Informatics for Integrating Biology and the Bedside

i2b2 Cell Messaging

Ontology Management (ONT) Cell

Document Version: *1.5.1*

i2b2 Software Version: *1.5*

Table of Contents

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.bisti.nih.gov/ncbc/). One goal 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 communicate using standard, pre-defined i2b2 XML request and response 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> as shown, below in Figure 1.

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 it's own <message_header>, <response_header> and <message_body> and it may optionally echo the request's <request_header> as shown, below in Figure 1.

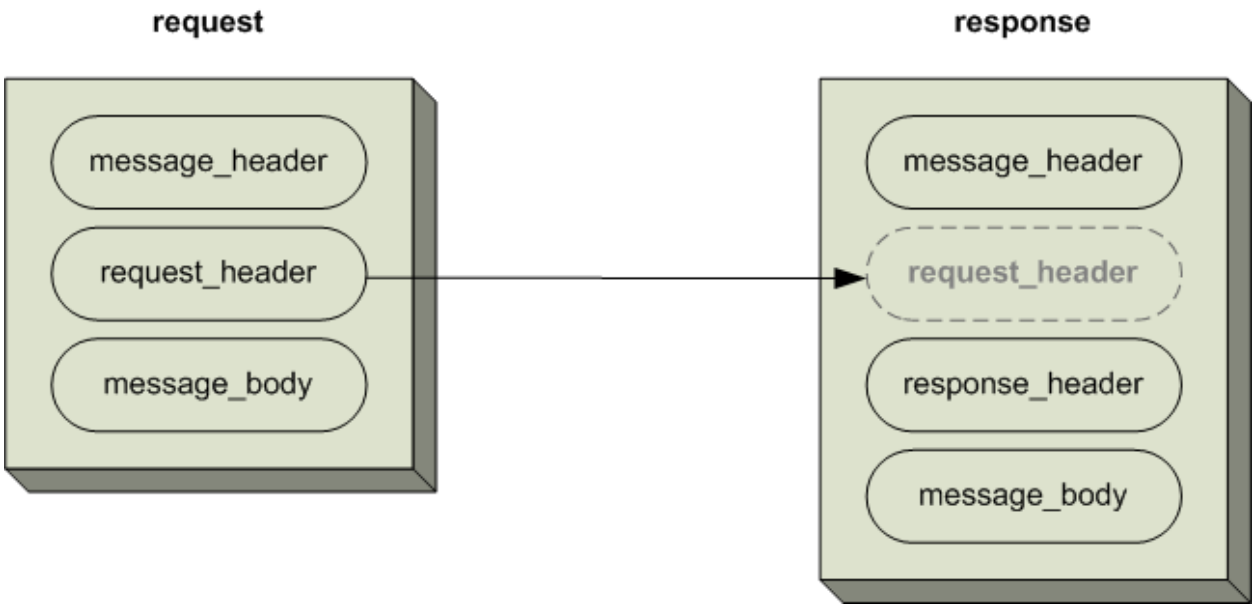


Figure 1: The basic structure of a request and response message. The request_header in the request can be echoed in the response.

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 request and response messages contain this <message_header> tag which has control information such as sending application, receiving application and 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 request and response messages contain a <message_body> tag which may contain any well-formed xml. Individual cells may define cell-specific XML that will be put inside <message_body> tag. This cell-specific XML need not 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 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 relationships 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 diagram below, which consists of a table of diagnoses terms and uses a single rule to build all diagnosis queries.

