Informatics for Integrating Biology and the Bedside

i2b2 Cell Messaging

Ontology Management (ONT) Cell

Document Version: *1.7.08-006*

i2b2 Software Version: *1.7.08*

TABLE OF CONTENTS

DOCUMENT MANAGEMENT

Revision Number	Date	Author	Description of change
1.7.1	10/25/12	Janice Donahoe	Created 1.7 version of the document.
1.7.00-002	08/12/2015	Janice Donahoe	Fixed spelling and grammar errors.
1.7.07-003	01/05/2016	Janice Donahoe	In section 2.2.15.1 (Return Information Associated with a Modifier), updated the criteria used when determining which modifier(s) to be returned. This change was made in the 1.7.07 Software.
1.7.08-004	07/26/2016	S. Wayne Chan	Added 4 DBlookup Messages. Corrected minor subsection header levels.
1.7.08-005	08/18/2016	Lori Phillips	Added loadMetadata message
1.7.08-006	10/06/2016	Janice Donahoe	Updated the details of the new Dblookup messages.

1 INTRODUCTION

This document gives an overview of i2b2 cell messaging as well as a more detailed description of message formats specific to the **Ontology Management (ONT) Cell**.

1.1 The i2b2 Hive

Informatics for Integrating Biology and the Bedside (i2b2) is one of the sponsored initiatives of the NIH Roadmap National Centers for Biomedical Computing (http://www.bist.nigh.gov/ncbc/). One of the goals of i2b2 is to produce a comprehensive set of software tools to enable clinical investigators to collect and manage their project related research data, including clinical and genomic data; that is, a software suite for the modern clinical research chart. Since different applications from different sources must be able to communicate with each other, a distributed computing model is needed, one that integrates multiple web-based applications in a standardized way.

The i2b2 hive and associated web services are the infrastructure used to create this integration. The hive is comprised of a collection of cells representing unique functional units. Cells in the hive have an array of roles, such as data storage, data analysis, ontology or identity management, natural language processing, and data conversion, derivation or de-identification. Each cell is a self-contained modular application that communicates with other cells via XML web services. A common i2b2 messaging protocol has been defined to enable the cells to interact with each other, sharing business logic, processes and data.

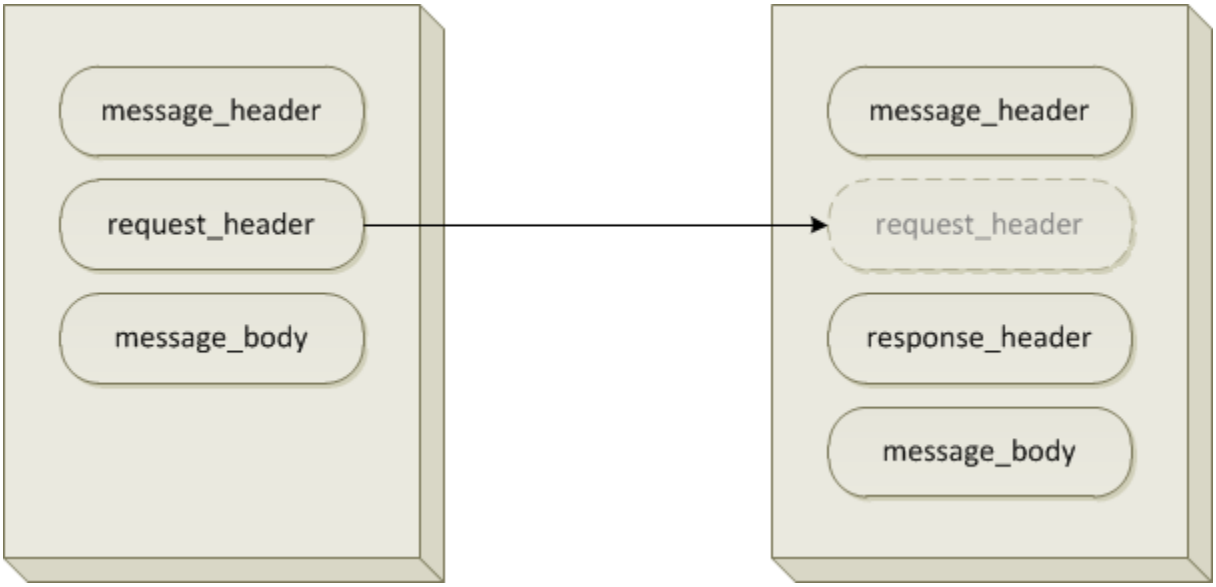
1.2 i2b2 Messaging Overview

All cells in the **i2b2 hive** must communicate using standard *i2b2 XML messages*. This message specifies certain properties that are common to all cells and are essential to the administration tasks associated with sending, receiving and processing messages.

A ***request message*** is sent from a client to a service and contains information inside the top level <*request*> tag that allows the service to satisfy the request. The <*request*> tag contains a <*message_header*>, <*request_header*> and <*message_body*>.

The service sends back a ***response message***, inside a top-level <*response*> tag, which informs the client about the status of the request and may also contain the actual results. The <*response*> tag contains its own <*message_header*>, <*response_header*> and <*message_body*> and it may optionally echo the request's <*request_header*>.

The following image illustrates the basic top-level elements contained within the request and response messages.



1.2.1 Message Header

All requests are sent using a *<request>* tag and responses are returned using a *<response>* tag. The same *<message_header>* tag is used for both. Both the request and response message contain this *<message_header>* tag which has control information such as the sending application, receiving application and the message type.

1.2.2 Request Header

The request must contain a *<request_header>* tag which includes information about how to process a request such as the amount of time it is willing to wait for a response. The *<request_header>* tag may optionally be echoed back in the response.

1.2.3 Response Header

The response must include a *<response_header>* tag which includes general information about the response such as status and error messages or where to look for the results if they are not included with the response.

1.2.4 Message Body

Both the request and response messages contain a <*message_body*> tag which will contain any well-formed xml. Individual cells may define cell-specific XML that will be put inside the <*message_body*> tag. This cell-specific XML does not need to extend the i2b2 message schema since the i2b2 schema will allow insertion of tags from any namespace into the <*message_body*> tag.

1.3 i2b2 XML Schema Definitions

The i2b2 XML schema consists of three XSD files.

1.3.1 i2b2.xsd

This schema defines the type for the <*message_header*> and <*message_body*> tags. This schema is included in the *i2b2_request.xsd* and the *i2b2_response.xsd*.

1.3.2 i2b2_request.xsd

This schema defines the type for the top-level <*request*> tag and the <*request_header*> tag. It is used for validating i2b2 request messages.

1.3.3 i2b2_response.xsd

This schema defines the type for the top-level <*response*> tag and the <*response_header*> tag. It is used for validating i2b2 response messages.

Additional Resources

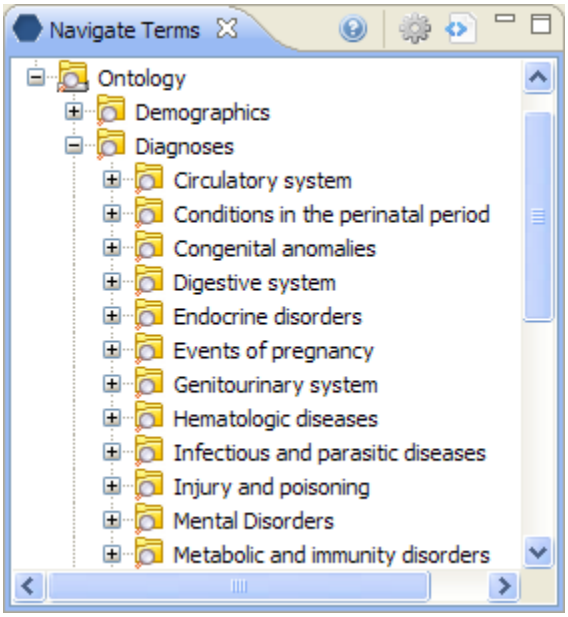
Additional details about the <*request*>, <*response*>, <*message_header*>, <*request_header*>, and <*response_header*> tags can be found in a separate document describing the generic i2b2 message. The remainder of this document describes the contents of the <*message_body*> for the Ontology Management (ONT) Cell.

2 ONTOLOGY MANAGEMENT (ONT) CELL MESSAGING DETAIL

Definitions for the i2b2 vocabularies reside in the Ontology Management (ONT) Cell. This cell 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”.

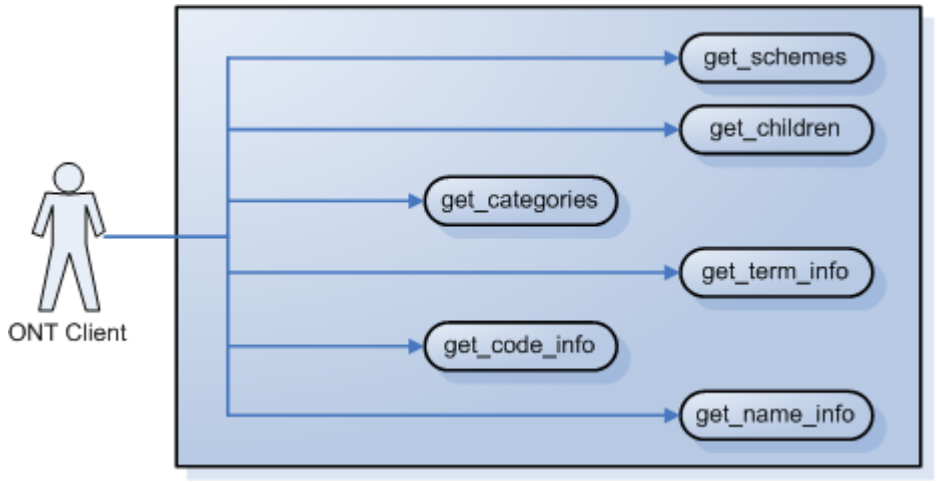
A category is defined as a set of data for which there is a common rule or rules for querying against the Clinical Research Chart (CRC). A category is usually represented visually as a table of terms. An example of a category is the Diagnosis category shown in the below image, which consists of a table of diagnoses terms and uses a single rule to build all diagnosis queries.



Vocabularies in the ONT cell may originate from different sources; the codes from each source are distinguished from the others by a unique prefix that is appended to the source code. Each distinct vocabulary and their associated codes are called a **scheme**.

2.1 Use Case

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



2.1.1 Operations

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

Service	Description
get_categories	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.
get_children	Expands any level of a vocabulary category, providing information about its children, for a given user.
get_schemes	Returns a list of schemes available in the system. This operation basically provides information about the different kinds of coding systems that exist.
get_name_info	Returns information needed about all nodes related to a given search keyword or name.
get_code_info	Returns information about a code, such as the name associated with a particular code.
get_term_info	Returns information about a particular node.
add_child	Adds a child term to a specified editable parent node.
delete_child	Deletes an editable tree node.

modify_child	Modifies content within and existing ontology term.
update_crc_concept	Notifies the Ontology cell to synchronize metadata terms with a dimension table.
get_ont_process_status	Returns the status information about the dimension table synchronization process.
get_dirty_state	Returns the state information about the need to synchronize with the CONCEPT_DIMENSION table.
update_concept_totalnum	Notifies the Ontology cell to get the patient count from the CRC for this concept and then updates the TOTALNUM for this concept in the metadata table.
get_modifiers	Returns a list of modifiers for a concept if they exist.
get_modifier_children	Expands any level of a modifier folder, providing information about its children.
get_modifier_info	Returns information about a particular modifier.
get_all_dblookups	Returns a list of all the entries (rows) in the ONT_DB_LOOKUP table.
set_dblookup	Adds or updates a specific entry (row) in the ONT_DB_LOOKUP table.
get_dblookup	Returns all the data for a specific entry (row) in the ONT_DB_LOOKUP table.
delete_dblookup	Deletes a specific entry (row) in the ONT_DB_LOOKUP database table.
load_metadata	Loads metadata to a specified table

Note

In your i2b2 Database the **ONT_DB_LOOKUP** table is part of the **I2B2HIVE** schema (*I2B2HIVE.ONT_DB_LOOKUP*).

2.1.2 Database Lookup Services / Messages

The following four services and messages that will be invoked when querying the **ONT_DB_LOOKUP** table in the *I2B2Hive* schema of your i2b2 database.

Service	XML Message
OntologyService/**getAllDblookups**	get_all_dblookups
OntologyService/**setDblookup**	set_dblookup

OntologyService/**getDblookup**	get_dblookup
OntologyService/**deleteDblookup**	delete_dblookup

The details for each of these messages is covered in the subsequent sections.

2.1.2.1 Restrictions

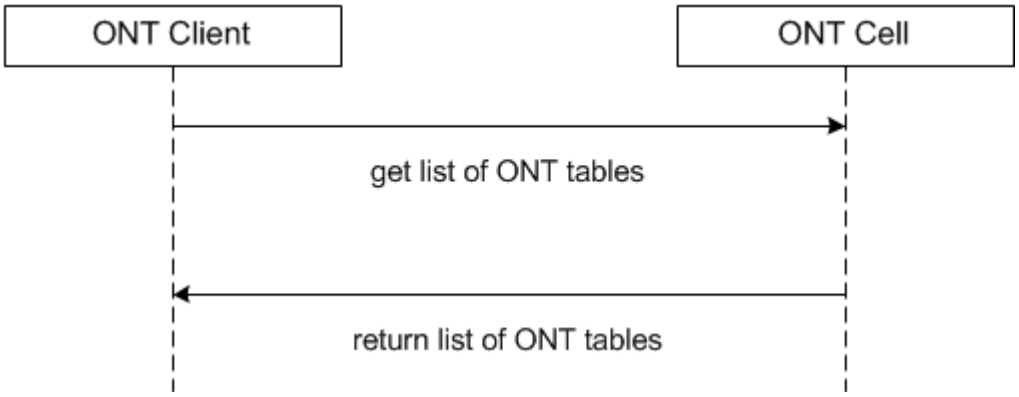
The role of **ADMIN** is required when running any of the *DBLookup messages*. The Ontology Cell with verify with the PM Cell that user is an ADMIN and if they are not then an error message will be returned in the response message.

2.2 Messages

2.2.1 get_categories

A **get_categories** message returns a *role-specific* list of categories that will be displayed as the roots of the Navigate Terms query tree and a list of categories in the Find Terms tool. No other information needs to be passed to the service.

User information is provided in the <*message_header*>; roles will be provided by the **Project Management (PM) cell**.



2.2.1.1 Get a List of Categories Associated with a User

The **get_categories** message is sent by the query tool to populate the root nodes and by the Find Terms tool to populate the list of categories available to a user.

The sequence of events is as follows (assumes max is not an issue)

1. The client requests a list of categories for a given user (get_categories)

   Request type = default for Find Terms; core for Navigate Terms

2. The ONT server performs the following steps:

   a. Get a list of roles available for this user from the PM cell (this also serves to validate the user)

   b. Query the table of tables for the list of categories associated with this user's roles.

3. The client maps the list of tables to the appropriate usage, either to Navigate Terms root node or Find Terms category list.

2.2.1.2 GET_CATEGORIES Request Message

```
<message_body>
    <get_categories type="default" blob="false" "hiddens="true"="" synonyms="false"/>
</message_body>
```

2.2.1.2.1 Possible "type" Settings

Value	Description	Example
default	Return table name / key pairs	Find Terms request

core	Return all data except system / date information	Navigate Terms request

2.2.1.2.2 Possible “blob” Settings

Value	Description	Example
false	Do not return data stored as a blob or clob	xml, comments
true	Return xml and comments	

2.2.1.2.3 Possible “hiddens” Settings

Some categories exist but for various reasons are not displayed in the query tree.

Value	Description
false	Do not return hidden categories
true	Include hidden categories

2.2.1.2.4 Possible “synonyms” Settings

Some categories may be listed as synonyms for others.

