocabulary Data Object

### 3.2.2.2 Element Catalog

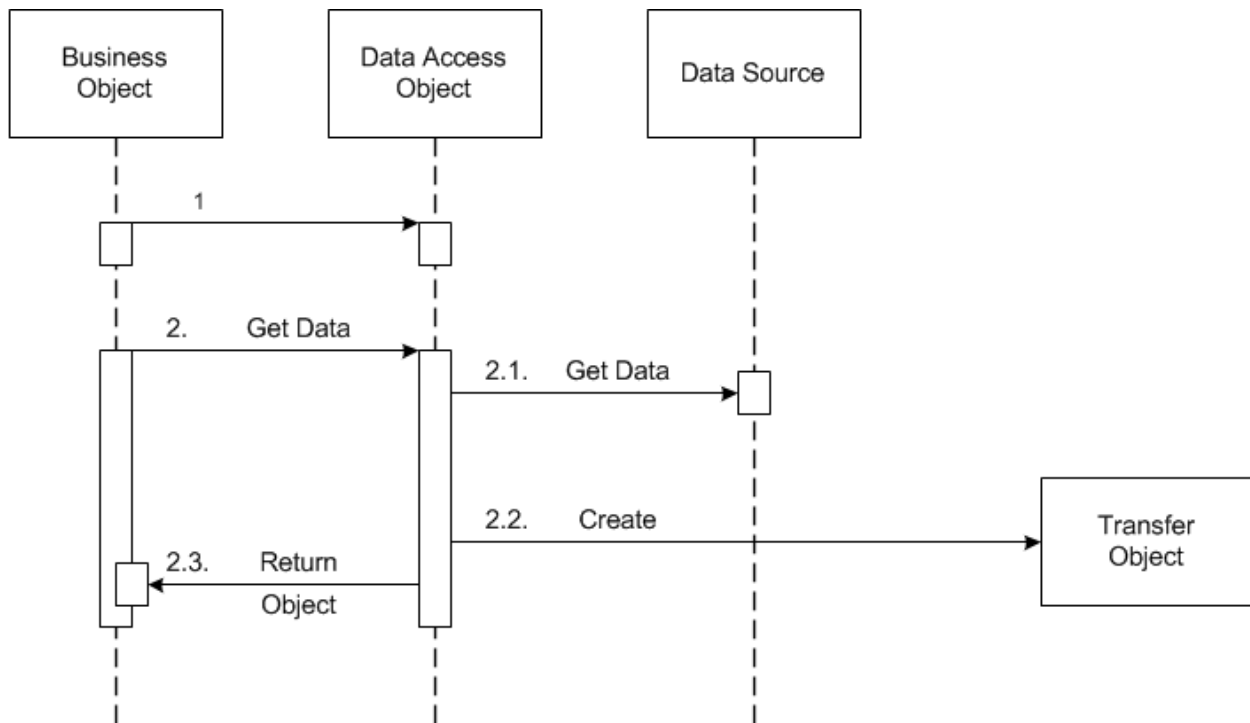
Element Name	Type	Description
ONT Module	Module	Authenticates the user through the PM Server System.
Ontology Webservice	Communication Module	Provides web service interface to ontology operations.
Request Handler	Business Object	Delegates Ontology requests to Data Access Object layer to perform database operations.
Request DAO	Data Access Object	Supports database query operations.
Vocabulary Data Object	Transfer Object	Object representation of persisted data.

### 3.2.2.3 Context Diagram



The Request Handler determines which **Persistent Storage Location (PSL)** is associated with a project request.

### 3.2.2.4 Sequence Diagram



### 3.3 Mappings of Styles

The following table is a mapping between the elements in the *Component & Connector Client-Server view* shown in section 3.1.1, and the *Modules Uses view* and *Decomposition view* shown in sections 3.2.1 and 3.2.2.

The relationship shown is ***is-implemented-by***, i.e. the elements from the C&C view shown at the top of the table are implemented by any selected elements from the Modules views, denoted by and “X” in the corresponding cell.