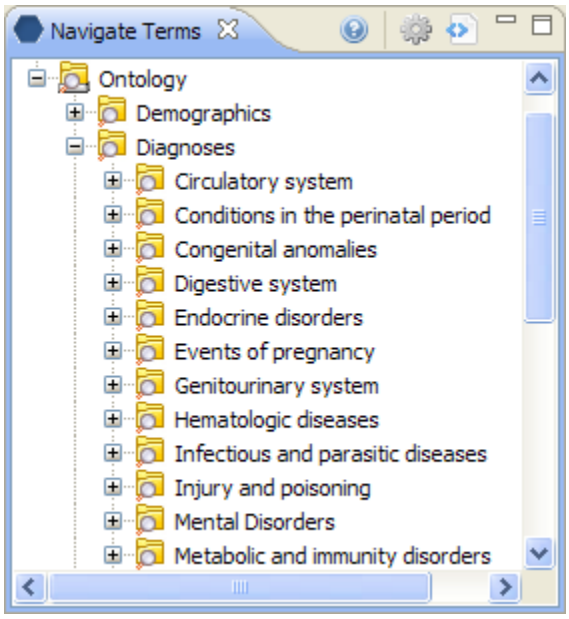
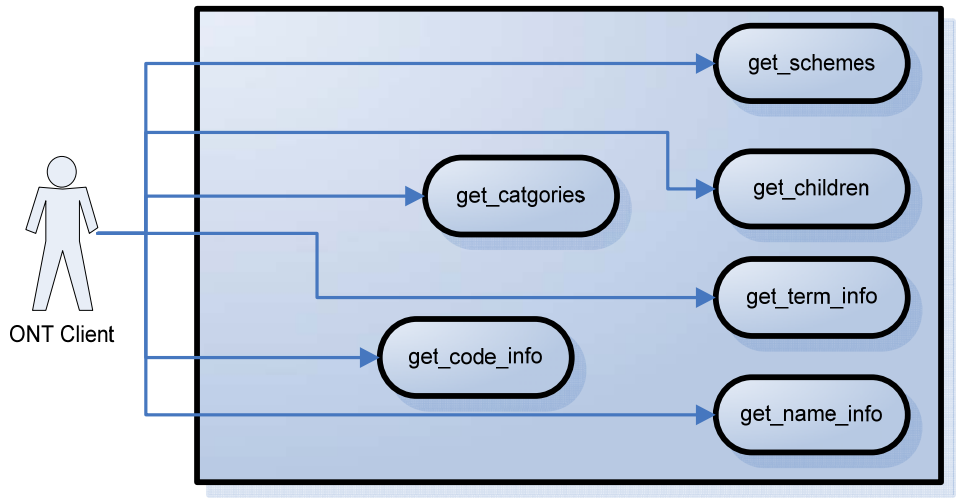


Figure 2 Navigate Terms View

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

2.1 Use Case

The diagram below depicts common use cases 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:

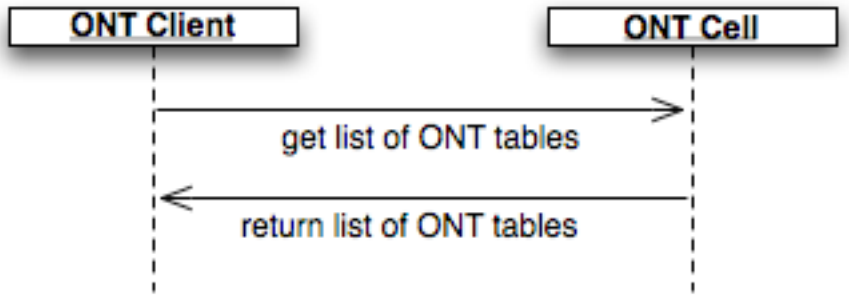
Service	Description
get_categories	Returns a list of categories available for a given user. These categories are displayed in a tree format. The top level of the tree consists of all the categories a particular user has permission to see.
get_children	Expands any level of a vocabulary category, providing 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 an existing Ontology term
update_crc_concept	Notifies the Ontology cell to synchronize metadata terms with a dimension table
get_ont_process_status	Returns status information about the dimension table synchronization process
get_dirty_state	Returns state information about the need to synchronize with the concept_dimension table

2.2 Messages

2.2.1 get_categories

A **get_categories** message returns a *role-specific* list of tables that will be displayed as roots of the Navigate Terms query tree and 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 TABLES ASSOCIATED WITH A USER

The **get_categories** message is sent by the query tool to populate the root nodes and by the find tool to populate the list of categories available to a user. It is also used internally by the other messages to help confirm that the user has table access permission.