Value	Description
false	Do not return data categorized as “synonym”
true	Include data categorized as “synonym”

2.2.1.3 get_categories Response Message

2.2.1.3.1 Response Message for a Find Terms Request

The request has the following settings:

type=default

blob=false

hiddens=false

synonyms=false

Response Message:

```
<message_body>
    <concepts>
        <concept>
            <key>\\i2b2\RPDR\Diagnoses</key>
            <name>Diagnoses</name>
        </concept>
    </concepts>
</message_body>
```

2.2.1.3.2 Response Message for a Navigate Terms Request

The request has the following settings:

type=core

blob=true

hiddens=false

synonyms=false

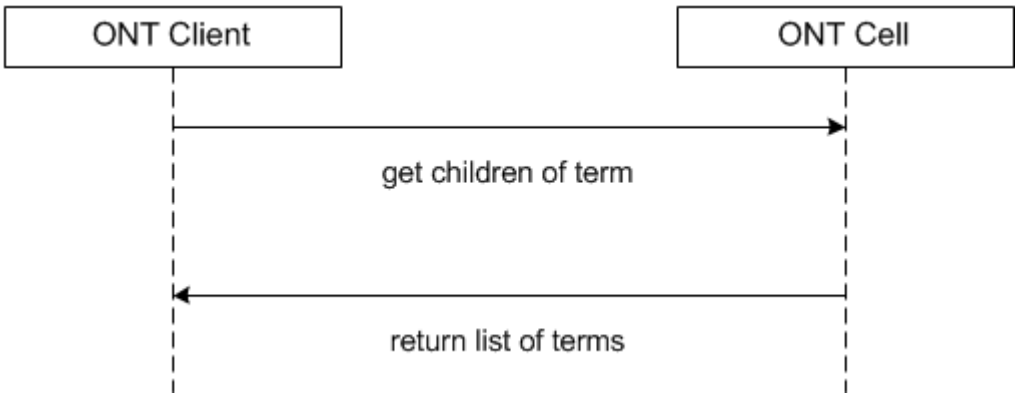
Response Message:

```
<message_body>
    <concepts>
        <concept>
```

```
            <level>0</level>
            <key>\\i2b2\RPDR</key>
            <name>Ontology</name>
            <synonym_cd>N</synonym_cd>
            <visualattributes>CA </visualattributes>
            <totalnum/>
            <basecode/>
            <metadataxml/>
            <facttablecolumn>concept_cd</facttablecolumn>
            <tablename>concept_dimension</tablename>
            <columnname>concept_path</columnname>
            <columndatatype>T</columndatatype>
            <operator>LIKE</operator>
            <dimcode>\RPDR</dimcode>
            <comment/>
            <tooltip>Ontology</tooltip>
        </concept>
    </concepts>
</message_body>
```

2.2.2 get_children

The **get_children** message returns all the children of a particular term. A client may want a list of all the children in order to expand a node of the vocabulary tree when a user is browsing through the tree.



2.2.2.1 Populating Children of Tree Nodes

The **get_children** message is used to populate tree nodes in the ontology Navigate Terms and Find Terms views. In both of these cases the table / path (root) to search are known.

The sequence of events is as follows:

1. The client sends a message with the following settings:

   max = 200 (or higher)

   type = core

> **Note**
>
> The default for the "max" setting is 200. If you wish to set this default higher you can do this by adding the following parameter to the Ontology cell:
>
> **Parameter Name:** ONTMAX
>
> **Parameter Value:** enter a number that represents the maximum number of terms to return.

2. ONT server performs the following steps:

   a. Parses <*parent*> to obtain the table key and path. Queries the table of tables to confirm that the user / role can access the table that is reference by the table key passed in. This call returns the table name referenced by that key. If not, return coded error. The client receives error message with code "TABLE_ACCESS_DENIED".

   b. If max is set, the database is queried for that number of children associated with the parent passed in.

   c. If count < max or no mas set, the database is queried for the entire list of children that meets the parent criteria.

   d. If count > max a coded error message is sent back.

3. If no errors, the client receives a list of children and populates the tree.

4. If max exceeded, the client receives an error message with the code "MAX_EXCEEDED". A dialog box is displayed to ask if the user wants to see all nodes. If no – done. If yes – client sends another message with max=empty.

2.2.2.2 get_children Request Message

The **get_children** message implies that the user is passing a key / path for a parent and wants the children returned. The parent tag will tell the service what metadata table / path to search in and for the **get_children** message must be specified. The structure of a parent is organized as follows:

```
\\table_key\path
```

The key (i2b2) plus the path (\RPDR\Diagnoses\Circulatory system (390-459)) is the parent:

```
<parent>\\i2b2\RPDR\Diagnoses\Circulatory system (390-459)</parent>
```

The remaining attributes provide information about the results to be returned. If the number of rows found is greater than max, then an error message will be return in the i2b2 header. If max is left out then it is interpreted that there is no max. The hiddens and synonyms attributes tell whether to return hiddens and synonyms. By default hiddens and synonyms are false, so if they are left out it will be false. The type tells which columns to select (default / core / all). By default, the type is set to default so you don't have to actually include it if you want the default set of columns returned. Each message will interpret "default" to be a different set of columns. **get_children's** default set of columns is set to all columns except the blob and the system / date information. If type = core, then all columns except the blob and the system / date information will be returned (same as "default" in this case). If type = all then all columns except the blob are returned. The blob attribute indicates whether or not to return the blob along with the default / core / all return columns.

```
<message_body>
    <get_children max="200" hiddens="true" synonyms="true" type="default" blob="false">
        <parent>\\i2b2\RPDR\Diagnoses\Circulatory system (390-459)</parent>
    </get_children>
</message_body>
```

2.2.2.2.1 Possible “hiddens” Settings

Some ontology terms exist but for various reasons are not displayed in the query tree.

Value	Description
false	Do not return data categorized as “hidden”
true	Include data categorized as “hidden”

2.2.2.2.2 Possible “synonyms” Settings

Some ontology terms are listed as synonyms for other terms.

Value	Description
false	Do not return data categorized as “synonym”
true	Include data categorized as “synonym”

2.2.2.2.3 Possible “type” Settings

Value	Description
default	Return all data except system / date information
core	Return all data except system / date information (same as default)
all	Return all data

2.2.2.2.4 Possible “blob” Settings

Value	Description	Example

false	Do not return data stored as a blob or clob	xml, comments
true	Return xml and comments	

2.2.2.3 get_children Response Message

The request has the following settings:

type=default

blob=true

Response Message:

```
<message_body>
    <concepts>
        <concept>
            <level>1</level>
            <key>\\i2b2\RPDR\Diagnoses\Circulatory system (390-459)\Acute Rheumatic
fever (390-392)</key>
            <name>Acute Rheumatic fever</name>
            <synonym_cd>N</synonym_cd>
            <visualattributes>FA </visualattributes>
            <totalnum/>
            <basecode/>
            <metadataxml/>
            <facttablecolumn>concept_cd</facttablecolumn>
            <tablename>concept_dimension</tablename>
            <columnname>concept_path</columnname>
            <columndatatype>T</columndatatype>
            <operator>LIKE</operator>
            <dimcode>\ RPDR\Diagnoses\Circulatory system (390-459)\Acute Rheumatic
fever (390-392)</dimcode>
            <comment/>
            <tooltip>Diagnoses \ Circulatory system \ Acute Rheumatic fever</tooltip>
        </concept>
    </concepts>
</message_body>
```

2.2.3 get_name_info

The **get_name_info** message returns the information needed to populate a tree node for a given search keyword or name. This message requires the user to pass a string that is queried against the *name* column.

2.2.3.1 Generate Tree NoDes for a Given Name

To generate a list of base tree nodes associated with a given search keyword or name, the sequence of events is as follows:

1. The client requests node(s) for a given name / category. If the request message indicates all categories should be searched, loop through all known categories for the user (type = default)
2. The ONT server performs the following steps:
    a. Query the table of tables to confirm that the user / role can access category passed in. If not, return coded error. The client receives an error message with the code “TABLE_ACCESS_DENIED”
    b. If max is set, query the database for the number of entries that meet the search criteria.
    c. If count <max or no max is set, query the database for entries that meet the search criteria.
    d. If count > max send an error message back.
3. The client generates a list of nodes that match the search criteria.
4. The client receives an error message with the code “MAX_EXCEEDED” and displays a dialog asking if the user wants to see all nodes. If no – done. If yes – client sends another message with max empty.

2.2.3.2 get_name_info Request Message

This message requires the user to pass a string that is queried against the “*name*” column. The category attribute does not have to be included and if it is not, all categories the user is allowed to see will be searched.

The remaining attributes provide information about the results to be returned. If the number of rows found is greater than the max, then an error message will be returned in the i2b2 header. If the max is left out then it is interpreted that there is no max. The hiddens and synonyms attributes tell whether to return hiddens and synonyms. By default hiddens and synonyms are false, so if they are left out it will be false. The type tells which columns to select (default / core / all). By default, the *type* is set to default. Each message will interpret the default to be a different set of columns. The default set of columns for **get_name_info** includes the name column only.

- If **type = core**, then all columns except the blob and the system / date information will be returned.
- If **type = all** then all columns except the blob are returned.

The *blob attribute* indicates whether or not to return the blob along with the default / core / all return columns.

The <*match_str*> tag tells the service which string to search for. It is implied by the message **get_name_info** that the column to search is the name. The *strategy attribute* explains how the search must match (exact, left, right, contains).

```
<message_body>
    <get_name_info category="diagnosis" max="200”"hiddens="true" synonyms="true"
type="core" blob = "false">
        <match_str strategy="contains">asthma</match_str>
    </get_name_info>
</message_body>
```

2.2.3.2.1 Possible “hiddens” Settings

Some ontology terms exist but for various reasons are not displayed in the query tree.

Value	Description
false	Do not return data categorized as “hidden”
true	Include data categorized as “hidden”

2.2.3.2.2 Possible “synonyms” Settings

Some ontology terms are listed as synonyms for other terms.

Value	Description
false	Do not return data categorized as “synonym”
true	Include data categorized as “synonym”

2.2.3.2.3 Possible “blob” Settings

Value	Description	Example
false	Do not return data stored as a blob or clob	xml, comments
true	Return xml and comments	

2.2.3.2.4 Possible “strategy” Settings

Value	Description
contains	Return data whose name contains the match string
exact	Return data whose name exactly matches the match string
left	Return data whose name starts with the match string
right	Return data whose name ends with the match string

2.2.3.3 get_name_info Response Message

The request has the following settings:

type=core

blob=false

Example:

```
<message_body>
    <concepts>
        <concept>
            <level>3</level>
            <key>\\i2b2\RPDR\Medications\MUL\(LME219) respiratory agents\(LME220)
antiasthmatic combinations</key>
            <name>Antiasthmatic combinations</name>
            <synonym_cd>N</synonym_cd>
            <visualattributes>FA </visualattributes>
            <totalnum>0</totalnum>
            <facttablecolumn>concept_cd</facttablecolumn>
            <tablename>concept_dimension</tablename>
            <columnname>concept_path</columnname>
            <columndatatype>T</columndatatype>
            <operator>like</operator>
            <dimcode>\RPDR\Medications\MUL\(LME219) respiratory agents\(LME220)
antiasthmatic combinations</dimcode>
            <tooltip>medications \ respiratory agents \ antiasthmatic combinations</tooltip>
        </concept>
    </concepts>
</message_body>
```

2.2.4 get_term_info

A **get_term_info** message implies that the user is passing a key / path for a node (“self”) and wants information about that node returned.

2.2.4.1 Return Information Associated with a Node

The **get_term_info** message is used by the timeline to obtain information associated with a node. The sequence of events is as follows:

1. The client requests information for a given term (type = core).

2. The ONT server performs the following steps:

    a. Parses <self> to obtain the table key and path. Queries the able of tables to confirm that the user / role can access the table that is reference by the table key passed in. This call returns the table name referenced by that key. If not, return coded error.

    b. Query the database for the node that meets <self> criteria.

3. If error, the client receives an error message with the code TABLE_ACCESS_DENIED.

4. The client receives information about the node.

2.2.4.2 get_term_info Request Message

A **get_term_info** message implies that the user is passing a key / path for a node ("self") and wants information about that node returned. The attributes provide information about the results to be returned. The hiddens and synonyms attribute tells whether to return hiddens and synonyms. Be default hiddens and synonyms are false, so if they are left out it will be false. The type tells which columns to select (default / core / all). By default, the type is set to default so you don't have to include it if you want the default set of columns returned. Each message will interpret "default" to be a different set of columns. getTermInfo's default set of columns consists of all columns except the blob and the system / date information. If type = core, then all columns except the blob and the system / date information will be returned (same as default). If type = all then all columns except the blob are returned. The blob attribute indicates whether or not to return the blob along with the default / core / all return columns.

```
<message_body>
    <get_term_info max="200" hiddens="false" synonyms="false" type="default"
blob="true">
        <self>\\i2b2\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive
diseases (490-496)\(493) Asthma</self>
    </get_term_info>
</message_body>
```

2.2.4.2.1 Possible "hiddens" Settings

Some ontology terms exist but for various reasons are not displayed in the query tree.

Value	Description
false	Do not return data categorized as "hidden"
true	Include data categorized as "hidden"

2.2.4.2.2 Possible "synonyms" Settings

Some ontology terms are listed as synonyms for other terms.

Value	Description
false	Do not return data categorized as "synonym"
true	Include data categorized as "synonym"

2.2.4.2.3 Possible "type" Settings

Value	Description
default	Return all data except system / date information
core	Return all data except system / date information (same as default)
all	Return all data

2.2.4.2.4 Possible "blob" Settings

Value	Description	Example
false	Do not return data stored as a blob or clob	xml, comments
true	Return xml and comments	

2.2.4.3 get_term_info Response Message

The request has the following settings:

type=default

blob=false

Example:

```
<message_body>
    <concepts>
        <concept>
            <level>4</level>
            <key>\\i2b2\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive
diseases (490-496)\(493) Asthma</key>
            <name>Asthma </name>
            <synonym_cd>N</synonym_cd>
            <visualattributes>FA </visualattributes>
            <totalnum/>
            <basecode>ICD9:493</basecode>
            <facttablecolumn>concept_cd</facttablecolumn>
            <tablename>concept_dimension</tablename>
            <columnname>concept_path</columnname>
            <columndatatype>T</columndatatype>
            <operator>LIKE</operator>
            <dimcode>\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive
diseases (490-496)\(493) Asthma</dimcode>
            <comment/>
            <tooltip>Diagnoses \ Respiratory system \ Chronic obstructive diseases \
Asthma</tooltip>
        </concept>
    </concepts>
</message_body>
```

The request has the following settings:

type=default

blob=true

Example:

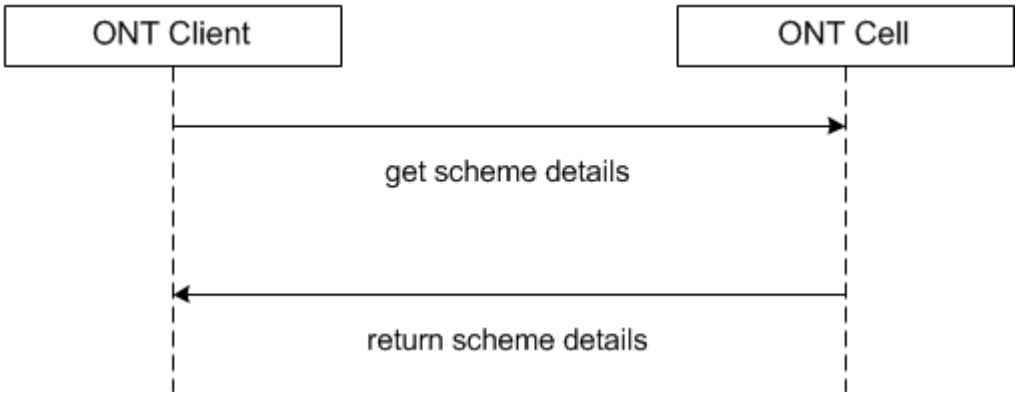
```
<message_body>
    <concepts>
        <concept>
            <level>5</level>
            <key>\\rpdr\RPDR\Labtests\LAB\(LLB16) Chemistry\(LLB31) Anemia Related
Studies\B12USAT\BC1-107</key>
            <name>B12 unsat bind (Test:bc1-107)</name>
            <synonym_cd>N</synonym_cd>
            <visualattributes>LA </visualattributes>
            <totalnum>0</totalnum>
            <basecode>BC1-107</basecode>
            <metadataxml>
                <ValueMetadata>
                    <Version>3.02</Version>
                    <CreationDateTime>10/07/2002 15:08:07</CreationDateTime>
                    <TestID>BC1-107</TestID>
                    <TestName>B12 UNSAT BIND</TestName>
                    <DataType>PosFloat</DataType>
                    <CodeType>TST</CodeType>
                    <Loinc>2171-7</Loinc>
                    <Flagstouse>HL</Flagstouse>
                    <Oktousevalues>Y</Oktousevalues>
                    <MaxStringLength />
                    <LowofLowValue>1000</LowofLowValue>
                    <HighofLowValue>1000</HighofLowValue>
                    <LowofHighValue>2000</LowofHighValue>
                    <HighofHighValue>2000</HighofHighValue>
                    <LowofToxicValue />
                    <HighofToxicValue />
                    <EnumValues>
                        <Val />
                    </EnumValues>
                    <CommentsDeterminingExclusion>
                        <Com>contamin</Com>
                        <Com>hemoly</Com>
                    </CommentsDeterminingExclusion>
                    <UnitValues>
                        <NormalUnits>ng/l</NormalUnits>
                        <EqualUnits>pg/ml</EqualUnits>
                        <ExcludingUnits />
                        <ConvertingUnits>
                            <Units />
```

```
                        <MultiplyingFactor />
                    </ConvertingUnits>
                </UnitValues>
                <Analysis>
                    <Enums />
                    <Counts />
                    <New />
                </Analysis>
            </ValueMetadata>
        </metadataxml>
        <facttablecolumn>concept_cd</facttablecolumn>
        <tablename>concept_dimension</tablename>
        <columnname>concept_path</columnname>
        <columndatatype>T</columndatatype>
        <operator>like</operator>
        <dimcode>\RPDR\Labtests\LAB\(LLB16) Chemistry\(LLB31) Anemia Related
Studies\B12USAT\BC1-107</dimcode>
        <tooltip>Labtests \ Chemistry \ Anemia Related Studies \ B12 Unsaturated
Binding (Group:B12USAT) \ B12 unsat bind (Test:bc1-107)</tooltip>
    </concept>
  </concepts>
</message_body>
```

2.2.5 get_schemes

The **get_schemes** message provides information about existing coding systems, called schemes. It returns a list of all the source systems.