	ONT Server	Project Management Server	Metadata Database
<b>ONT Service</b>	X	X	
<b>Ontology Webservice</b>	X		
<b>Request Handler</b>	X		
<b>Request DAO</b>	X		X
<b>Vocabulary Data Object</b>	X		

## 4 DATA VIEW

### 4.1 Selecting the Data Source

Both the metadata and the patient data are distributed to projects through the existence of independent databases (in SQL Server) or schemas (in Oracle). These will be referred to in the rest of the document as the “**persistent storage location**” or **PSL**. These PSL’s are organized so that the data from two metadata representations can be merged to a “Super” data set. While a person is working on a specific project, they will be directed to data in a PSL associated with that project.

In order to support the i2b2 project distribution strategy, the user is enrolled in numerous projects recorded within the i2b2 project management cell. The projects available to the user are returned in the web service call to the Project Management cell. The logic of selecting the correct PSL for the project is embodied in the following table.

DB_LOOKUP		
PK	C_DOMAIN_ID	VARCHAR(255)
PK	C_PROJECT_PATH	VARCHAR(255)
PK	C_OWNER_ID	VARCHAR(255)
	C_DB_FULLSCHEMA	VARCHAR(255)
	C_DB_DATASOURCE	VARCHAR(255)
	C_DB_SERVERTYPE	VARCHAR(255)
	C_DB_NICENAME	VARCHAR(255)
	C_DB_TOOLTIP	VARCHAR(255)
	C_COMMENT	CLOB
	C_ENTRY_DATE	DATE
	C_CHANGE_DATE	DATE
	C_STATUS_CD	CHAR(1)

The logic for selecting the PSL is as follows:

1. There are two methods to select the correct PSL, an implicit one, and an explicit one. Both rely only on information available within the i2b2 header.
  - a. The implicit one relies upon the data within the <domain> tag, the <username> tag, and the <project\_id> tag.
  - b. The explicit one relies upon the data only within the <project\_id> tag. It has the format represented as the following string:

|"DOMAIN" | "PROJECT" \ "sub-project" \ "sub-sub-project"\ | "USER\_ID"|

**Note**

These may not actually match the domain and username that is actually being used (since it is being built by the client), and must be checked when the PM cell is accessed.

2. The table is meant to provide a series of default locations if ones are not specifically listed. If a project is listed in the *C\_PROJECT\_PATH* column, then that PSL may be used, otherwise a domain source will be used.
3. If a username is listed in the *C\_OWNER\_ID* column, and the project also matches the *PROJECT\_ID*, the PSL in that row may be used otherwise a project PSL will be used. If the project PSL does not exist, the domain PSL will be used.

For example, only if the *domain \ project \ user\_id* is an EXACT match to the entries in the database will that PSL be used.

4. The project id may have associated sub-projects that will be represented as *project \ sub-project \ sub-sub-project* string. If a sub-project is identified but only the project exists in the table then the project PSL would be used.
5. The project may not have an entry in the table and in that case any project (and sub-projects) would be designated the PSL of the domain.
6. If a general domain PSL is not available in the table and only a specific project is associated with the domain in the table, then any incoming messages not associated with that project will return an error.
7. In the table, the "@" character is used to represent the absence of an entry (rather than a blank or a null).

8. In the explicit string and in the <project\_id> a “@” can be used to optionally represent a blank column.

Other columns are specified as follows:

1. The column *C\_DB\_FULLSCHEMA* is used to contain the path to a table when the data source is used. Software is written so that the absence of the delimiter (usually a “.”) does not need to be explicitly stated.
2. The column *C\_DB\_DATASOURCE* is used to contain a short string that represents a data source configured in some other location.
3. The column *C\_DB\_SERVERTYPE* can be “ORACLE” or “SQLSERVER”.
4. The column *C\_DB\_NICENAME* is a string that can be used in the client software to describe a data source.
5. The column *C\_DB\_TOOLTIP* contains a longer (hierarchical) representation of the nice name.

To restate, many cells need to access some kind of persistent storage, and these cells will organize their persistent storage so that it is self-contained and can be apportioned in a way consistent with the project-based requirements of i2b2 that are described above. To that end, a table exists in many cells to make the decision of what persistent storage location to which a specific user will be directed, depending on the project and domain to which they are associated.