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

1. Client requests list of tables 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 tables associated with this users' 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"/>
</message_body>
```

2.2.1.2.1 Possible "type" Settings

default = Return table name/key pairs

Example: Find Terms request

core = Return all data except system/date information

Example: Navigate Terms request

2.2.1.2.2 Possible "blob" Settings

false = Do not return data stored as a blob or clob

Example: xml, comments

true = Return xml and comments

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
```

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
```

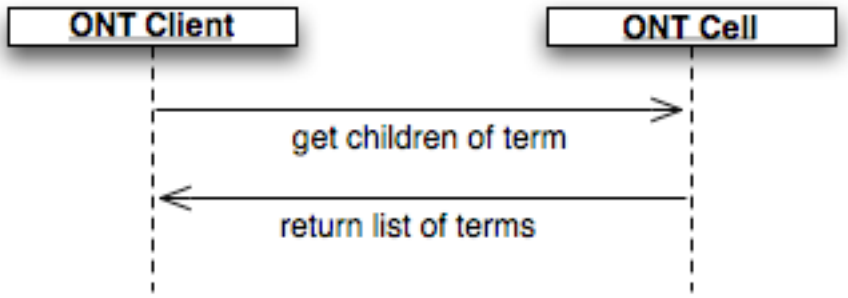
Response message:

```
<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

'**Get_Children**' returns all the children of a particular term. A client may want a list of all 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 tools. In both of these cases the table/path (root) to search are known.

The sequence of events is as follows:

1. Client sends message with "max" set to 200 or higher; "type" = default
2. ONT server performs following steps:
    c. Calls getCategories to get list of tables for user. This validates user as well.
    d. Parses <parent> to obtain the table key and path. Queries table of tables to confirm that user/role can access the table that is referenced by table key passed in. This call returns the table name referenced by that key. If not, return coded error. Client receives error message with code 'TABLE_ACCESS_DENIED'.

e. If max is set, the database is queried for that number of children associated with the parent passed in.
f. If count < max or no max set, the database is queried for the entire list of children that meets the parent criteria
g. Client receives list of children and populates tree.
h. If count > max a coded error message is sent back. Client receives error message with 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

A **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 parent is organized as follows: \\table_key\path. So 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 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_children's default set of columns include things like key, name, visual_attributes, tooltip... 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.

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

2.2.2.2.1 Possible "hiddens" Settings

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

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

2.2.2.2.2 Possible “synonyms” Settings

Some ontologies are listed as synonyms for other ontologies.

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

2.2.2.2.3 Possible “type” Settings

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

false = Do not return data stored as a blob or clob
Example: 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 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 that user (type = default)
2. The ONT server performs the following steps:

a. Call get_categories to get list of tables for user. This validates user as well.
b. Query the table of tables to confirm that user/role can access category passed in. If not, return coded error. Client receives error message with code 'TABLE_ACCESS_DENIED'.
c. If max is set, query the database for the number of entries that meet the search criteria
d. If count < max or no max is set, query the database for entries that meet the search criteria.
e. The client generates a list of nodes that match search criteria
f. If count > max send an error message back. Client receives error message with code 'MAX_EXCEEDED' and displays dialog asking if 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 max, then an error message will be 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. Each message will interpret default to be a different set of columns. The default set of columns for **get_name_info** includes items that are necessary for the visualization of the data; such as *key*, *name*, *visual_attributes*, *tooltip*.

- 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

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"/>
    <match_str strategy="contains">asthma</match_str>
    </get_name_info>
</message_body>
```

2.2.3.2.1 Possible "hiddens" Settings

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

false = Do not return data categorized as "hidden"

true = Include data categorized as "hidden"

2.2.3.2.2 Possible "synonyms" Settings

Some ontologies are listed as synonyms for other ontologies.

false = Do not return data categorized as "synonym"

true = Include data categorized as "synonym"

2.2.3.2.3 Possible "type" Settings

default = Return name information only

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

all = Return all data

2.2.3.2.4 Possible "blob" Settings

false = Do not return data stored as a blob or clob

Example: xml, comments

true = Return xml and comments.

2.2.3.2.5 Possible “strategy” Settings

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 timeline to obtain information associated with a node. The sequence of events is as follows:

1. Client sends message with type set
2. Ont server performs following steps:
3. Calls getCategories to get list of tables for user. This validates user as well.
4. Parses <self> to obtain table key and path. Queries table of tables to confirm that user/role can access table that is referenced by table key passed in. This call returns the table name referenced by that key. If not, return coded error.
5. Client receives error message with code TABLE_ACCESS_DENIED.
6. Query db for node that meets <self> criteria.
7. Client receives information about 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. If the number of rows found is greater than max, then an error message will be returned in the i2b2 header. If max is left out then it is interpreted that there is no max. The hiddens and synonyms tells 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 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 will consist of name only. 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.