This use case provides information about schemes to the client, who might want a list of all the source systems that contribute vocabulary.



A **get_schemes** message returns a list of schemes that will be displayed in the Ontology Find Terms tool. User information is provided in the *message_header*.

2.2.5.1 Generate Scheme Categories for a given User / Project

To populate the list of schemes available to a user the sequence of events is as follows:

1. A client requests a list of schemes for a given user or project (type=default).
2. The ONT server performs the following steps:
    a. Get the project / role available for the user from the PM Cell; this also serves to validate the user.
    b. Query the table of schemes and pass back a list of schemes associated with the project / role.
3. The client populates the scheme categories in the Find Terms tool.

2.2.5.2 get_schemes Request Message

```
<message_body>
    <get_schemes type="default"/>
</message_body>
```

2.2.5.2.1 Possible “type” Settings

Value	Description
default	Return key / name pairs

2.2.5.3 get_schemes Response Message

The get_schemes request has the following settings:

type=default

Response Message:

```
<message_body>
    <concepts>
        <concept>
            <key>ICD9:</key>
            <name>ICD9</name>
        </concept>
    </concepts>
</message_body>
```

2.2.6 get_code_info

The **get_code_info** message provides information about codes. A get_code_info message implies that the user is passing a *scheme:code pair* and wants the associated information about that pair returned.

2.2.6.1 Return Name / Information Associated with a Code

To generate a list of base tree nodes or a list of names associated with a given scheme:code pair, the sequence of events is as follows:

1. A client requests information for a given scheme:code (type=default)
2. The ONT server performs the following steps:
    a. Get the project / role available for the user from the PM cell; this also serves to validate the user.

b. Query the table of tables to confirm that the user / role can access category passed in. If not, return coded error.

c. Queries a list of tables for entries that match the scheme:code pair.

3. If error, the client receives an error message with the code TABLE_ACCESS_DENIED.
4. The client maps name / information to request.

2.2.6.2 get_code_info Request Message

A **get_code_info** message implies that the user is passing a *scheme:code pair* and wants the associated information about that pair returned.

The attributes provide information about the results to be returned. The hiddens and synonyms attributes tell us whether to return hiddens and synonyms. By default hiddens and synonyms are false, so if they are left out it will be false. The type tells which columns to select (default / core / all). By default, the type is set to default so you don't have to include it if you want the default set of columns returned. Each message will interpret the default to be a different set of columns. The default set of columns for **get_code_info** consists of name only.

- If **type = core**, then all columns except the blob and the system / date information will be returned.
- If **type = all** then all columns except the blob are returned.

The *blob attribute* indicates whether or not to return the blob along with the default / core / all return columns.

Example:

```
<message_body>
    <get_code_info hiddens="true" synonyms="true" type="default" blob="false">
        <match_str strategy="exact">ICD9:493</match_str>
    </get_code_info>
</message_body>
```

2.2.6.2.1 Possible “type” Settings

Value	Description
default	Return only the name
core	Return all data except the system / date information
all	Return all data

2.2.6.3 get_code_info Response Message

The ONT request to map scheme:code pair to a name has the following settings:

type=default

blob=false

Response Message:

```
<message_body>
    <concepts>
        <concept>
            <name>Asthma</name>
        </concept>
    </concepts>
</message_body>
```

The find terms search by code has the following settings:

type=core

blob=false

Response Message:

```
<message_body>
    <concepts>
        <concept>
            <level>4</level>
            <key>\\i2b2\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive
diseases (490-496)\(493) Asthma</key>
            <name>Asthma</name>
            <synonym_cd>N</synonym_cd>
            <visualattributes>FA </visualattributes>
            <totalnum>0</totalnum>
            <basecode>ICD9:493</basecode>
            <facttablecolumn>concept_cd</facttablecolumn>
            <tablename>concept_dimension</tablename>
            <columnname>concept_path</columnname>
            <columndatatype>T</columndatatype>
            <operator>LIKE</operator>
            <dimcode>\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive
diseases (490-496)\(493) Asthma</dimcode>
            <tooltip>Diagnoses \ Respiratory system \ Chronic obstructive diseases \
Asthma</tooltip>
        </concept>
    </concepts>
</message_body>
```

2.2.7 add_child

The **add_child** message provides information about a metadata item to be added to the database. An add_child message implies that the user is adding a leaf, folder or container to a given folder or container.

2.2.7.1 Add a Node to the Tree

To add a node to the tree, the sequence of events is as follows:

1. The client requests to add a leaf or folder to a given (editable) parent node.
2. The Ontology server performs the following steps:
    a. Parses the node information to obtain the key / table_cd

b. Queries the TABLE_ACCESS table for the table name associated with the TABLE_CD

c. Inserts the new leaf, folder or container into the Ontology metadata table

3. The client populates the selected parent node with the new node.

2.2.7.2 add_child Request Message

An **add_child** message requires the user to specify the node to be added. No additional attribute settings are necessary.

Example:

```
<message_body>
    <ns6:add_child>
        <level>1</level>
        <key>\\i2b2\Custom Ontology\Test folder\</key>
        <name>Test folder</name>
        <synonym_cd>N</synonym_cd>
        <visualattributes>FAE</visualattributes>
        <totalnum>0</totalnum>
        <basecode />
        <facttablecolumn>concept_cd</facttablecolumn>
        <tablename>concept_dimension</tablename>
        <columnname>concept_path</columnname>
        <columndatatype>T</columndatatype>
        <operator>LIKE</operator>
        <dimcode>\Custom Ontology\Test folder\</dimcode>
        <comment />
        <tooltip>\ Custom Ontology \ Test folder</tooltip>
        <sourcesystem_cd />
        <valuetype_cd />
    </ns6:add_child>
</message_body>
```

2.2.7.3 add_child Response Message

A **status type** of *DONE* or *ERROR* is specified in the response header. No specialized *<message_body>* is returned to the client.

2.2.8 delete_child

The **delete_child** message is sent to delete an editable node from the metadata tree. The attribute *include_children=true* indicates that children should be deleted also.

2.2.8.1 Delete a Node in the Tree

To delete a node in the tree, the sequence of events is as follows:

1. The client specifies a leaf, folder or container to be deleted.
2. The Ontology server performs the following steps:
    a. Parses the node information to obtain the key / table_cd
    b. Query the TABLE_ACCESS table for the table name associated with the TABLE_CD.
    c. Update the metadata table to delete the corresponding node and its synonyms.
    d. If *include_children = true* then also delete the children of this node.
3. The client removes the leaf, folder or container.

2.2.8.2 delete_child Request Message

This message requires the user to specify the node to be deleted. No additional attribute settings are necessary.

Example:

```
<message_body include_children="true">
    <ns6:delete_child>
        <level>1</level>
        <key>\\i2b2\Custom Ontology\Test folder\</key>
        <name>Test folder</name>
        <synonym_cd>N</synonym_cd>
        <basecode />
    </ns6:delete_child>
</message_body>
```

2.2.8.3 delete_child Response Message

A **status type** of *DONE* or *ERROR* is specified in the response header. No specialized <*message_body*> is returned to the client.

2.2.9 modify_child

The **modify_child** message is sent to edit the content of a node in the metadata tree.

2.2.9.1 Modify a Node in the Tree

To modify a node in the tree, the sequence of events is as follows:

1. The client specifies a leaf, folder or container to be modified.
2. The Ontology server performs the following steps:
    a. Parses the node information to obtain the key / table_cd
    b. Queries the TABLE_ACCESS table for the table name associated with the TABLE_CD.
    c. Update the metadata table with the new information for the corresponding node.
3. The client refreshes the leaf, folder or container.

2.2.9.2 modify_child Request Message

This message requires the user to specify the modified node's content. An attribute indicates whether synonyms changes should be applied to the synonyms. The attribute *inclSynonyms* = *true* indicates that no synonyms were added or removed ruing this edit session and we would like to apply the modifications to them. *inclSynonyms* = *false* indicates that synonyms were added or removed during this edit session; in this case the synonyms are deleted and reinserted anew.

Example:

```
<message_body>
    <ns6:modify_child  incl_synonyms="false">
        <level>1</level>
        <key>\\i2b2\Custom Ontology\Test folder\</key>
        <name>Test folder</name>
        <synonym_cd>N</synonym_cd>
        <visualattributes>FAE</visualattributes>
        <totalnum>0</totalnum>
        <basecode />
        <facttablecolumn>concept_cd</facttablecolumn>
        <tablename>concept_dimension</tablename>
        <columnname>concept_path</columnname>
        <columndatatype>T</columndatatype>
        <operator>LIKE</operator>
        <dimcode>\Custom Ontology\Test folder\</dimcode>
        <comment />
        <tooltip>\ Custom Ontology \ Test folder</tooltip>
        <sourcesystem_cd />
        <valuetype_cd />
    </ns6:modify_child>
</message_body>
```

2.2.9.3 modify_child Response Message

A **status type** of *DONE* or *ERROR* is specified in the response header. No specialized <*message_body*> is returned to the client.

2.2.10 update_crc_concept

The **update_crc_concept** message is sent to start the metadata / dimension table synchronization process.

2.2.10.1 Update Concepts in CONCEPT_DIMENSION

To update all concepts, the sequence of events is as follows:

1. The client specifies a desire to begin the synchronization process.
2. The Ontology server performs the following steps:
    a. Identifies all nodes that have been created or edited (VISUAL_ATTRIBUTE contains “E” in the third character)
    b. Uploads a file to the File Repository Cell (FRC) that contains all nodes identified above.
    c. Notifies the CRC to start an upload.
3. The client displays the progress of the process.

2.2.10.2 update_crc_concept Request Message

This message helps to synchronize the Ontology concepts with the CRC cell. i.e. Updates appropriate CRC dimension tables like the CONCEPT-DIMENSION, PROVIDER_DIMENSION and the MODIFIER_DIMENSION tables. This message has an *operation_type* attribute that may be set to “update_only” or “synchronize_all”. “update_only” instructs the ONT cell to update the appropriate CRC dimension tables with newly constructed metadata (terms with “E” in the third character in VISUAL_ATTRIBUTES). “synchronize_all” is more complicated: it rebuilds the dimension table anew with all metadata terms.

Example:

```
<message_body>
    <update_crc_concept operation_type="update_only"/>
</message_body>
```

2.2.10.2.1 Possible “hiddens” Settings

Some ontology terms exist but for various reasons are not displayed in the query tree. This setting defaults to true.

Value	Description
false	Do not update data categorized as “hidden” in the CRC
true	Include data categorized as “hidden” in the CRC update

2.2.10.2.2 Possible “synonyms” Settings

Some ontology terms are listed as synonyms for other terms. This setting defaults to false.

Value	Description
false	Do not include data categorized as “synonym” in the CRC
true	Include data categorized as “synonym” in the CRC update

2.2.10.3 update_crc_concept Response Message

A **status type** of *DONE* or *ERROR* is specified in the response header. The <*message_body*> provides process status for the requested process id.

Example:

```
<message_body>
    <ontology_process_status>
        <process_id>26</process_id>
        <process_step_cd>ONT_BUILD_PDO_START</process_step_cd>
        <start_date>2010-04-05T00:00:00.000-04:00</start_date>
```

```
        <process_status_cd>PROCESSING</process_status_cd>
        <message />
    </ontology_process_status>
</message_body>
```

2.2.11 get_process_status

The **get_process_status** message is sent to retrieve the status of the dimension table synchronization process.

2.2.11.1 Get Concept Synchronization Process Status

To get the process status, the sequence of events is as follows:

1. The client specifies a request to obtain status for a specified process_id
2. The ontology server returns the status for the specified process status parameters.
3. The client updates the progress of the process.

2.2.11.2 get_ont_process_status Request Message

This message requires the user to specify the process we are obtaining the status for.

Example:

```
<message_body>
    <get_ont_process_status>
        <process_id/>
        <process_type_cd/>
        <process_start_date>
            <start_time>2010-04-05T00:00:00.000-04:00</start_time>
            <end_time>2010-04-05T00:00:00.000-04:00</end_time>
        </process_start_date>
        <process_end_date>
            <start_time>2010-04-05T00:00:00.000-04:00</start_time>
```

```
            <end_time>2010-04-05T00:00:00.000-04:00</end_time>
        </process_end_date>
        <process_status_cd>COMPLETED</process_status_cd>
    </get_ont_process_status>
</message_body>
```

2.2.11.3 get_ont_process_status Response Message

A **status type** of *DONE* or *ERROR* is specified in the response header. The *<message_body>* provides process status for the requested process id.

Example:

```
<message_body>
    <ontology_process_status_list>
        <ontology_process_status>
            <process_id>26</process_id>
            <process_step_cd>ONT_SENTTO_CRCLOADER</process_step_cd>
            <start_date>2010-04-05T00:00:00.000-04:00</start_date>
            <end_date>2010-04-05T00:00:00.000-04:00</end_date>
            <process_status_cd>COMPLETED</process_status_cd>
            <crc_upload_id>624</crc_upload_id>
            <message />
        </ontology_process_status>
    </ontology_process_status_list>
</message_body>
```

The **process_step_cd** provides information about the type of process that is currently in progress. The following codes are in use:

process_step_cd	Description
ONT_BUILD_PDO_START	Indicates that a file containing edited metadata is being created.
ONT_SENTTO_FRC	Indicates that the metadata file has been sent to the File Repository Cell (FRC)
ONT_SENTTO_CRCLOADER	Indicates that the CRC has been instructed to upload the file sent to the FRC.

ONT_PATIENT_COUNT_UPDATE	Indicates that the process is updating the patient count information for the concepts.

The **process_status_cd** provides information about the status of the step identified at the process_step_cd.

process_status_cd	Description
ERROR	Indicates that an error occurred
PROCESSING	Indicates that the process is in progress
COMPLETED	Indicates that the process has competed
ABORT	Indicates that the process should set itself to KILLED status because the user started another instance of this process.
KILLED	Indicates that the process was killed because the user started another instance of this process.

2.2.12 get_dirty_state

The **get_dirty_state** message is sent to get state information about the need to synchronize with a dimension table.

2.2.12.1 Get Dimension Synchronize State Information

To get dirty state information, the sequence of events is as follows:

1. The client specifies a request to obtain the dirty state information
2. The ontology server returns the dirty state information

2.2.12.2 get_dirty_state Request Message

This message has no requirements or attributes settings.

Example:

```
<message_body>
    <get_dirty_state/>
</message_body>
```

2.2.12.3 get_dirty_state Response Message

A **status type** of *DONE* or *ERROR* is specified in the response header. The <*message_body*> provides the dirty state information.

Example:

```
<message_body>
    <dirty_state>NONE|ADD|DELETE_EDIT</dirty_state>
</message_body>
```

The following **dirty_state** codes are in use:

dirty_state code	Description
NONE	Indicates that no synchronization or update actions are required
ADD	Indicates an update action is required
DELETE_EDIT	Indicates a synchronization action is required

2.2.13 update_concept_totalnum

The **update_concept_totalnum** message is used to populate the patient's total count information (totalnum) in the metadata table. This service is used to synchronize the TOTALNUM value in the metadata table, if there is updates to just the datamart tables. Syncing

the TOTALNUM column helps in the query performance and also it is displayed in the Ontology's navigation view.

2.2.13.1 Update the TOTALNUM in the Metadata Table

To update the patient total count, the sequence of events is as follows:

1. The client specifies a request to start the update patient count process.
2. The ontology server returns the status of the patient count process

2.2.13.2 update_concept_totalnum Request Message

This message accepts the operation type parameter.

Example:

```
<update_concept_totalnum  operation_type = "restart_only | synchronize_all">
</update_concept_totalnum>
```

operation_type	Description
synchronize_all	This option is used to refresh the concepts TOTAL_NUM information. This will clear the TOTAL_NUM column of the metadata table by setting NULL value before updating it.
Restart_only	This option will update the concepts TOTAL_NUM information, whose TOTAL_NUM value is NULL.

2.2.13.3 update_concept_totalnum Response Message

A **status type** of *DONE* or *ERROR* is specified in the response header. The <*message_body*> provides the information about the state of the process.

Example:

```
<message_body>
    <ontology_process_status>
        <process_id>1</process_id>
        <process_type_cd>ONT_PATIENT_COUNT_UPDATE</process_type_cd>
        <process_step_cd>PROCESSED 10/95900</process_step_cd>
        <process_status_cd>PROCESSING</process_status_cd>
        <start_date>2010-04-05T00:00:00.000-04:00</start_date>
        <end_date>2010-04-05T00:00:00.000-04:00</end_date>
    </ontology_process_status>
</message_body>
```

The **process_status_cd** provides information about the status of the step identified at the process_step_cd.

process_status_cd	Description
ERROR	Indicates that an error occurred
PROCESSING	Indicates that the process is in progress
COMPLETED	Indicates that the process has competed
ABORT	Indicates that the process should set itself to KILLED status because the user started another instance of this process.
KILLED	Indicates that the process was killed because the user started another instance of this process.