## 4.2 Schemas within the Metadata Data Source

The following schemas provide data used by the ONT system.

### 4.2.1 TABLE\_ACCESS Table

This table is used to obtain a list of root elements of the Metadata tree. Each root element of the tree is represented by a single row in this table. The primary identifier of each Metadata root element in the **TABLE\_ACCESS** table is in the *C\_TABLE\_CD* column. All messages that need to point to a specific Metadata root element will use this identifier. For example, in the <key> element of many messages this identifier is represented as `\\C_TABLE_NAME\` (see messaging

specification). Within the *C\_TABLE\_NAME* column is the actual name in the PSL of the metadata table. The *C\_PROTECTED\_ACCESS* Boolean column designates whether one must have the protected access role (*DATA\_PROT*) to obtain data from this table. The other columns are defined in a similar manner to the columns of the Metadata table, with the following special notes:

The *C\_DIMCODE* is used to allow the entire contents of a table to be queried in the data repository if the dimension table of the data repository has been set up in this fashion. While the *C\_HLEVEL* is traditionally 0 in this table (indicating one metadata table for each root element of the metadata tree), it is possible to “flatten” the metadata tree so that multiple entries of a single metadata table appear as root nodes (see examples shown below).

<b>TABLE_ACCESS</b>	
<b>C_TABLE_CD</b>	<b>VARCHAR(50)</b>
<b>C_TABLE_NAME</b>	<b>VARCHAR(50)</b>
C_PROTECTED_ACCESS	CHAR(1)
<b>C_HLEVEL</b>	<b>INT</b>
<b>C_FULLNAME</b>	<b>VARCHAR(700)</b>
<b>C_NAME</b>	<b>VARCHAR(2000)</b>
<b>C_SYNONYM_CD</b>	<b>CHAR(1)</b>
<b>C_VISUALATTRIBUTES</b>	<b>CHAR(3)</b>
C_TOTAL_NUM	INT
C_BASECODE	VARCHAR(50)
C_METADATAXML	CLOB
<b>C_FACTTABLECOLUMN</b>	<b>VARCHAR(50)</b>
<b>C_DIMTABLENAME</b>	<b>VARCHAR(50)</b>
<b>C_COLUMNNAME</b>	<b>VARCHAR(50)</b>
<b>C_COLUMNDATATYPE</b>	<b>VARCHAR(50)</b>
<b>C_OPERATOR</b>	<b>VARCHAR(10)</b>
<b>C_DIMCODE</b>	<b>VARCHAR(700)</b>
C_COMMENT	CLOB
C_TOOLTIP	VARCHAR(900)



TABLE_ACCESS		
	C_ENTRY_DATE	DATE
	C_CHANGE_DATE	DATE
	C_STATUS_CD	CHAR(1)
	VALUETYPE_CD	VARCHAR(50)

A traditional TABLE\_ACCESS table is shown below: it has one entry for each table.

	TABLE_CD	C_TABLE_NAME	C_HLEVEL	C_NAME
1	BIRN	BIRN	0	Clinical Trials
2	i2b2	i2b2	0	Ontology
3	CUST	CUSTOM_META	0	Custom Metadata

The following TABLE\_ACCESS table shows entries for multiple root elements from a single metadata table. The nine i2b2 elements shown here appear under the “Ontology” root element in the above example, resulting in a “flattened” hierarchy.

	TABLE_CD	C_TABLE_NAME	C_HLEVEL	C_NAME
1	BIRN	BIRN	0	Clinical Trials
2	CUST	CUSTOM_META	0	Custom Metadata
3	i2b2_DEMO	I2B2	1	Demographics
4	i2b2_DIAG	I2B2	1	Diagnoses
5	i2b2_EXPR	I2B2	1	Expression Profiles Data
6	i2b2_LABS	I2B2	1	Laboratory Tests
7	i2b2_MEDS	I2B2	1	Medications

	TABLE_CD	C_TABLE_NAME	C_HLEVEL	C_NAME
8	i2b2_PROC	I2B2	1	Procedures
9	i2b2_PROV	I2B2	1	Providers
10	i2b2_REP	I2B2	1	Reports
11	i2b2_VISIT	I2B2	1	Visit Details

## 4.2.2 Metadata Tables

The **Metadata** table encapsulates the vocabulary used in the data repository. A **term** (concept or provider) is a row from the metadata table. It is the primary object used to pass vocabulary information to the requesting client. Typically a PSL will contain numerous metadata tables, each with a name that indicates the domain that the vocabulary contained within represents.