```
<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>
```

2.2.4.3 POSSIBLE “HIDDENS” SETTINGS

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

false = Do not return data categorized as “hidden”

true = Include data categorized as “hidden”

2.2.4.4 POSSIBLE “SYNONYMS” SETTINGS

Some ontologies are listed as synonyms for other ontologies.

false = Do not return data categorized as “synonym”

true = Include data categorized as “synonym”

2.2.4.5 POSSIBLE “TYPE” SETTINGS

default = Return all data except system/date

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

all = Return all data

2.2.4.6 POSSIBLE “BLOB” SETTINGS

false = Do not return data stored as a blob or clob

Example: xml, comments

true = Return xml and comments.

2.2.4.7 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:

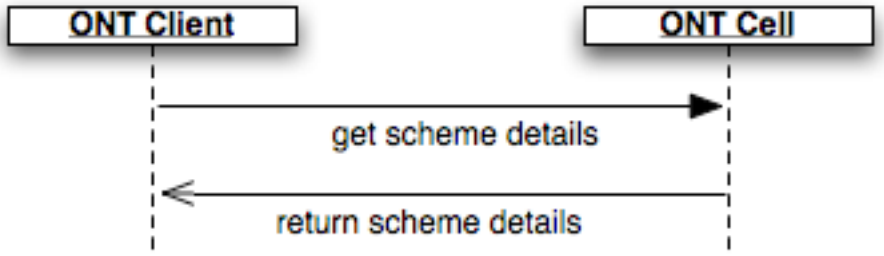
```
<message_body>
    <concepts>
        <concept>
            <level>5</level>
            <key>\\rpdr\RPDR\Labtests\LAB\(LLB16) Chemistry\(LLB31) Anemia Related
            Studies\B12USAT\BC1-107</key>
            <name>B12 unsat bind (Test:bc1-107)</name>
            <synonym_cd>N</synonym_cd>
            <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

'**get_schemes**' 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 (getSchemes type="default")
2. The ONT server performs the following steps:
   a. Get project/role available for the user from 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

default = Return key/name pairs

2.2.5.3 GET_SCHEMES RESPONSE MESSAGE

The find terms 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

‘**get_code_info**’ 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 (get_code_info type= default)
2. The ONT server performs the following steps:
    a. Calls getCategories to get valid list of table data for user. This validates user as well.
    b. Queries list of tables for entries that match scheme:code pair.

3. 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 *category attribute* will tell the service what metadata table to search in and does not have to be included. If category is not included, all tables the user is allowed to access will be searched.

The 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 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 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, 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. The default set of columns for **get_code_info** consists of name only.

- 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_code_info max=”200” 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

default = Return name only

core = Return all data except system/date information

all = Return all data

2.2.6.3 GET_CODE_INFO RESPONSE MESSAGE

The ONT request to map name to scheme: code pair 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 name 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

'**add_child**' 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. Client requests to add a leaf or folder to a given (editable) parent node.
   a. Ontology server performs following steps:
   b. Parses node information to obtain key/table_cd
   c. Queries table_access table for table name associated with table_cd.
   d. Inserts new leaf, folder or container into the Ontology metadata table

2. Client populates selected parent node with 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. Client specifies a leaf, folder or container to be deleted.
2. Ontology server performs following steps:
    a. Parses node information to obtain key/table_cd
    b. Query table_access table for table name associated with table_cd.
    c. Update metadata table to delete the corresponding node and its synonyms.
    d. If “include_children = true”; delete children of this node also.
4. Client removes 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.

```
<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. Client specifies a leaf, folder or container to be modified.
2. Ontology server performs following steps:
    a. Parses node information to obtain key/table_cd
    b. Queries table_access table for table name associated with table_cd.
    c. Updates metadata table with new information for corresponding node.
3. Client refreshes 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.
"inclSynonyms = true" indicates that no synonyms were added or removed during 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.