2.2.14 get_modifiers

The **get_modifiers** message returns all the modifiers of a particular term. A client may want a list of all modifiers for a given term.

2.2.14.1 Populating Modifiers of Tree Nodes

The **get_modifiers** message is used to populate the modifier tree nodes in the ontology Navigate Terms tool. The table / path (root) to search are known.

The sequence of events is as follows:

1. The client sends a message with the *type* = *core*.
2. The ONT server performs the following steps:
    a. Parses <self> to obtain the table key and path. Queries the table of tables to confirm that the user / role can access the table that is referenced by the table key passed in. If not, return coded error.
    b. The database is queried for the entire list of modifiers whose C_ORIGINAL_ITEM meets the criteria specified by the path.
3. The client receives a list of modifiers and populates the tree.
4. If error, the client receives the error message with the code TABLE_ACCESS_DENIED.

2.2.14.2 get_modifiers Request Message

A **get_modifiers** message implies that the user is passing a key / path for a term ("self") and wants its modifiers returned. The <*self*> tag will tell the service what metadata table / path to search in and for the **get_modifiers** message must be specified. The structure of term "self" is organized as follows:

```
\\table_key\path
```

The key (i2b2) plus the path (\Diagnoses\Circulatory system (390-459)) is the term:

```
<parent>\\i2b2\Diagnoses\Circulatory system (390-459)</parent>
```

The remaining attributes provide information about the results to be returned. The hiddens and synonyms attributes tell whether to return hiddens and synonyms. By default hiddens and synonyms are false, so if they are left out it will be false. The type tells which columns to select (default / core / all). By default, the type is set to default so you don't have to actually include it if you want the default set of columns returned. Each message will interpret "default" to be a different set of columns. **get_modifiers** default set of columns includes all columns except the blob and the system / date information. Likewise, if attributes = core, then all columns except

the blob and the system / date information will be returned. If attributes = all then all columns except the blob are returned. The blob attribute indicates whether or not to return the blob along with the default / core / all return columns.

Example:

```
<message_body>
    <get_modifiers hiddens="true" synonyms="true" type="core" blob="false">
        <self>\\i2b2\Diagnoses\Circulatory system (390-459)</self>
    </get_modifiers>
</message_body>
```

2.2.14.2.1 Possible “hiddens” Settings

Some modifiers exist but for various reasons are not displayed in the query tree.

Value	Description
false	Do not return data categorized as “hidden”
true	Include data categorized as “hidden”

2.2.14.2.2 Possible “synonyms” Settings

Some modifiers are listed as synonyms for other modifiers.

Value	Description
false	Do not return data categorized as “synonym”
true	Include data categorized as “synonym”

2.2.14.2.3 Possible “type” Settings

Value	Description
default	Return all data except system / date information
core	Return all data except system / date information (same as default)
all	Return all data

2.2.14.2.4 Possible “blob” Settings

Value	Description	Example
false	Do not return data stored as a blob or clob	xml, comments
true	Return xml and comments	

2.2.14.3 get_modifiers Response Message

The request has the following settings:

type=core

blob=false

Response Message:

```
<message_body>
    <ns6:modifiers>
        <modifier>
            <level>1</level>
            <applied_path>\i2b2\Diagnoses\%</applied_path>
            <key>\\i2b2_DIAG\Mild\</key>
```

```
            <fullname>\Mild\</fullname>
            <name>Mild</name>
            <visualattributes>RA </visualattributes>
            <synonym_cd>N</synonym_cd>
            <totalnum xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true" />
            <basecode>mild</basecode>
            <tooltip>Mild</tooltip>
        </modifier>
        <modifier>
            <level>1</level>
            <applied_path>\i2b2\Diagnoses\%</applied_path>
            <key>\\i2b2_DIAG\Moderate\</key>
            <fullname>\Moderate\</fullname>
            <name>Moderate</name>
            <visualattributes>RA </visualattributes>
            <synonym_cd>N</synonym_cd>
            <totalnum xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true" />
            <basecode>moderate</basecode>
            <tooltip>Moderate</tooltip>
        </modifier>
        <modifier>
            <level>1</level>
            <applied_path>\i2b2\Diagnoses\%</applied_path>
            <key>\\i2b2_DIAG\Severe\</key>
            <fullname>\Severe\</fullname>
            <name>Severe</name>
            <visualattributes>DA </visualattributes>
            <synonym_cd>N</synonym_cd>
            <totalnum xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true" />
            <basecode>severe</basecode>
            <tooltip>Severe</tooltip>
        </modifier>
    </ns6:modifiers>
</message_body>
```

2.2.15 get_modifier_info

A **get_modifier_info** message implies that the user is passing a key / path for a modifier (“self”) and wants information about that modifier returned.

2.2.15.1 Return Information Associated with a Modifier

The **get_modifier_info** message is used by the CRC to obtain information associated with a modifier. The sequence of events is as follows:

1. The client requests information for a given modifier (*type* = *core*).
2. The ONT server performs the following steps:
    a. Parses <self> to obtain the table key and path. Queries the table of tables to confirm that the user / role can access the table that is referenced by the table key passed in. If not, return coded error.
    b. Query the database for the modifier that meets <*self*>, <*applied_path*> criteria.
        - If an exact match is found then the information about the requested modifier will be sent.
        - If the <*self*> is found but the <*applied_path*> is not then **all** modifiers with the requested <*self*> will be returned.
3. The client receives information about the modifier.

2.2.15.2 get_modifier_info Request Message

A **get_modifier_info** message implies that the user is passing a key / path for a modifier ("self") and wants information about that modifier returned. The attributes provide information about the results to be returned. The hiddens and synonym tell whether to return hidden and synonyms. By default hiddens and synonyms are false, so if they are left out it will be false. The type tells which columns to select (default / core / all). By default, the type is set to default so you don't have to include it if you want the default set of columns returned. Each message will interpret "default" to be a different set of columns. GetModifierInfo's default set of columns consists of all columns except the blob and the system / date information. If *type* = *core*, then all columns except the blob and the system / date information will be returned (same as default). If *type* = *all*, then all columns except the blob are returned. The blob attribute indicates whether or not to return the blob along with the default / core / all return columns.

Example:

```
<message_body>
```

```
    <get_modifier_info hiddens="false" synonyms="false" type="core" blob="true">
      <self>\\i2b2\Severe\</self>
      <applied_path>\i2b2\Diagnoses\%</applied_path>
    </get_modifier_info>
  </message_body>
```

2.2.15.2.1 Possible “hiddens” Settings

Some modifiers exist but for various reasons are not displayed in the query tree.

Value	Description
false	Do not return data categorized as “hidden”
true	Include data categorized as “hidden”

2.2.15.2.2 Possible “synonyms” Settings

Some modifiers are listed as synonyms for other modifiers.

Value	Description
false	Do not return data categorized as “synonym”
true	Include data categorized as “synonym”

2.2.15.2.3 Possible “type” Settings

Value	Description
default	Return all data except system / date information
core	Return all data except system / date information (same as default)
all	Return all data

2.2.15.2.4 Possible “blob” Settings

Value	Description	Example
false	Do not return data stored as a blob or clob	xml, comments
true	Return xml and comments	

2.2.15.3 get_modifier_info Response Message

The request has the following settings:

type=default

blob=false

Response Message:

```
<message_body>
    <modifiers>
        <modifier>
            <level>1</level>
            <applied_path>\i2b2\Diagnoses\%</applied_path>
            <key>\\i2b2\Severe\</key>
            <fullname>\Severe\</fullname>
            <name>Severe</name>
            <visualattributes>DA </visualattributes>
            <synonym_cd>N</synonym_cd>
            <totalnum xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true" />
            <basecode>severe</basecode>
            <tooltip>Severe</tooltip>
        </modifier>
    </modifiers>
```

```
</message_body>
```

2.2.16 get_modifier_children

The **get_modifier_children** message returns all the children of a particular modifier. A client may want a list of all children in order to expand a modifier of the vocabulary tree when a user is browsing through the tree.

2.2.16.1 Populating Children of Tree Nodes That are Modifiers

The **get_modifier_children** message is used to populate tree nodes that are modifiers within the Ontology Navigate Terms tool. The table / path (root) to search are known.

The sequence of events is as follows:

1. The client sends a message with the following settings:

   max = 200 (or higher)

   type = core

2. ONT server performs the following steps:

   a. Parses <*parent*> to obtain the table key and path. Queries the table of tables to confirm that the user / role can access the table that is referenced by the table key passed in. If not, return coded error.

   b. If max is set, the database is queried for that number of children associated with the parent and <*applied_path*> that is passed in.

   c. If count < max or no max set, the database is queried for the entire list of children that meets the parent criteria.

   d. If count > max a coded error message is sent back.

3. If no errors, the client receives a list of children and populates the tree.

4. If max exceeded, the client receives an error message with the code "MAX_EXCEEDED". A dialog box is displayed to ask if the user wants to see all nodes. If no – done. If yes – client sends another message with max=empty.

2.2.16.2 get_modifier_children Request Message

A **get_modifier_children** message implies that the user is passing a key / path for a parent and wants the children returned. The parent tag will tell the service what metadata table / path to search in and for the **get_modifier_children** message must be specified. The structure of a parent is organized as follows:

```
\\table_key\path
```

The key (i2b2) plus the path (\Severe\) is the parent:

```
<parent>\\i2b2\Severe\</parent>
```

The <*applied_path*> tag is used to further identify the modifier we are requesting children for. Finally, the <*applied_concept*> tag identifies the concept (key \ path pair) we are obtaining modifier children for. This information is needed in order to properly exclude potential modifier children from the concept we are ultimately linking this modifier to. Both the <*applied_path*> tag and the <*applied_concept*> tag must be included.

The remaining attributes provide information about the results to be returned. If the number of rows found is greater than max, then an error message will returned in the i2b2 header. If max is left out then it is interpreted that there is no max. The hiddens and synonyms attributes tell whether to return hiddens and synonyms. By default hiddens and synonyms are false, so if they are left out it will be false. The type tells which columns to select (default / core / all). By default, type is set to default so you don't have to actually include it if you want the default set of columns returned. Each message will interpret "default" to be a different set of columns. **get_modifier_children's** default set of columns is set to all columns except the blob and the system / date information. If *type* = *core*, then all columns except the blob and the system / date information will be returned (same as "default" in this case). If *type* = *all*, then all columns except the blob are returned. The blob attribute indicates whether or not to return the blob along with the default / core / all return columns.

Example:

```
<message_body>
```

```
    <get_modifier_children max="200" hiddens="true" synonyms="true" type="default"
blob="false">
        <parent>\\i2b2\Severe\</parent>
        <applied_path>\i2b2\Diagnoses\%</applied_path>
        <applied_concept>\\i2b2\i2b2\Diagnoses\Mental Disorders (290-
319)\</applied_concept>
    </get_modifier_children>
</message_body>
```

2.2.16.2.1 Possible “hiddens” Settings

Some modifiers exist but for various reasons are not displayed in the query tree.

Value	Description
false	Do not return data categorized as “hidden”
true	Include data categorized as “hidden”

2.2.16.2.2 Possible “synonyms” Settings

Some modifiers are listed as synonyms for other modifiers.

Value	Description
false	Do not return data categorized as “synonym”
true	Include data categorized as “synonym”

2.2.16.2.3 Possible “type” Settings

Value	Description
default	Return all data except system / date information

core	Return all data except system / date information (same as default)
all	Return all data

2.2.16.2.4 Possible “blob” Settings

Value	Description	Example
false	Do not return data stored as a blob or clob	xml, comments
true	Return xml and comments	

2.2.16.3 get_modifier_children Response Message

The request has the following settings:

type=default

blob=false

Response Message:

```
<message_body>
    <modifiers>
        <modifier>
            <level>2</level>
            <applied_path>\i2b2\Diagnoses\%</applied_path>
            <key>\\i2b2\Severe\Lethal\</key>
            <fullname>\Severe\Lethal\</fullname>
            <name>Lethal</name>
            <visualattributes>RA </visualattributes>
            <synonym_cd>N</synonym_cd>
            <totalnum xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true" />
            <basecode>lethal</basecode>
            <tooltip>Severe \ Lethal</tooltip>
```

```
        </modifier>
        <modifier>
            <level>2</level>
            <applied_path>\i2b2\Diagnoses\%</applied_path>
            <key>\\i2b2\Severe\Type I hypersensitivity\</key>
            <fullname>\Severe\Type I hypersensitivity\</fullname>
            <name>Type I hypersensitivity</name>
            <visualattributes>RA </visualattributes>
            <synonym_cd>N</synonym_cd>
            <totalnum xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true" />
            <basecode>typeIhyper</basecode>
            <tooltip>Severe \ Type I hypersensitivity</tooltip>
        </modifier>
    </modifiers>
</message_body>
```

2.2.17 get_modifier_name_info

The **get_modifier_name_info** message returns the information needed to populate a tree node for a given search keyword or name. This message requires the user to pass a string that is queried against the *name* column.

2.2.17.1 Search for Associated Modifiers With A Given Name

To generate a list of modifiers associated with a concept that contains a given search keyword or name, the sequence of events is as follows:

1. The client requests modifiers(s) associated with a concept that may contain a given name or keyword.
2. The ONT server performs the following steps:
    a. Parses <*self*> to obtain the table key and path. Queries the table of tables to confirm that the user / role can access the table that is referenced by the table key that is passed in. If not, the client receives an error message with the code “TABLE_ACCESS_DENIED”
    b. If max is set, query the database for the number of entries that meet the search criteria.
    c. If count <max or no max is set, query the database for entries that meet the search criteria.

d. If count > max send an error message back.

3. The client generates a list of modifiers that match the search criteria.

4. The client receives an error message with the code “MAX_EXCEEDED” and displays a dialog asking if the user wants to see all nodes. If no – done. If yes – client sends another message with max empty.

2.2.17.2 get_modifier_name_info Request Message

This message requires the user to pass a string that is queried against the “*name*” column. The associated concept that we are finding modifiers for is also passed.

The remaining attributes provide information about the results to be returned. If the number of rows found is greater than the max, then an error message will be returned in the i2b2 header. If the max is left out then it is interpreted that there is no max. The hiddens and synonyms attributes tell whether to return hiddens and synonyms. By default hiddens and synonyms are false, so if they are left out it will be false. The type tells which columns to select (default / core / all). By default, the *type* is set to default. Each message will interpret the default to be a different set of columns. The default set of columns for **get_modifier_name_info** includes the name column only.

- If **type = limited**, then all columns except the blob and the system / date and related dimension table information will be returned.
- If **type = core**, then all columns except the blob and the system / date information will be returned.
- If **type = all** then all columns except the blob are returned.

The *blob attribute* indicates whether or not to return the blob along with the default / limited / core / all return columns.

The <*match_str*> tag tells the service which string to search for. It is implied by the message **get_modifier_name_info** that the column to search is the name. The *strategy attribute* explains how the search must match (exact, left, right, contains).

```
<message_body>
    <get_modifier_name_info blob="true" type="limited" max="200" synonyms="true"
hiddens="false">
        <match_str strategy="contains">mild</match_str>
```

```
        <self>\\i2b2_DIAG\i2b2\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma\</self>
    </get_modifier_name_info>
</message_body>
```

2.2.17.2.1 Possible “hiddens” Settings

Some ontology modifiers exist but for various reasons are not displayed in the query tree.

Value	Description
false	Do not return data categorized as “hidden”
true	Include data categorized as “hidden”

2.2.17.2.2 Possible “synonyms” Settings

Some ontology modifiers are listed as synonyms for other terms.

Value	Description
false	Do not return data categorized as “synonym”
true	Include data categorized as “synonym”

2.2.17.2.3 Possible “type” Settings

Value	Description
default	Return name information only
limited	Return all data except system / date and dimension table information
core	Return all data except system / date information
all	Return all data

2.2.17.2.4 Possible “blob” Settings

Value	Description	Example
false	Do not return data stored as a blob or clob	xml, comments
true	Return xml and comments	

2.2.17.2.5 Possible “strategy” Settings

Value	Description
contains	Return data whose name contains the match string
exact	Return data whose name exactly matches the match string
left	Return data whose name starts with the match string
right	Return data whose name ends with the match string

2.2.17.3 get_modifier_name_info Response Message

The request has the following settings:

type=limited

blob=false

Example:

```
<message_body>
    <modifiers>
        <modifier>
            <level>1</level>
```

```
            <applied_path>\i2b2\Diagnoses\%</applied_path>
            <key>\\i2b2_DIAG\Severity\Mild\</key>
            <fullname>\Severity\Mild\</fullname>
            <name>Severity: Mild</name>
            <visualattributes>RA </visualattributes>
            <synonym_cd>N</synonym_cd>
            <totalnum />
            <basecode>SNO:255604002</basecode>
            <comment />
            <tooltip>Severity \ Mild</tooltip>
        </modifier>
        <modifier>
            <level>1</level>
            <applied_path>\i2b2\Diagnoses\%</applied_path>
            <key>\\i2b2_DIAG\Severity\Mild To Mod\</key>
            <fullname>\Severity\Mild To Mod\</fullname>
            <name>Severity: Mild to moderate</name>
            <visualattributes>RA </visualattributes>
            <synonym_cd>N</synonym_cd>
            <totalnum />
            <basecode>SNO:371923003</basecode>
            <comment />
            <tooltip>Severity \ Mild To Mod</tooltip>
        </modifier>
    </modifiers>
</message_body>
```

2.2.18 get_modifier_code_info

The **get_modifier_code_info** message provides information needed to populate a tree node for a given search keyword. This message requires the user to pass a string that is queried against the *C_BASECODE* column.