METADATA		
<b>PK</b>	<b>C_FULLNAME</b>	<b>VARCHAR(700)</b>
<b>PK</b>	<b>C_NAME</b>	<b>VARCHAR(2000)</b>
	<b>C_HLEVEL</b>	<b>INT</b>
	<b>C_SYNONYM_CD</b>	<b>CHAR(1)</b>
	<b>C_VISUALATTRIBUTES</b>	<b>CHAR(3)</b>
	C_TOTAL_NUM	INT
	C_BASECODE	VARCHAR(50)
	C_METADATAXML	CLOB
	<b>C_FACTTABLECOLUMN</b>	<b>VARCHAR(50)</b>
	<b>C_TABLENAME</b>	<b>VARCHAR(50)</b>
	<b>C_COLUMNNAME</b>	<b>VARCHAR(50)</b>
	<b>C_COLUMNDATATYPE</b>	<b>VARCHAR(50)</b>
	<b>C_OPERATOR</b>	<b>VARCHAR(10)</b>
	<b>C_DIMCODE</b>	<b>VARCHAR(700)</b>
	C_COMMENT	CLOB

METADATA		
	C_TOOLTIP	VARCHAR(900)
	<b>M_APPLIED_PATH</b>	<b>VARCHAR(700)</b>
	<b>UPDATE_DATE</b>	<b>DATE</b>
	DOWNLOAD_DATE	DATE
	IMPORT_DATE	DATE
	SOURCESYSTEM_CD	VARCHAR(50)
	VALUETYPE_CD	VARCHAR(50)
	M_EXCLUSION_CD	VARCHAR(25)
	C_PATH	VARCHAR(700)
	C_SYMBOL	VARCHAR(50)

Also included in this table are modifiers associated with the terms contained within the metadata table. The column *M\_APPLIED\_PATH* specifies the concept path or *C\_FULLNAME* of the concept the modifier is associated with. If a modifier applies to a concept and its descendants the value in the *M\_APPLIED\_PATH* is appended with a “%”. Entries in the metadata table that are not modifiers should have the *M\_APPLIED\_PATH* set to “@”.

Additionally, it is possible to specify concept(s) a modifier is not associated with. Assume a modifier has been assigned to a root concept and its descendants with an applied path of *Diagnoses\%*. To exclude this modifier from a child (“Mental Disorders”) of the root concept (“Diagnoses”) we add an entry for the modifier with an applied path of *Diagnoses\Mental disorders\* and set the *EXCLUSION\_CD* to “X”. If we had wished to exclude this modifier from the descendants of “Mental disorders” we would append the applied path with a “%”.

Finally in Release 1.6.00 we added two optional columns in support of external tools. The column *C\_PATH* contains the *C\_FULLNAME* of a node’s parent. The column *C\_SYMBOL* is a unique, abbreviated form of the node’s *C\_NAME*. A node’s *C\_PATH*, concatenated with its *C\_SYMBOL* form the node’s *C\_FULLNAME*.

### 4.2.3 Schemes Table

The Schemes schema contains the unique prefixes obtained by different source codes. For example, codes from the National Drug Code are prepended with the “NDC” prefix, while codes

from the United Medical Language System are prepended with the “UMLS” prefix. This schema contains all the schemes recognized by the ONT system.