2.2.18.1 Return Name / Information Associated with a Code

To generate a list of base tree nodes or a list of names associated with a given keyword, the sequence of events is as follows:

1. A client requests modifier(s) associated with a concept. These modifiers are specified as a basecode containing a given keyword.

2. The ONT server performs the following steps:

a. Parses <*self*> to obtain the table key and path. Queries the table of tables to confirm that the user / role can access the table that is referenced by the table key that is passed in. If not, the client receives an error message with the code "TABLE_ACCESS_DENIED"

b. If max is set, query the database for the number of entries that meet the search criteria.

c. If count <max or no max is set, query the database for entries that meet the search criteria.

d. If count > max send an error message back.

3. The client generates a list of modifiers that match the search criteria.

4. The client receives an error message with the code "MAX_EXCEEDED" and displays a dialog asking if the user wants to see all nodes. If no – done. If yes – client sends another message with max empty.

2.2.18.2 get_modifier_code_info Request Message

A **get_modifier_code_info** message implies that the user is passing a *code* and wants the associated modifier information about that code returned.

The attributes provide information about the results to be returned. The hiddens and synonyms attributes tell us whether to return hiddens and synonyms. By default hiddens and synonyms are false, so if they are left out it will be false. The type tells which columns to select (default / core / all). By default, the type is set to default so you don't have to include it if you want the default set of columns returned. Each message will interpret the default to be a different set of columns. The default set of columns for **get_modifier_code_info** consists of name only.

- If **type = limited**, then all columns except the blob, the system / date information, and the related dimension table information will be returned.

- If **type = core**, then all columns except the blob and the system / date information will be returned.

- If **type = all**, then all columns except the blob are returned.

The *blob attribute* indicates whether or not to return the blob along with the default / core / all return columns.

Example:

```
<message_body>
    <get_modifier_code_info blob="true" type="limited" max="200" synonyms="true"
hiddens="false">
        <match_str strategy="exact">SNO:255604002</match_str>
        <self>\\i2b2_DEMO\i2b2\Diagnoses\Circulatory system (390-459)\Arterial vascular
disease (440-447)\(444) Arterial embolism and throm~\</self>
    </get_modifier_code_info>
</message_body>
```

2.2.18.2.1 Possible “type” Settings

Value	Description
default	Return only the name
limited	Return all data except the system / date and dimension information
core	Return all data except the system / date information
all	Return all data

2.2.18.3 get_modifier_code_info Response Message

The ONT request to map a code to a modifier has the following settings:

type=limited

blob=false

Response Message:

```
<message_body>
    <modifiers>
```

```
            <modifier>
                <level>1</level>
                <applied_path>\i2b2\Diagnoses\%</applied_path>
                <key>\\i2b2_DEMO\Severity\Mild\</key>
                <fullname>\Severity\Mild\</fullname>
                <name>Severity: Mild</name>
                <visualattributes>RA </visualattributes>
                <synonym_cd>N</synonym_cd>
                <totalnum />
                <basecode>SNO:255604002</basecode>
                <comment />
                <tooltip>Severity \ Mild</tooltip>
            </modifier>
        </modifiers>
    </message_body>
```

2.2.19 add_modifier

The **add_modifier** message provides information about a metadata modifier to be added to the database. An add_modifier message implies that the user is adding a modifier leaf, folder or container to a given term or modifier folder or container. This message is used in both *addModifier* and *excludeModifier* Ontology cell operations.

2.2.19.1 Add a Modifier to the Tree

To add a modifier to the tree, the sequence of events is as follows:

1. The client requests to add a leaf or folder to a given (editable) parent term or modifier node.
2. The Ontology server performs the following steps:
    a. Parses the modifier information to obtain the key / TABLE_CD
    b. Queries the TABLE_ACCESS table for the table name associated with the TABLE_CD
    c. Inserts the new modifier leaf, folder or container into the Ontology metadata table
3. The client populates the selected parent node with the new modifier.

2.2.19.2 add_modifier Request Message

An **add_modifier** message requires the user to specify the modifier to be added. No additional attribute settings are necessary.

Example:

```
<message_body>
    <ns6:add_modifier>
        <level>1</level>
        <applied_path>\Custom Metadata\Smoking status\Smoker\</applied_path>
        <key>\\CUST\Heavy\</key>
        <name>Heavy</name>
        <visualattributes>RAE</visualattributes>
        <synonym_cd>N</synonym_cd>
        <totalnum xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"
/>
        <basecode>heavy</basecode>
        <facttablecolumn>modifier_cd</facttablecolumn>
        <tablename>modifier_dimension</tablename>
        <columnname>modifier_path</columnname>
        <columndatatype>T</columndatatype>
        <operator>LIKE</operator>
        <dimcode>\Heavy\</dimcode>
        <comment />
        <tooltip>Heavy [\Custom Metadata\Smoking status\Smoker\]</tooltip>
        <sourcesystem_cd>lcp5_manualentry</sourcesystem_cd>
    </ns6:add_modifier>
</message_body>
```

2.2.19.3 add_modifier Response Message

A **status type** of *DONE* or *ERROR* is specified in the response header. No specialized *<message_body>* is returned to the client.

2.2.20 get_all_dblookups

As part of the **GetAllDBLookup** *service*, the client application will send a **get_all_dblookups** *message* to retrieve a list of all the database connections setup for the Ontology cell.

2.2.20.1 Processing by the Ontology Cell

The **get_all_dblookups** message is used to return a list of all the entries in the ONT_DB_LOOKUP table. The sequence of events for this process are:

STEP 1: Send Request Message

The client sends a **request message** to the Ontology Cell asking for a list of all entries (rows) in the ONT_DB_LOOKUP table.

STEP 2: Verify User Access

The Ontology Cell will send a request message to the PM Cell in order verify the user has the appropriate level of access.

- User is an Admin: The Ontology will continue processing the request.
- User is not an Admin: an error message will be returned.

STEP 3: Query the i2b2 Database

A **select query** is sent to the i2b2 database to retrieve all rows in the **ONT_DB_LOOKUP** table that meet the following criteria.

1. The **C_DOMAIN_ID** in the ONT_DB_LOOKUP table matches the **<domain> value** in the request message.

   Example:

C_DOMAIN_ID value *(ONT_DB_LOOKUP Table)*		**<domain> value** *(request message)*
i2b2demo	**=**	i2b2demo

2. The **C_OWNER_ID** in the ONT_DB_LOOKUP table either matches the **<username> value** in the request message or contains an "@".

STEP 3: Send Response Message

The Ontology sends a response message to the Client. The message contains a list of all the entries in the ONT_DB_LOOKUP table that met the above criteria.

2.2.20.2 Message Structure

The **get_all_dblookups** *request / response* messages follow the standard i2b2 messaging structure. This section and the ones that follow contain additional information that is specific to the get_all_dblookups messages.

> **Note**
>
> For additional information please see the *Messaging Overview* section within this document.

The request and response message structure for the get_all_dblookups service is divided into three parts:

1. Invocation URL
2. Request Message
3. Response Message

For the request message, the <*request*> and *invocation URL* sections are required, and for the response message, the <*response*> and <*result_status*> sections are required.

```
<i2b2:request>
```

```
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/getAllDblookups</redirect_url>
        </proxy>
        ...
    </message_header>
    <request_header>
        ...
    </request_header>
    <message_body>
        ...
    </message_body>
</i2b2:request>



<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="value">Status Message</status>
        </result_status>
    </response_header>
    <message_body>
        ...
    </message_body>
</ns5:response>
```

2.2.20.2.1 Message Elements

Element Name	Description
request	The <request> is modeled as an object using a polymorphic approach. All operation specific request objects inherit a base **RequestType *object***.
invocation url	The form of the message invocation URL is as follows: http://[*ipAddress*]:[*port#*]/i2b2/services/OntologyService/getAllDblookups
response	The <response> is also modeled as an object using a polymorphic approach. All operation

	specific response objects inherit a base **ResponseType *object*** containing a **StatusType *attribute***.
result_status	The <response_header> contains a <result_status> element, which will carry the status of the request. There are several types of statuses available: • DONE • ERROR • FATAL_ERROR • WARNING • INFO

2.2.20.3 Request Message: get_all_dblookups

```
<message_body>
    <ns4:get_all_dblookups type="default" />
</message_body>
```

2.2.20.3.1 Attributes

The only **attribute** available for *get_all_dblookups* request messages is:

Attribute Name	Description
type	Always set to "default".

2.2.20.3.2 XML Elements

<get_all_dblookups>	Container for get_all_dblookups data request

2.2.20.4 Response Message: get_all_dblookups

```
<response_header>
    <result_status>
        <status type="value">Status Message</status>
    </result_status>
</response_header>
<message_body>
    <ns3:dblookups>
        <dblookup project_path="value">
            <domain_id>value</domain_id>
            <owner_id>value</owner_id>
            <db_fullschema>value</db_fullschema>
            <db_datasource>value</db_datasource>
            <db_servertype>value</db_servertype>
            <db_nicename>value</db_nicename>
        </dblookup>
    </ns3:dblookups>
</message_body>
```

2.2.20.4.1 XML Elements

<dblookups>	**Container that wraps the <dblookup> container returned in the response message.** **The service will return all records in the ONT_DB_LOOKUP table therefore this container may contain multiple <dblookup> containers.**

<dblookup>	**Container that wraps an object which holds the data for a single record (row) in the ONT_DB_LOOKUP table.**
project_path	Contains the data stored in the **c_project_path** column in the ONT_DB_LOOKUP table. The path of the project is stored in this column.

<domain_id>	Contains the data stored in the **c_domain_id** column in the ONT_DB_LOOKUP table. The domain in which a project belongs to is stored in this column.
<owner_id>	Contains the data stored in the **c_owner_id** column in the ONT_DB_LOOKUP table. The owner of the project is stored in this column. The value may be "@".
<db_fullschema>	Contains the data stored in the **c_db_fullschema** column in the ONT_DB_LOOKUP table. The name of the Ontology database / schema is stored in this column.
<db_datasource>	Contains the data stored in the **c_datasource** column in the ONT_DB_LOOKUP table. The data source for this project is stored in this column. The value represents the connection configuration for the Ontology Cell to communicate with the database.
<db_servertype>	Contains the data stored in the **c_servertype** column in the ONT_DB_LOOKUP table. The type of database is stored in this column. The value may be any of the supported database management systems (Oracle, PostgreSQL, or SQL Server).
<db_nicename>	Contains the data stored in the **c_nicename** column in the ONT_DB_LOOKUP table. A simple name that used to easily identify the database is stored in this column.

Tip

Please refer to the *Ontology_Architecture* document for additional information about the ONT_DB_LOOKUP table in the i2b2 Database.

2.2.20.5 Use Cases

2.2.20.5.1 ADMIN User Requests All DB Lookups

The get_all_dblookups service sends a request message to the Ontology cell and generates the output based on the user's level of access and the data requested.

In this use case, a user who has the role of '*ADMIN*' has requested a list of all the entries in the ONT_DB_LOOKUP table. A response message will be returned with the requested data.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/getAllDblookups
</redirect_url>
        </proxy>
        ...
    </message_header>
    <request_header>
        ...
    </request_header>
    <message_body>
        <pm:get_all_dblookup>
        </pm:get_all_dblookup>
    </message_body>
</i2b2:request>
```

Response Message:

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="DONE">Ontology processing completed</status>
        </result_status>
    </response_header>
    <message_body>
        <ns3:dblookups>
            <dblookup project_path="/Demo_Oracle/">
                <domain_id>i2b2demo</domain_id>
                <owner_id>@</owner_id>
                <db_fullschema>i2b2demodata</db_fullschema>
                <db_datasource>java:/QueryToolDemoDS</db_datasource>
                <db_servertype>ORACLE</db_servertype>
                <db_nicename>Demo</db_nicename>
            </dblookup>
            <dblookup project_path="/Demo_SQL/">
                <domain_id>i2b2demo</domain_id>
                <owner_id>@</owner_id>
```

```
                <db_fullschema>i2b2demodata.dbo</db_fullschema>
                <db_datasource>java:/QueryToolDemoMartSQLDS</db_datasource>
                <db_servertype>SQLSERVER</db_servertype>
                <db_nicename>Demo Mart</db_nicename>
            </dblookup>
        </ns3:dblookups>
    </message_body>
</ns5:response>
```

2.2.20.5.2 Non-ADMIN User Requests All DB Lookups

The get_all_dblookups service sends a request message to the Ontology cell and generates the output based on the user's level of access and the data requested.

In this use case, a user has requested a list of all the entries in the ONT_DB_LOOKUP table, however they do not have the 'ADMIN' role. Therefore, the response message will be returned with an error message instead of the list of database connections.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/getAllDblookups
</redirect_url>
        </proxy>
        ...
    </message_header>
    <request_header>
        ...
    </request_header>
    <message_body>
        <pm:get_all_dblookup>
        </pm:get_all_dblookup>
    </message_body>
</i2b2:request>
```

Response Message:

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="ERROR">Access denied, user not an admin!"</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.21 set_dblookup

As part of the **setDblookup** *service*, the client application will send a **set_dblookup** *message* to either add a new database connection or update an existing one.

2.2.21.1 Processing by the Ontology Cell

The **set_dblookup** message is used to either add a new entry or update an existing entry in the ONT_DB_LOOKUP table. The sequence of events for this process are:

STEP 1: Verify User Access

The Ontology Cell will verify the user has the appropriate level of access.

- User is not an Admin: an error message will be returned.
- User is an Admin: the Ontology will continue processing the request.

STEP 2: Query the i2b2 Database

An **update query** is sent to the i2b2 database to either ***update an existing row*** in the ONT_DB_LOOKUP table or ***add a new row***. The criteria to find an existing entry in the ONT_DB_LOOKUP table.

1. The **C_DOMAIN_ID** in the ONT_DB_LOOKUP table matches the **<domain> value** in the request message.

   Example:

C_DOMAIN_ID value *(ONT_DB_LOOKUP Table)*		**<domain> value** *(request message)*
i2b2demo	**=**	i2b2demo

2. The **C_OWNER_ID** in the ONT_DB_LOOKUP table either matches the **<username>** value in the request message or contains an "@".

3. The **C_PROJECT_PATH** in the ONT_DB_LOOKUP table matches the **<project_path> value** from the request *message body attribute* in the request message.

   Example:

C_PROJECT_PATH value *(ONT_DB_LOOKUP Table)*		*<set_dblookup* **project_path=""**> **value** *(request message)*
i2b2demo	**=**	Test20160518

Existing Entry

If a match is found, then the columns of that existing entry will be updated according to the corresponding parameter values in the request message.

New Entry

If a match is not found, then a new entry with the provided values will be added to the table.

STEP 3: Send Response Message

Once the update or add is completed, the Ontology sends a response message to the Client. The message simply lets the Client know the process has been completed.

2.2.21.2 Message Structure

The **set_dblookup** *request / response* messages follow the standard i2b2 messaging structure. This section and the ones that follow contain additional information that is specific to the set_dblookup messages.

> **Note**
>
> For additional information please see the *Messaging Overview* section within this document.

The request and response message structure for the set_dblookup service is divided into three parts:

1. Invocation URL
2. Request Message
3. Response Message

For the **request message**, the <*request*> and *invocation URL* sections are required, and for the **response message**, the <*response*> and <*result_status*> sections are required.

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/setDblookup</redirect_url>
        </proxy>
        …
    </message_header>
    <request_header>
        …
    </request_header>
    <message_body>
        …
    </message_body>
</i2b2:request>


<ns5:response>
```

```
        <message_header>
            ...
        </message_header>
        <response_header>
            <result_status>
                <status type="DONE">Ontology processing completed</status>
            </result_status>
        </response_header>
        <message_body>
            ...
        </message_body>
    </ns5:response>
```

2.2.21.2.1 Message Elements

Element Name	Description
request	The <request> is modeled as an object using a polymorphic approach. All operation specific request objects inherit a base **RequestType *object***.
invocation url	The form of the message invocation URL is as follows: http://[*ipAddress*]:[*port#*]/i2b2/services/OntologyService/setDblookup
response	The <response> is also modeled as an object using a polymorphic approach. All operation specific response objects inherit a base **ResponseType *object*** containing a **StatusType *attribute***.
result_status	The <response_header> contains a <result_status> element, which will carry the status of the request. There are several types of statuses available: • DONE • ERROR • FATAL_ERROR • WARNING • INFO

2.2.21.3 Request Message: set_dblookup

```
<message_body>
    <ns4:set_dblookup project_path="/test20160518/" />
</message_body>
```

> **Note**
>
> The value of "/test20160518/" is simply an example intended to help illustrate this request message.