SCHEMES		
<b>PK</b>	<b>C_KEY</b>	<b>VARCHAR(50)</b>
	<b>C_NAME</b>	<b>VARCHAR(5)</b>
	C_DESCRIPTION	VARCHAR(100)

An example of a Schemes table is shown below:

	<b>C_KEY</b>	<b>C_NAME</b>	<b>C_DESCRIPTION</b>
1	NDC:	NDC	National Drug Code
2	DSG-NLP:	DSG-NLP	Natural Language Processing Data
3	UMLS:	UMLS	United Medical Language System
4	LCS-I2B2:	LCS-I2B2	Local coding system for NLP results
5	ICD9:	ICD9	ICD9 code for diagnoses and procedures
6	LOINC:	LOINC	Lab codes

## 4.2.4 Process Status Table

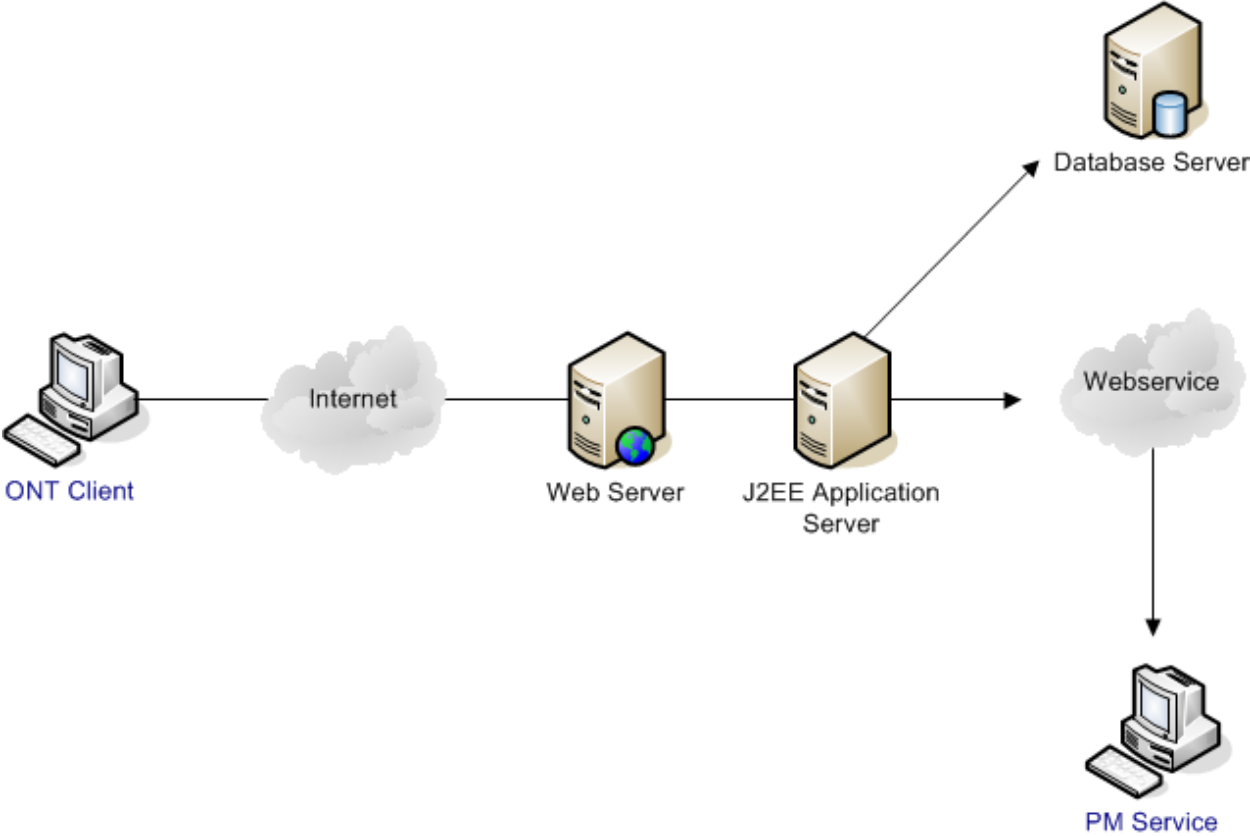
Release 1.5 introduced the ability to create new metadata or edit existing metadata within the i2b2 workbench. Once metadata has been created or edited it is necessary to synchronize these changes with the appropriate dimension table. The CONCEPT\_DIMENSION table would need to be updated with the new concept term; likewise the PROVIDER\_DIMENSION table would need to be updated with new provider terms.