Example:

```
<message_body>
    <pm:set_dblookup project_path="test20160518/">
            <domain_id>i2b2demo</ domain_id>
            <owner_id>@</owner_id>
            <db_fullschema>i2b2demodata</db_fullschema>
            <db_datasource>java:/QueryToolDemoDS</db_datasource>
            <db_servertype>ORACLE </db_servertype>
            <db_nicename>Demo</db_nicename>
            <db_tooltip>testing, ..., 1, 2, 3, 4</db_tooltip>
            <comment>Just a test project</comment>
            <staus_cd></staus_cd>
    <pm:set_ dblookup>
</message_body>
```

2.2.21.3.1 Attributes

The only **attribute** available for *set_dblookup* request messages is:

Attribute Name	Description
project_path	The value entered here is used to identify an existing entry in the ONT_DB_LOOKUP table.

2.2.21.3.2 XML Elements

<set_dblookup>	**Container that wraps an object which holds the data for a single record (row) that is being added or updated in the ONT_DB_LOOKUP table.**
project_path	Contains the data stored in the **c_project_path** column in the ONT_DB_LOOKUP table. The path of the project is stored in this column.
<domain_id>	Contains the data stored in the **c_domain_id** column in the ONT_DB_LOOKUP table. The domain in which a project belongs to is stored in this column.
<owner_id>	Contains the data stored in the **c_owner_id** column in the ONT_DB_LOOKUP table. The owner of the project is stored in this column. The value may be "@".
<db_fullschema>	Contains the data stored in the **c_db_fullschema** column in the ONT_DB_LOOKUP table. The name of the Ontology database / schema is stored in this column.
<db_datasource>	Contains the data stored in the **c_datasource** column in the ONT_DB_LOOKUP table. The data source for this project is stored in this column. The value represents the connection configuration for the Ontology Cell to communicate with the database.
<db_servertype>	Contains the data stored in the **c_servertype** column in the ONT_DB_LOOKUP table. The type of database is stored in this column. The value may be any of the supported database management systems (Oracle, PostgreSQL, or SQL Server).
<db_nicename>	Contains the data stored in the **c_nicename** column in the ONT_DB_LOOKUP table. A simple name that used to easily identify the database is stored in this column.
<db_tooltip>	Contains the data stored in the **db_tooltip** column in the ONT_DB_LOOKUP table. A longer, sometimes hierarchical representation of the "nicename" is stored in this column.
<comment>	Contains the data stored in the **c_comment** column in the ONT_DB_LOOKUP table.
<status_cd>	Contains the data stored in the **c_status_cd** column in the ONT_DB_LOOKUP table.

Tip

Please refer to the *Ontology_Architecture* document for additional information about the ONT_DB_LOOKUP table in the i2b2 Database.

2.2.21.3.3 Required Attributes and Elements

In order to process the request, the following attributes and elements cannot be empty or missing from the xml request message. Missing information will result in an error when the message is processed by the Ontology.

- project_path (attribute)
- domain_id
- owner_id
- db_fullschema
- db_datasource
- db_servertype
- db_nicename

2.2.21.4 Response Message: set_dblookup

```
<response_header>
    <result_status>
        <status type="DONE">Ontology processing completed</status>
    </result_status>
</response_header>
```

2.2.21.5 Use Cases

2.2.21.5.1 ADMIN User Requests to Add a New or Edit an Existing Record

The **set_dblookup** service sends a request message to the Ontology cell and processes the request based on the user's level of access and the data requested.

In this use case, a user who has the role of '*ADMIN*' has requested to either add a new or edit an existing record in the ONT_DB_LOOKUP table. Once the record has been added or updated a response message with the appropriate status will be returned.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/setDblookup</re
direct_url>
        </proxy>
        ...
    </message_header>
    <message_body>
        <pm:set_dblookup project_path="/test20160518/">
            <domain_id>i2b2demo</domain_id>
            <owner_id>@</owner_id>
            <db_fullschema>i2b2demodata</db_fullschema>
            <db_datasource>java:/QueryToolDemoDS</db_datasource>
            <db_servertype>ORACLE</db_servertype>
            <db_nicename>Test</db_nicename>
            <db_tooltip></db_tooltip>
            <comment></comment>
            <status_cd></status_cd>
        </pm:set_dblookup>
    </message_body>
</i2b2:request>
```

Response Message:

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="DONE">Ontology processing completed</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.21.5.2 Non-ADMIN User Requests to Add a New or Edit an Existing Record

The **set_dblookup** service sends a request message to the Ontology cell and processes the request based on the user's level of access and the data requested.

In this use case, a user has requested to either add a new or edit an existing record in the ONT_DB_LOOKUP table, however they do not have the 'ADMIN' role. Therefore, the response message will be returned with an error message instead of adding / editing the record.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/setDblookup</re
direct_url>
        </proxy>
        ...
    </message_header>
    <message_body>
        <pm:set_dblookup project_path="/test20160518/">
            <domain_id>i2b2demo</domain_id>
            <owner_id>@</owner_id>
            <db_fullschema>i2b2demodata</db_fullschema>
            <db_datasource>java:/QueryToolDemoDS</db_datasource>
            <db_servertype>ORACLE</db_servertype>
            <db_nicename>Test</db_nicename>
            <db_tooltip></db_tooltip>
            <comment></comment>
            <status_cd></status_cd>
        </pm:set_dblookup>
    </message_body>
</i2b2:request>
```

Response Message:

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="DONE">Ontology processing completed</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.21.5.3 Required Information Not Sent in Request

The **set_dblookup** service sends a request message to the Ontology cell and processes the request based on the user's level of access and the data requested.

In this use case, a user with the appropriate level of access has requested to either add a new or edit an existing record in the ONT_DB_LOOKUP table, however an attribute or key element is missing or empty. Therefore, the response message will be returned with an error message instead of adding / editing the record in the table.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/setDblookup</re
direct_url>
        </proxy>
        ...
    </message_header>
    <message_body>
        <pm:set_dblookup project_path="">
            <domain_id>i2b2demo</domain_id>
            <owner_id>@</owner_id>
            <db_fullschema>i2b2demodata</db_fullschema>
            <db_datasource>java:/QueryToolDemoDS</db_datasource>
            <db_servertype>ORACLE</db_servertype>
            <db_nicename>Test</db_nicename>
            <db_tooltip></db_tooltip>
```

```
            <comment></comment>
            <status_cd></status_cd>
        </pm:set_dblookup>
    </message_body>
</i2b2:request>
```

In the example above, the value for the required *project_path* attribute is missing from the request message.

Response Message:

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="ERROR">'project_path', 'domain_id', 'owner_id', 'db_fullschema',
'db_datasource', 'db_servertype', or 'db_nicename' can't be missing or blank!</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.22 get_dblookup

As part of the **getDblookup** *service*, the client application will send a **get_dblookup** *message* to either add a new database connection or update an existing one.

2.2.22.1 Processing by the Ontology Cell

The **get_dblookup** message is used to return data an existing entry in the ONT_DB_LOOKUP table. The sequence of events for this process are:

STEP 1: Verify User Access

The Ontology Cell will verify the user has the appropriate level of access.

- User is not an Admin: an error message will be returned.
- User is an Admin: the Ontology will continue processing the request.

STEP 2: Query the i2b2 Database

A **select query** is sent to the i2b2 database to retrieve the data on a specific entry in the ONT_DB_LOOKUP table. The criteria to find an existing entry in the ONT_DB_LOOKUP table.

1. The **C_DOMAIN_ID** in the ONT_DB_LOOKUP table matches the **<domain> value** in the request message.

   Example:

C_DOMAIN_ID value *(ONT_DB_LOOKUP Table)*		**<domain> value** *(request message)*
i2b2demo	**=**	i2b2demo

2. The **C_OWNER_ID** in the ONT_DB_LOOKUP table either matches the **<username>** value in the request message or contains an "@".

3. The **C_PROJECT_PATH** in the ONT_DB_LOOKUP table matches the **project_path value** from the request *message body attribute* in the request message.

   Example:

C_PROJECT_PATH value *(ONT_DB_LOOKUP Table)*		*<get_dblookup* **field="project_path" value=""**> *(request message)*
i2b2demo	**=**	test20160518

STEP 3: Send Response Message

The Ontology sends a response message to the Client. The message contains a list of all the entries in the ONT_DB_LOOKUP table.

2.2.22.2 Message Structure

The **get_dblookup** *request / response* messages follow the standard i2b2 messaging structure. This section and the ones that follow contain additional information that is specific to the get_dblookup messages.

> **Note**
>
> For additional information please see the *Messaging Overview* section within this document.

The request and response message structure for the set_dblookup service is divided into three parts:

1. Invocation URL
2. Request Message
3. Response Message

For the request message, the <*request*> and *invocation URL* sections are required, and for the response message, the <*response*> and <*result_status*> sections are required.

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/getD
blookup</redirect_url>
        </proxy>
        ...
    </message_header>
    <request_header>
        ...
```

```
    </request_header>
    <message_body>
        ...
    </message_body>
</i2b2:request>

<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="DONE">Ontology processing completed</status>
        </result_status>
    </response_header>
    <message_body>
        ...
    </message_body>
</ns5:response>
```

2.2.22.2.1 Message Elements

Element Name	Description
request	The <request> is modeled as an object using a polymorphic approach. All operation specific request objects inherit a base **RequestType *object***.
invocation url	The form of the message invocation URL is as follows: http://[*ipAddress*]:[*port#*]/i2b2/services/OntologyService/getDblookup
response	The <response> is also modeled as an object using a polymorphic approach. All operation specific response objects inherit a base **ResponseType *object*** containing a **StatusType *attribute***.
result_status	The <response_header> contains a <result_status> element, which will carry the status of the request. There are several types of statuses available: • DONE • ERROR • FATAL_ERROR • WARNING • INFO

2.2.22.3 Request Message: get_dblookup

```
<message_body>
    <pm:get_dblookup field="value" value="value" />
</message_body>
```

> **Note**
>
> The value of "/test20160518/" is simply an example intended to help illustrate this request message.

Example:

```
<message_body>
    <pm:get_dblookup field="project_path" value="/test20160518/" />
</message_body>
```

If the *'field'* attribute is omitted, then *"project_path"* will be used as the default.

Example:

```
<message_body>
    <pm:get_dblookup value="/test20160518/"/>
</message_body>
```

> **Important Requirement**
>
> An error will occur if the 'field' attribute is left blank or the 'value' attribute is missing or blank.
>
> *Examples:* The following examples will result in an error when received by the Ontology Cell.

2.2.22.3.1 Attributes

The **attributes** available for *get_dblookup* request messages are:

Attribute Name	Description
field	The value for the "field" attribute is equivalent to the column name in the ONT_DB_LOOKUP table. The only difference is the field does not have the leading 'c_' in its name. ***Example:*** field="*project_path*" maps to the *c_project_path* column in the ONT_DB_LOOKUP table.
value	The specific data in the specified field to look for when querying the ONT_DB_LOOKUP table.

2.2.22.3.2 XML Elements

<get_dblookup>	Container for get_dblookup data request

Tip

Please refer to the *Ontology_Architecture* document for additional information about the ONT_DB_LOOKUP table in the i2b2 Database.

2.2.22.3.3 Required Attributes

In order to process the request, the following attributes cannot be empty or missing from the xml request message. Missing information will result in an error when the message is processed by the Ontology.

- field: attribute is missing its value (left blank)
- attribute: attribute missing or empty (left blank)

Examples

The following examples will result in an error when received by the Ontology Cell:

```
<ns4:get_dblookup field="project_path" />
<ns4:get_dblookup field="project_path" value="" />
<ns4:get_dblookup value="" />
<ns4:get_dblookup />
```

2.2.22.4 Response Message: get_dblookup

```
<response_header>
    <result_status>
        <status type="DONE">Ontology processing completed</status>
    </result_status>
</response_header>
<message_body>
    <ns4:dblookups>
        <dblookup project_path="test20160518/">
            <domain_id>i2b2demo</domain_id>
            <owner_id>@</owner_id>
            <db_fullschema>i2b2demodata</db_fullschema>
            <db_datasource>java:/QueryToolDemoDS</db_datasource>
            <db_servertype>ORACLE</db_servertype>
            <db_nicename>Demo</db_nicename>
            <db_tooltip>Demo Database Connection</db_tooltip>
            <comment>Demo Database used to demonstrate the software</comment>
            <entry_date>2016-10-05 13:00:00</entry_date>
            <change_date>2016-10-05 13:59:43</change_date>
```

```
        </dblookup>
    </ns4:dblookups>
</message_body>
```

2.2.22.4.1 XML Elements

<dblookups>	**Container that wraps the <dblookup> container returned in the response message.** **The service will return all records in the ONT_DB_LOOKUP table therefore this container may contain multiple <dblookup> containers.**

<dblookup>	**Container that wraps an object which holds the data for a single record (row) in the ONT_DB_LOOKUP table.**
project_path	Contains the data stored in the **c_project_path** column in the ONT_DB_LOOKUP table. The path of the project is stored in this column.
<domain_id>	Contains the data stored in the **c_domain_id** column in the ONT_DB_LOOKUP table. The domain in which a project belongs to is stored in this column.
<owner_id>	Contains the data stored in the **c_owner_id** column in the ONT_DB_LOOKUP table. The owner of the project is stored in this column. The value may be "@".
<db_fullschema>	Contains the data stored in the **c_db_fullschema** column in the ONT_DB_LOOKUP table. The name of the Ontology database / schema is stored in this column.
<db_datasource>	Contains the data stored in the **c_datasource** column in the ONT_DB_LOOKUP table. The data source for this project is stored in this column. The value represents the connection configuration for the Ontology Cell to communicate with the database.
<db_servertype>	Contains the data stored in the **c_servertype** column in the ONT_DB_LOOKUP table. The type of database is stored in this column. The value may be any of the supported database management systems (Oracle, PostgreSQL, or SQL Server).
<db_nicename>	Contains the data stored in the **c_nicename** column in the ONT_DB_LOOKUP table.

	A simple name that used to easily identify the database is stored in this column.

Tip

Please refer to the *Ontology_Architecture* document for additional information about the ONT_DB_LOOKUP table in the i2b2 Database.

2.2.22.5 Use Cases

2.2.22.5.1 ADMIN User Requests Data on a Specific Database Connection

The **get_dblookup** service sends a request message to the Ontology cell and generates the output based on the user's level of access and the data requested.

In this use case, a user who has the role of '*ADMIN*' has requested data on a specific entry in the ONT_DB_LOOKUP table. The response message will be returned with the requested data.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/getDblookup</re
direct_url>
        </proxy>
        ...
    </message_header>
    <request_header>
        ...
    </request_header>
    <message_body>
        <pm: get_dblookup value="/test20160518/"/>
    </message_body>
```

```
</i2b2:request>
```

Response Message:

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="DONE">Ontology processing completed</status>
        </result_status>
    </response_header>
    <message_body>
        <ns3:dblookups>
            <dblookup project_path="/test20160518/">
                <domain_id>i2b2demo</domain_id>
                <owner_id>@</owner_id>
                <db_fullschema>i2b2demodata</db_fullschema>
                <db_datasource>java:/QueryToolDemoDS</db_datasource>
                <db_servertype>ORACLE</db_servertype>
                <db_nicename>Demo</db_nicename>
            </dblookup>
        </ns3:dblookups>
    </message_body>
</ns5:response>
```

2.2.22.5.2 Non-ADMIN User Requests Data on a Specific Database Connection

The **get_dblookup** service sends a request message to the Ontology cell and generates the output based on the user's level of access and the data requested.

In this use case, a user has requested data on a specific entry in the ONT_DB_LOOKUP table, however they do not have the 'ADMIN' role. Therefore, the response message will be returned with an error message instead of the database connection information.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/getDblookup</re
direct_url>
        </proxy>
        ...
    </message_header>
    <request_header>
        ...
    </request_header>
    <message_body>
        <pm: get_dblookup value="/test20160518/"/>
    </message_body>
</i2b2:request>
```

Response Message:

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="ERROR">Access denied, user not an admin!"</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.22.5.3 Requested Data Not Found

The **get_dblookup** service sends a request message to the Ontology cell and generates the output based on the user's level of access and the data requested.

In this use case, a user with the appropriate level of access has requested data on a specific entry in the ONT_DB_LOOKUP table, however the requested data does not exist in the table. Therefore, the response message will be returned with an error message instead of the database connection information.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/getDblookup</re
direct_url>
        </proxy>
        ...
    </message_header>
    <request_header>
        ...
    </request_header>
    <message_body>
        <pm: get_dblookup value="/demo-456/"/>
    </message_body>
</i2b2:request>
```

Response Message:

```
<response_header>
    <result_status>
        <status type="DONE">No dblookup row was found! - Ontology processing
completed</status>
    </result_status>
</response_header>
```

2.2.22.5.4 Required Information Not Sent in Request

The **get_dblookup** service sends a request message to the Ontology cell and generates the output based on the user's level of access and the data requested.

In this use case, a user with the appropriate level of access has requested data on a specific entry in the ONT_DB_LOOKUP table, however the field attribute is empty or the value attribute is missing or empty. Therefore, the response message will be returned with an error message instead of the database connection information.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/getDblookup</re
direct_url>
        </proxy>
        ...
    </message_header>
    <request_header>
        ...
    </request_header>
    <message_body>
        <pm: get_dblookup field="project_path"/>
    </message_body>
</i2b2:request>
```

In the example above, the required *value* attribute is missing from the request message.

Response Message:

```
<response_header>
    <result_status>
        <status type="ERROR">'field' can't be blank, or 'value' can't be missing or
blank!</status>
    </result_status>
</response_header>
```

2.2.23 delete_dblookup

As part of the **deleteDblookup** *service*, the client application will send a **delete_dblookup** *message* to remove a specific record from the ONT_DB_LOOKUP table.

2.2.23.1 Processing by the Ontology Cell

The **delete_dblookup** message is used to remove an existing entry from the ONT_DB_LOOKUP table. The sequence of events for this process are:

STEP 1: Verify User Access

The Ontology Cell will verify the user has the appropriate level of access.

- User is not an Admin: an error message will be returned.
- User is an Admin: the Ontology will continue processing the request.

STEP 2: Query the i2b2 Database

A **delete query** is sent to the i2b2 database to ***delete a specific row*** in the ONT_DB_LOOKUP table. The criteria to find an existing entry in the ONT_DB_LOOKUP table.

1. The **C_DOMAIN_ID** in the ONT_DB_LOOKUP table matches the **value** for the **domain_id *attribute*** in the request message.

   Example:

C_DOMAIN_ID value *(ONT_DB_LOOKUP Table)*		*<delete_dblookup* **domain_id=""> value** *(request message)*
i2b2demo	**=**	i2b2demo

2. The **C_OWNER_ID** in the ONT_DB_LOOKUP table either matches the **value** for the **user_id *attribute*** in the request message or contains an "**@**".

C_OWNER_ID value *(ONT_DB_LOOKUP Table)*		*<delete_dblookup* **user_id=""> value** *(request message)*
i2b2demo	=	@

3. The **C_PROJECT_PATH** in the ONT_DB_LOOKUP table matches the **value** for the **project_path *attribute*** in the request message.

   Example:

C_PROJECT_PATH value *(ONT_DB_LOOKUP Table)*		*<delete_dblookup* **project_path=""> value** *(request message)*
i2b2demo	**=**	test20160518

STEP 3: Send Response Message

Once the deletion is completed, the Ontology sends a response message to the Client. The message simply lets the Client know the process has been completed.

2.2.23.2 Message Structure

The **delete_dblookup** *request / response* messages follow the standard i2b2 messaging structure. This section and the ones that follow contain additional information that is specific to the set_dblookup messages.

> **Note**
>
> For additional information please see the *Messaging Overview* section within this document.

The request and response message structure for the delete_dblookup service is divided into three parts:

1. Invocation URL
2. Request Message
3. Response Message

For the request message, the *<request>* and *invocation URL* sections are required, and for the response message, the *<response>* and *<result_status>* sections are required.

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/deleteDblookup</redirect_url>
        </proxy>
        ...
    </message_header>
    <request_header>
        ...
    </request_header>
    <message_body>
        ...
    </message_body>
</i2b2:request>


<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="DONE">Ontology processing completed</status>
        </result_status>
    </response_header>
    <message_body>
        ...
    </message_body>
</ns5:response>
```

2.2.23.2.1 Message Elements

Element Name	Description
request	The <request> is modeled as an object using a polymorphic approach. All operation

	specific request objects inherit a base **RequestType *object***.
invocation url	The form of the message invocation URL is as follows: http://[*ipAddress*]:[*port#*]/i2b2/services/OntologyService/deleteDblookup
response	The <response> is also modeled as an object using a polymorphic approach. All operation specific response objects inherit a base **ResponseType *object*** containing a **StatusType *attribute***.
result_status	The <response_header> contains a <result_status> element, which will carry the status of the request. There are several types of statuses available: • DONE • ERROR • FATAL_ERROR • WARNING • INFO

2.2.23.3 Request Message: delete_dblookup

```
<message_body>
    <pm:delete_dblookup project_path="xyz" domain_id="xyz" owner_id="@"/>
</message_body>
```

2.2.23.3.1 Attributes

The **attributes** available for *delete_dblookup* request messages are:

Attribute Name	Description
project_path	Equivalent to the *c_project_path* column in the ONT_DB_LOOKUP table. The value sent for this parameter will be used to search the c_project_path column for matching record(s).
domain_id	Equivalent to the *c_domain_id* column in the ONT_DB_LOOKUP table. The value sent for this parameter will be used to search the c_domain_id column for matching record(s).
owner_id	Equivalent to the *c_owner_id* column in the ONT_DB_LOOKUP table. The value sent for this parameter will be used to search the c_owner_id column for matching record(s).

Example:

The following example illustrates a typical message body for this request, with the attributes being *project_path*, *domain_id*, and *owner_id*:

```
<message_body>
    <pm:delete_dblookup project_path="/test20160518/" domain_id="i2b2demo"
owner_id="@"/>
</message_body>
```

2.2.23.3.2 XML Elements

<delete_dblookup>	Container for delete_dblookup data request

> **Tip**
>
> Please refer to the *Ontology_Architecture* document for additional information about the ONT_DB_LOOKUP table in the i2b2 Database.

2.2.23.3.3 Required Attributes

In order to process the request, the following attributes cannot be empty or missing from the xml request message. Missing information will result in an error when the message is processed by the Ontology.

- project_path
- domain_id

- owner_id

2.2.23.4 Response Message delete_dblookup

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="value">message</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.23.5 Use Cases

2.2.23.5.1 ADMIN User Requests Deletion of Record

The **delete_dblookup** service sends a request message to the Ontology cell and processes the request based on the user's level of access and the data requested.

In this use case, a user who has the role of '*ADMIN*' has requested a specific record be deleted from the ONT_DB_LOOKUP table. Once the record is deleted a response message with the appropriate status will be returned.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/deleteDblookup<
/redirect_url>
        </proxy>
        ...
    </message_header>
    <message_body>
```

```
        <pm:delete_dblookup project_path="/test20160518/" domain_id="i2b2demo"
owner_id="@"/>
    </message_body>
</i2b2:request>
```

Response Message:

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="DONE">Ontology processing completed</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.23.5.2 Non-ADMIN User Requests Deletion of Record

The **delete_dblookup** service sends a request message to the Ontology cell and process the request based on the user's level of access and the data requested.

In this use case, a user has requested a specific record be deleted from the ONT_DB_LOOKUP table, however they do not have the 'ADMIN' role. Therefore, the record will not be deleted and the response message will be returned with an error message.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/deleteDblookup<
/redirect_url>
        </proxy>
        ...
```

```
    </message_header>
    <message_body>
        <pm:delete_dblookup project_path="/test20160518/" domain_id="i2b2demo"
owner_id="@"/>
    </message_body>
</i2b2:request>
```

Response Message:

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="ERROR">Access denied, user not an admin!"</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.23.5.3 Requested Record Doesn't Exist

The **delete_dblookup** service sends a request message to the Ontology cell and process the request based on the user's level of access and the data requested.

In this use case, a user with the appropriate level of access has requested a specific record be deleted from the ONT_DB_LOOKUP table, however the record does not exist in the table. Therefore, the response message will be returned with an error message.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/deleteDblookup<
/redirect_url>
        </proxy>
        ...
```

```
    </message_header>
    <message_body>
        <pm:delete_dblookup project_path="/test20160518/" domain_id="i2b2demo"
owner_id="@"/>
    </message_body>
</i2b2:request>
```

Response Message:

```
<response_header>
    <result_status>
        <status type="DONE”>no dblookup row was deleted (could be due to no target row
found)! - Ontology processing completed</status>
    </result_status>
</response_header>
```

2.2.23.5.4 Required Information Not Sent in Request

The **delete_dblookup** service sends a request message to the Ontology cell and process the request based on the user's level of access and the data requested.

In this use case, a user with the appropriate level of access has requested a specific record be deleted from the ONT_DB_LOOKUP table, however a required parameter or value is missing. Therefore, the response message will be returned with an error message.

Request Message:

```
<i2b2:request>
    <message_header>
        <proxy>

    <redirect_url>http://[ipAddress]:[port#]/i2b2/services/OntologyService/deleteDblookup<
/redirect_url>
        </proxy>
        ...
    </message_header>
    <message_body>
        <pm:delete_dblookup domain_id="i2b2demo" owner_id="@"/>
```

```
    </message_body>
</i2b2:request>
```

In the example above, the required *project_path* attribute is missing from the request message.

Response Message:

```
<ns5:response>
    <message_header>
        ...
    </message_header>
    <response_header>
        <result_status>
            <status type="ERROR">
                'project_path', or 'domain_id', 'owner_id' can't be missing or blank!"
            </status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.24 load_metadata

The **load_metadata** message provides information about a new record to be added to a specified table. This message is used in the *load_metadata* Ontology cell operations.

2.2.24.1 Load a metadata record

To load a metadata table record, the sequence of events is as follows:

1. The client requests to load a metadata record into a specified table.
2. The Ontology server performs the following steps:
    a. Determines which table is being loaded (table_access, schemes or specified metadata table)
    b. Parses the metadata record
    c. If loading the table_access table

i. Determines if the table_cd is unique

ii. If table_cd is unique; loads the table_access record; if not, it notifies the client that the record already exists.

d. If loading the schemes table

i. Determines if the scheme is unique

ii. If scheme is unique; loads the record; if not, it notifies the client that the record already exists.

e. If loading a specified metadata table, if table_name exists; loads the record; if not, it creates the table and then loads the record.

3. User is notified that either the record was successfully loaded or the record already exists.

2.2.24.2 load_metadata Request Message

A **load_metadata** message requires the user to specify the record to be added. No additional attribute settings are necessary.

Example:

```
<message_body>
    <ns6:load_metadata>
        <table_name>table_access</ table_name>
        <metadata>
            <ontology_data>
                <table_cd>ICD100CM_2015AA</table_cd>
                <table_name>ICD100CM_2015AA</table_name>
                <protected_access>N</protected_access>
                <level>0</level>
                <fullname>\ICD10CM_2015AA\</fullname>
                <name>ICD10CM 2015AA</name>
                <visualattributes>CAE</visualattributes>
                <synonym_cd>N</synonym_cd>
                <totalnum xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:nil="true" />
                <basecode/>
                <facttablecolumn>concept_cd</facttablecolumn>
```

```
                <tablename>concept_dimension</tablename>
                <columnname>concept_path</columnname>
                <columndatatype>T</columndatatype>
                <operator>LIKE</operator>
                <dimcode>\ICD10CM_2015AA\</dimcode>
                <comment />
                <tooltip> ICD10CM 2015AA</tooltip>
                <sourcesystem_cd>NCBO</sourcesystem_cd>
            </ontology_data>
        </metadata>
    </ns6:load_metadata>
</message_body>
```

2.2.24.3 load_metadata Response Message

A **status type** of *DONE* or *ERROR* is specified in the response header. No specialized *<message_body>* is returned to the client.

3 ONT CELL XML SCHEMA DEFINITIONS

The Ontology Management XML schema consists of the following XSD files that define the <*message_body*> for the entire ONT cell.

3.1 ONT.xsd

Describes the schema components that are common to requests and responses.

3.2 ONT_QRY.xsd

Describes the request message body for all the operations described in section 3 of this document; a query for retrieving information from rows in the metadata tables.

3.3 ONT_RESP.xsd

Describes the response message body for all the operations described in section 3 of this document; an object that holds information from rows in a metadata table.

4 GLOSSARY

4.1 Message Tags & Attribute Definitions

4.1.1 ONT_QRY.xsd

<request>	Container for request information

<get_children>	Container for vocabulary data request
max	The maximum number of results requested.
hiddens	A flag to indicate if hidden terms should be returned.
synonyms	A flag to indicate if synonymous terms should be returned.
type	Indicates the amount of data to be returned (default / core / all).
blob	A flag to indicate if the blob / clob fields should be returned.

<parent>	The name of the parent node at which the request is targeted.

<get_term_info>	Container for vocabulary data request
max	The maximum number of results requested.
hiddens	A flag to indicate if hidden terms should be returned.
synonyms	A flag to indicate if synonymous terms should be returned.
type	Indicates the amount of data to be returned (default / core / all).
blob	A flag to indicate if the blob / clob fields should be returned.

<get_name_info>	Container for vocabulary data request
category	The category to search in

<get_code_info>	Container for vocabulary data request
category	The category to search in

<match_str>	Stores the value of a string that should be matched
strategy	Matching strategy (exact / left / right / contains)

<get_categories>	Container for vocabulary data request
type	Indicates the amount of data to be returned (default / core / all)
blob	A flag to indicate if the blob / clob fields should be returned

hiddens	A flag to indicate if hidden terms should be returned
synonyms	A flag to indicate if synonymous terms should be returned

<get_schemes>	Container for vocabulary data request
type	Indicates the amount of data to be returned (default / core / all)
blob	A flag to indicate if the blob / clob fields should be returned.

<add_child>	Container for term addition request
<level>	Also known as the hierarchy level. It represents the level of the item in the ontology tree where the root level is level 0.
<key>	A "*table code**hierarchy path*" pair: \\i2b2\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma Equates to a table code of "i2b2" and the hierarchy of "\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma"
<name>	The name of the metadata term. The following characters are not allowed for leaf names: * | \ / " < ? % Due to their prevalence in genomics the following characters are allowed but their

	general use is discouraged. _ > : The following characters are not allowed for folder or container names: * | \ / " < ? % > :
<synonym_cd>	Indicates whether or not the term is a synonym of another term. The value can be "Y" or "N".
<visualattribues>	Three character string that indicates the graphical representation of the term. Example: String XYZ, where X can be "C" (container), "F" (folder), "L" (leaf), "M" (multi) Y can be "A" (active), "I" (inactive), "H" (hidden) Z can be "E" (editable) or null
<totalnum>	The number of items found (not typically used).
<basecode>	The code representation of the term.
<metadataxml>	Additional xml that may be associated with the term (optional).
<facttablecolumn>	The observation fact table column associated with the term.
<tablename>	The dimension table associated with this term.
<columnname>	The name of the column in the dimension table associated with this term.
<columndatatype>	The data type for the column in the dimension table associated with this term. "T" = text "N" = number
<operator>	The SQL comparison operator associated with this term.
<dimcode>	The dimension code that is associated with the term.
<comment>	The optional comment associated with the term.
<tooltip>	A tooltip that is associated with the term.
<valuetype_cd>	Indicates the type of term DOC = the term represents a document or note LAB = the term is associated with a lab result KEY = a keyword associated with a term (may be used for NLP)

<delete_child>	**Container for term deletion request**
include_children	A flag to indicate if the children of this node should also be deleted.

<level>	Also known as the hierarchy level. It represents the level of the item in the ontology tree where the root level is level 0.
<key>	A "*table code**hierarchy path*" pair: \\i2b2\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma Equates to a table code of "i2b2" and the hierarchy of "\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma"
<name>	The name of the metadata term.
<synonym_cd>	Indicates whether or not the term is a synonym of another term. The value can be "Y" or "N".
<basecode>	The code representation of the term.

<modify_child>	**Container for request to modify a term**
incl_synonyms	A flag to indicate if synonyms of this node should also be modified.
<level>	Also known as the hierarchy level. It represents the level of the item in the ontology tree where the root level is level 0.
<key>	A "*table code**hierarchy path*" pair: \\i2b2\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma Equates to a table code of "i2b2" and the hierarchy of "\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma"
<name>	The name of the metadata term. The following characters are not allowed for leaf names: * | \ / " < ? % Due to their prevalence in genomics the following characters are allowed but their general use is discouraged. _ > : The following characters are not allowed for folder or container names: * | \ / " < ? % > :
<synonym_cd>	Indicates whether or not the term is a synonym of another term. The value can be "Y" or "N".
<visualattribues>	Three character string that indicates the graphical representation of the term.