The Process Status table provides information about the synchronization between the ontology metadata tables and the dimension tables.

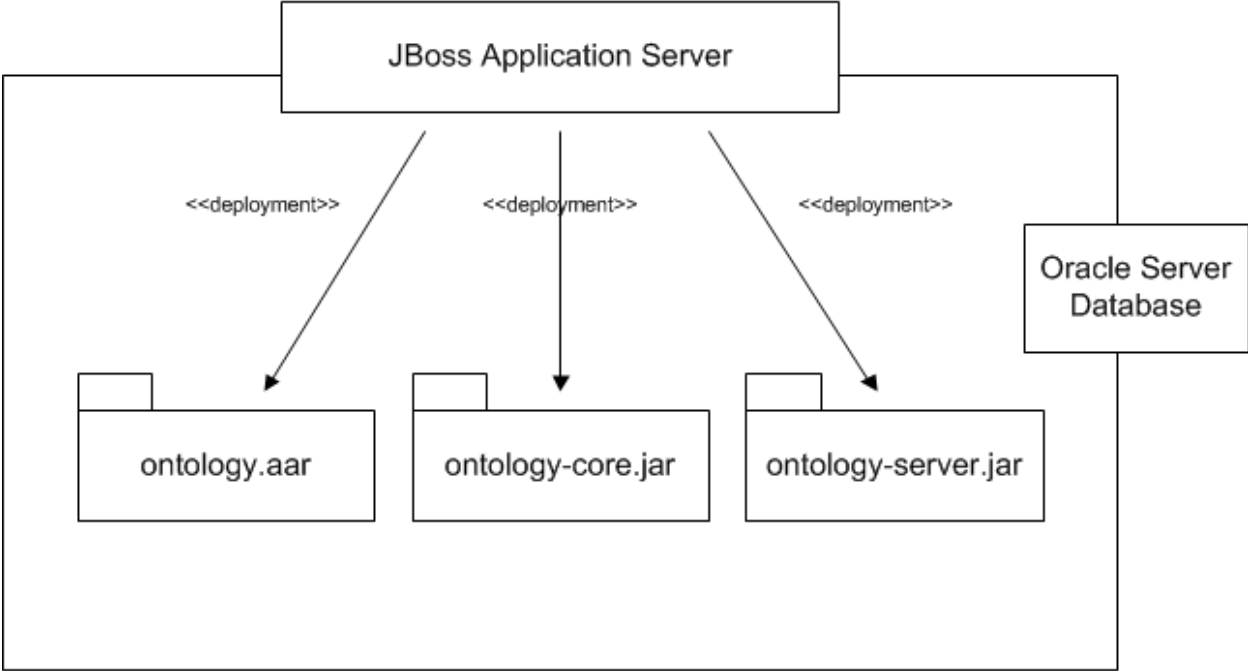
PROCESS STATUS		
PK	PROCESS_ID	INT
	PROCESS_TYPE_CD	VARCHAR(50)
	START_DATE	DATE
	END_DATE	DATE
	PROCESS_STEP_CD	VARCHAR(50)
	PROCESS_STATUS_CD	VARCHAR(50)
	CRC_UPLOAD_ID	VARCHAR(5)
	STATUS_CD	VARCHAR(50)
	MESSAGE	VARCHAR(2000)
	ENTRY_DATE	DATE
	CHANGE_DATE	DATE
	CHANGEDBY_CHAR	CHAR(50)

# 5 DEPLOYMENT VIEW

## 5.1 Global Overview



## 5.2 Detailed Deployment Model



## REFERENCES

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R. and Stafford, J., *Documenting Software Architectures: Views and Beyond*. (Boston, MA: Addison-Wesley, 2003)

Philippe Kruchten, “Architectural Blueprints – The “4+1” View Model of Software Architecture, <http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf> (*IEEE Software* 12 (6), November 1996)

“Object Management Group UML 2.0 Specification”, <http://www.omg.org/technology/documents/formal/uml.htm> (Object Management Group)

i2b2 (Informatics for Integrating Biology and the Bedside) <https://www.i2b2.org/resrcs/hive.html>