	Example: String XYZ, where X can be "C" (container), "F" (folder), "L" (leaf), "M" (multi) Y can be "A" (active), "I" (inactive), "H" (hidden) Z can be "E" (editable) or null
<totalnum>	The number of items found (not typically used).
<basecode>	The code representation of the term.
<metadataxml>	Additional xml that may be associated with the term (optional).
<facttablecolumn>	The observation fact table column associated with the term.
<tablename>	The dimension table associated with this term.
<columnname>	The name of the column in the dimension table associated with this term.
<columndatatype>	The data type for the column in the dimension table associated with this term. "T" = text "N" = number
<operator>	The SQL comparison operator associated with this term.
<dimcode>	The dimension code that is associated with the term.
<comment>	The optional comment associated with the term.
<tooltip>	A tooltip that is associated with the term.
<valuetype_cd>	Indicates the type of term DOC = the term represents a document or note LAB = the term is associated with a lab result KEY = a keyword associated with a term (may be used for NLP)

<update_crc_concept>	**Container for the synchronize dimension table request**
operation_type	Indicates the synchronization operation request type. **value** / **Description** update_only / Only add new terms from metadata to dimension table synchronize_all / Perform a full synchronization between metadata and dimension table
hiddens	A flag to indicate if hidden terms should be included in the request.
synonyms	A flag to indicate if synonymous terms should be included in the request.

<get_ont_process_status>	Container for synchronization process status request
<process_id>	The ID of the process we are requesting the status for

<get_dirty_state>	Container for the dirty state request

<get_modifiers>	Container for vocabulary modifier data request
max	The maximum number of results requested.
hiddens	A flag to indicate if hidden modifiers should be returned.
synonyms	A flag to indicate if synonymous modifiers should be returned.
type	Indicates the amount of data to be returned (default / core / all).
blob	A flag to indicate if the blob / clob fields should be returned.

<get_modifier_children>	Container for vocabulary modifier data request
max	The maximum number of results requested.
hiddens	A flag to indicate if hidden modifiers should be returned.
synonyms	A flag to indicate if synonymous modifiers should be returned.
type	Indicates the amount of data to be returned (default / core / all).
blob	A flag to indicate if the blob / clob fields should be returned.
<parent>	The fullname of the parent modifier at which the request is targeted.
<applied_path>	The concept path that the parent modifier applies to.
<applied_concept>	The concept we are ultimately linking this modifier to.

<get_modifier_info>	Container for vocabulary modifier data request
max	The maximum number of results requested.
hiddens	A flag to indicate if hidden modifiers should be returned.
synonyms	A flag to indicate if synonymous modifiers should be returned.

type	Indicates the amount of data to be returned (default / core / all).
blob	A flag to indicate if the blob / clob fields should be returned.
<self>	The fullname of the modifier at which the request is targeted.
<applied_path>	The concept path the modifier applies to.

<get_modifier_name_info>	**Container for vocabulary modifier data request**
max	The maximum number of results requested.
hiddens	A flag to indicate if hidden modifiers should be returned.
synonyms	A flag to indicate if synonymous modifiers should be returned.
type	Indicates the amount of data to be returned (default / core / all).
blob	A flag to indicate if the blob / clob fields should be returned.
<self>	The fullname of the concept at which the requested for the specified modifier is targeted.
<match_str>	The keyword specifying the name of the modifier to return

<get_modifier_code_info>	**Container for vocabulary modifier data request**
max	The maximum number of results requested.
hiddens	A flag to indicate if hidden modifiers should be returned.
synonyms	A flag to indicate if synonymous modifiers should be returned.
type	Indicates the amount of data to be returned (default / core / all).
blob	A flag to indicate if the blob / clob fields should be returned.
<self>	The fullname of the concept at which the requested for the specified modifier is targeted.
<match_str>	The keyword specifying the code of the modifier to return

<get_all_dblookups>	**Container for get_all_dblookups data request**
type	Always set to "default".

<set_dblookup>	Container that wraps an object which holds the data for a single record (row) that is being added or updated in the ONT_DB_LOOKUP table.
project_path	Contains the data stored in the **c_project_path** column in the ONT_DB_LOOKUP table. The path of the project is stored in this column.
<domain_id>	Contains the data stored in the **c_domain_id** column in the ONT_DB_LOOKUP table. The domain in which a project belongs to is stored in this column.
<owner_id>	Contains the data stored in the **c_owner_id** column in the ONT_DB_LOOKUP table. The owner of the project is stored in this column. The value may be "@".
<db_fullschema>	Contains the data stored in the **c_db_fullschema** column in the ONT_DB_LOOKUP table. The name of the Ontology database / schema is stored in this column.
<db_datasource>	Contains the data stored in the **c_datasource** column in the ONT_DB_LOOKUP table. The data source for this project is stored in this column. The value represents the connection configuration for the Ontology Cell to communicate with the database.
<db_servertype>	Contains the data stored in the **c_servertype** column in the ONT_DB_LOOKUP table. The type of database is stored in this column. The value may be any of the supported database management systems (Oracle, PostgreSQL, or SQL Server).
<db_nicename>	Contains the data stored in the **c_nicename** column in the ONT_DB_LOOKUP table. A simple name that used to easily identify the database is stored in this column.
<db_tooltip>	Contains the data stored in the **db_tooltip** column in the ONT_DB_LOOKUP table. A longer, sometimes hierarchical representation of the "nicename" is stored in this column.
<comment>	Contains the data stored in the **c_comment** column in the ONT_DB_LOOKUP table.
<status_cd>	Contains the data stored in the **c_status_cd** column in the ONT_DB_LOOKUP table.

<get_dblookup>	Container for get_dblookup data request
field	The value for the "field" attribute is equivalent to the column name in the ONT_DB_LOOKUP table. The only difference is the field does not have the leading 'c_' in its name. ***Example:*** field="*project_path*" maps to the *c_project_path* column in the ONT_DB_LOOKUP table.
value	The specific data in the specified field to look for when querying the ONT_DB_LOOKUP table.

<delete_dblookup>	Container for delete_dblookup data request
project_path	Equivalent to the *c_project_path* column in the ONT_DB_LOOKUP table. The value sent for this parameter will be used to search the c_project_path column for matching record(s).
domain_id	Equivalent to the *c_domain_id* column in the ONT_DB_LOOKUP table. The value sent for this parameter will be used to search the c_domain_id column for matching record(s).
owner_id	Equivalent to the *c_owner_id* column in the ONT_DB_LOOKUP table. The value sent for this parameter will be used to search the c_owner_id column for matching record(s).

<metadataLoad>	Container for a list of ontologyData to load
<table_name>	The table the data will be loaded into (e.g. TABLE_ACCESS, SCHEMES, or metadata)

<ontologyData>	Container for ontologyData to load; an object that holds information from rows in a table_access, schemes, or metadata table
<table_cd>	A unique identifier for a category
<table_name>	Name of metadata table that a category is found in.
<level>	Also known as the hierarchy level. It represents the level of the concept item in the ontology tree where the root level is level 0.
<protected_access>	Y/N Boolean indicating if the category has protected access or not.
<fullname>	Fullname representing the hierarchy of the category. May also be represented as a "*table code**hierarchy path*" pair: \\i2b2\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma Equates to a table code of "i2b2" and the hierarchy of "\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma"
<name>	The name of the metadata term. The following characters are not allowed for leaf names: * | \ / " < ? % Due to their prevalence in genomics the following characters are allowed but their general use is discouraged.

	_ > : The following characters are not allowed for folder or container names: * | \ / " < ? % > :
<synonym_cd>	Indicates whether or not the term is a synonym of another term. The value can be "Y" or "N".
<visualattribues>	Three character string that indicates the graphical representation of the term. Example: String XYZ, where X can be "C" (container), "F" (folder), "L" (leaf), "M" (multi) Y can be "A" (active), "I" (inactive), "H" (hidden) Z can be "E" (editable) or null
<totalnum>	The number of items found (not typically used).
<basecode>	The code representation of the term.
<metadataxml>	Additional xml that may be associated with the term (optional).
<facttablecolumn>	The observation fact table column associated with the term.
<dimtablename>	The dimension table associated with this term.
<columnname>	The name of the column in the dimension table associated with this term.
<columndatatype>	The data type for the column in the dimension table associated with this term. "T" = text "N" = number
<operator>	The SQL comparison operator associated with this term.
<dimcode>	The dimension code that is associated with the term.
<comment>	The optional comment associated with the term.
<tooltip>	A tooltip that is associated with the term.
<valuetype_cd>	Indicates the type of term DOC = the term represents a document or note LAB = the term is associated with a lab result KEY = a keyword associated with a term (may be used for NLP)
<applied_path>	The path of the concept that a modifier applies to.
<exclusion_cd>	Indicates whether or not the applied_path represents paths from which the modifier should be excluded.
<path>	The fullname of a concept's parent

<symbol>	The symbol associated with a concept or modifier.
<status_cd>	The status code associated with a category.
<pcori_basecode>	A PCORI basecode assigned to the concept or modifier.
<key>	Prefix associated with the scheme (e.g. 'ICD9:')
<description>	A descriptor for the scheme
<download_date>	The download date associated with the term
<update_date>	The entry date associated with the term
<import_date>	The import data associated with a term
<change_date>	The change date associated with a category
<entry_date>	The entry date associated with a category

4.1.2 ONT_RESP.xsd

<response>	**Container for response information**

<concepts>	**Container for a list of terms returned in a response**

<concept>	**Container for a term returned in a response; an object that holds information from rows in a metadata table**
<level>	Also known as the hierarchy level. It represents the level of the concept item in the ontology tree where the root level is level 0.
<key>	A "*table code**hierarchy path*" pair: \\i2b2\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases

	(490-496)\(493) Asthma Equates to a table code of "i2b2" and the hierarchy of "\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma"
<name>	The name of the metadata term. The following characters are not allowed for leaf names: * | \ / " < ? % Due to their prevalence in genomics the following characters are allowed but their general use is discouraged. _ > : The following characters are not allowed for folder or container names: * | \ / " < ? % > :
<synonym_cd>	Indicates whether or not the term is a synonym of another term. The value can be "Y" or "N".
<visualattribues>	Three character string that indicates the graphical representation of the term. Example: String XYZ, where X can be "C" (container), "F" (folder), "L" (leaf), "M" (multi) Y can be "A" (active), "I" (inactive), "H" (hidden) Z can be "E" (editable) or null
<totalnum>	The number of items found (not typically used).
<basecode>	The code representation of the term.
<metadataxml>	Additional xml that may be associated with the term (optional).
<facttablecolumn>	The observation fact table column associated with the term.
<tablename>	The dimension table associated with this term.
<columnname>	The name of the column in the dimension table associated with this term.
<columndatatype>	The data type for the column in the dimension table associated with this term. "T" = text "N" = number
<operator>	The SQL comparison operator associated with this term.
<dimcode>	The dimension code that is associated with the term.
<comment>	The optional comment associated with the term.
<tooltip>	A tooltip that is associated with the term.

<valuetype_cd>	Indicates the type of term DOC = the term represents a document or note LAB = the term is associated with a lab result KEY = a keyword associated with a term (may be used for NLP)
<modifier>	Modifier associated with the concept, if it exists.

<ontology_process_status>	**Container for the requested synchronization process status**
<process_id>	The ID for the process we requested the status for
<process_type_cd>	Provides information about the type of process that is currently in progress. The following codes are in use: **value** / **Description** ONT_ADD_CONCEPT — A term has been added ONT_DELETE_CONCEPT — A term has been deleted ONT_EDIT_CONCEPT — A term has been edited ONT_UPDATE_CRC_CONCEPT — An "update_only" has been performed. ONT_SYNCALL_CRC_CONCEPT — A "synchronize_all" has been performed
<process_step_cd>	In the case of ONT_UPDATE_CRC_CONCEPT and ONT_SYNCALL_CRC_CONCEPT, provides additional information about the process step that is currently in progress. The following codes are in use: **value** / **Description** ONT_BUILD_PDO_START — Start the process ONT_SENTTO_FRC — The file has been sent to the File Repository Cell (FRC) ONT_SENTTO_CRCLOADER — Instructs the CRC to load the file
<start_date>	The date the process was requested
<end_date>	The date the process was completed
<process_status_cd>	Provides information about the process step: **value** / **Description**

	ERROR	An error has occurred
	PROCESSING	The process step is in progress
	COMPLETED	The process has completed
<crc_upload_id>	The corresponding CRC upload id for the ONT_SENTTO_CRC_LOADER process step	
<message>	Miscellaneous status message	

<dirty_state>	**Container for the dirty state information**
	The following **dirty state** codes are in use: **value** / **Description** NONE – Indicates that no synchronization or update actions are required ADD – Indicates an update action is required DELETE_EDIT – Indicates a synchronization action is required

<modifier>	**Container for a modifier returned in a response; an object that holds information from rows in a metadata table**
<level>	Also known as the hierarchy level. It represents the level of the concept item in the ontology tree where the root level is level 0.
<applied_path>	The path of the concept that the modifier applies to.
<key>	A "*table code**hierarchy path*" pair: \\i2b2\Mild Equates to a table code of "i2b2" and the hierarchy of "\Mild\"
<name>	The name of the metadata term. The following characters are not allowed for leaf names: * | \ / " < ? % Due to their prevalence in genomics the following characters are allowed but their general use is discouraged.

	_ > : The following characters are not allowed for folder or container names: * | \ / " < ? % > :
<synonym_cd>	Indicates whether or not the modifier is a synonym of another modifier. The value can be "Y" or "N".
<visualattribues>	Three character string that indicates the graphical representation of the term. Example: String XYZ, where X can be "O" (container), "D" (folder), "R" (leaf) Y can be "A" (active), "I" (inactive), "H" (hidden) Z can be "E" (editable) or null
<totalnum>	The number of patients having this modifier (not typically used).
<basecode>	The code representation of the modifier.
<metadataxml>	Additional xml that may be associated with the modifier (optional).
<facttablecolumn>	The observation fact table column associated with the modifier.
<tablename>	The dimension table associated with the modifier.
<columnname>	The name of the column in the dimension table associated with the modifier.
<columndatatype>	The data type for the column in the dimension table associated with the modifier. "T" = text "N" = number
<operator>	The SQL comparison operator associated with the modifier.
<dimcode>	The dimension code that is associated with the modifier.
<comment>	The optional comment associated with the modifier.
<tooltip>	A tooltip that is associated with the modifier.

<dblookups>	**Container that wraps the <dblookup> container returned in the response message.** **The service will return all records in the ONT_DB_LOOKUP table therefore this container may contain multiple <dblookup> containers.**

<dblookup>	Container that wraps an object which holds the data for a single record (row) in the ONT_DB_LOOKUP table.
project_path	Contains the data stored in the **c_project_path** column in the ONT_DB_LOOKUP table. The path of the project is stored in this column.
<domain_id>	Contains the data stored in the **c_domain_id** column in the ONT_DB_LOOKUP table. The domain in which a project belongs to is stored in this column.
<owner_id>	Contains the data stored in the **c_owner_id** column in the ONT_DB_LOOKUP table. The owner of the project is stored in this column. The value may be "@".
<db_fullschema>	Contains the data stored in the **c_db_fullschema** column in the ONT_DB_LOOKUP table. The name of the Ontology database / schema is stored in this column.
<db_datasource>	Contains the data stored in the **c_datasource** column in the ONT_DB_LOOKUP table. The data source for this project is stored in this column. The value represents the connection configuration for the Ontology Cell to communicate with the database.
<db_servertype>	Contains the data stored in the **c_servertype** column in the ONT_DB_LOOKUP table. The type of database is stored in this column. The value may be any of the supported database management systems (Oracle, PostgreSQL, or SQL Server).
<db_nicename>	Contains the data stored in the **c_nicename** column in the ONT_DB_LOOKUP table. A simple name that used to easily identify the database is stored in this column.

4.1.3 Optional <metadataxml> content

Currently this tag is used for concepts of a laboratory nature [LAB] or concepts related to NLP [KEY]

<ValueMetadata>	Container for information associated with a laboratory value [LAB] or NLP-related keyword [KEY]
<Version>	The version associated with this data
<CreationDateTime>	The timestamp associated with this data
<DataType>	Identifies the data type. Possible values are:

	For numeric values the datatypes are: PosFloat, Float, PosInteger, Integer For pre-determined text values the datatypes are: Enum For free form text values the datatypes are: String, largestring
<Flagstouse>	Represents the type of flag to use. Examples values: HL = High / Low (does not apply to String types) A = Abnormal
<Oktousevalues>	Indicates if it OK to use values Y indicates the value is a number (numeric types) N or empty indicates the value is not a number
<MaxStringLength>	The maximum length associated with a String data type

Note

The next six tags apply to **numeric** types.

<LowofLowValue>	Values less than this are categorized as *Very Low*.
<HighofLowValue>	Values between this and low of low are categorized as *Low*.
<LowofHighValue>	Values between this and high of low are categorized as *Medium*.
<HighofHighValue>	Values between this and low of high are categorized as *High*. Values greater than this are categorized as *Very High*.
<LowofToxicValue>	The low end of the toxic value range
<HighofToxicValue>	The high end of the toxic value range

Note

The container applies to **Enum** types.

<EnumValues>	Container of values associated with the enum data type
<Val description>	Enum data type value

Note

This container applies to any type associated with a **unit of measurement**

<UnitValues>	Container for the unit of measurement value information
<NormalUnits>	The unit of measurement associated with the value.
<EqualUnits>	An equivalent unit of measurement associated with the value.
<ConvertingUnits>	Container for unit conversion information
<Units>	Conversion unit of measurement
<MultiplyingFactor>	Conversion unit multiplication fact.