```
<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. Client specifies desire to begin synchronization process.
2. Ontology server performs following steps:
    a. Identifies all nodes that have been created or edited (visual_attribute contains ‘E’ in third character)
    b. Uploads a file to FRC that contains all nodes identified above.
    c. Notifies CRC to start an upload.
3. Client displays process progress.

2.2.10.2 UPDATE_CRC_CONCEPT REQUEST MESSAGE

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 dimension table 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.

```
<message_body>
   <ns6:update_crc_concept operation_type=”update_only”/>
</message_body>
```

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.

```
<message_body>
  <ns5: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 />
  </ns5:ontology_process_status>
 </message_body>
```

2.2.11 get_ont_process_status

The **get_process_status** message is sent to get status about the dimension table synchronization process.

2.2.11.1 GET CONCEPT SYNCHRONIZATION PROCESS STATUS

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

i. Client specifies a request to obtain status for a specified process_id.

ii. Ontology server returns status for specified process_id.

iii. Client updates process progress.

2.2.11.2 GET_ONT_PROCESS_STATUS REQUEST MESSAGE

This message requires the user to specify process we are obtaining status for. No additional attribute settings are necessary.

```
<message_body>
   <ns6:get_ont_process_status>
      <process_id>26</process_id>
   </ns6: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.

```
<message_body>
  <ns5: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 />
  </ns5:ontology_process_status>
</message_body>
```

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

ONT_BUILD_PDO_START	indicates that a file containing edited metadata is being created
ONT_SENTTO_FRC	indicates that metadata file has been sent to FRC
ONT_SENTTO_CRCLOADER	indicates that the CRC has been instructed to upload the file sent to the FRC

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

ERROR	indicates that an error occurred
PROCESSING	indicates that the process is in progress
COMPLETED	indicates that the process has completed.

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 DIRTY STATE

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

i. Client specifies a request to obtain dirty state information.
ii. Ontology server returns dirty state information

2.2.12.2 GET_DIRTY_STATE REQUEST MESSAGE

This message has no requirements or attribute settings.

```
<message_body>
   <ns6: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 dirty state information.

```
<message_body>
  <ns5:dirty_state>NONE|ADD|DELETE_EDIT</ns5:dirty_state>
</message_body>
```

The following **dirty state** codes are in use:

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

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 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	Flag to indicate if hidden terms should be returned.
synonyms	Flag to indicate if synonymous terms should be returned.
type	Indicates the amount of data to be returned (default/core/all).
blob	Flag to indicate if 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	Flag to indicate if hidden terms should be returned.
synonyms	Flag to indicate if synonymous terms should be returned.
type	Indicates the amount of data to be returned (default/core/all).
blob	Flag to indicate if blob/clob fields should be returned.

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

<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	Flag to indicate if blob/clob fields should be returned.

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

<add_child>	Container for term addition request
<level>	Also known as hierarchy level. 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 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".
<visualattributes>	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>	Code representation of the term.
<metadataxml>	Additional xml that may be associated with the term (optional).
<facttablecolumn>	Observation fact table column associated with this term.
<tablename>	Dimension table associated with this term.
<columnname>	Dimension table column associated with this term.
<columndatatype>	Dimension table column data type associated with this term where 'T' = text 'N' = number
<operator>	SQL comparison operator associated with this term('LIKE' or '=').
<dimcode>	Dimension code that is associated with the term.
<comment>	Optional comment associated with the term.
<tooltip>	Tooltip that is associated with the term.
<valuetype_cd>	Indicates the term type.

	'DOC' = term represents a document or note. 'LAB' = term is associated with a lab result 'KEY' = keyword associated with term (may be used for NLP)

<delete_child>	Container for term deletion request
include_children	Flag to indicate if children of this node should be deleted also
<level>	Also known as hierarchy level. 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 hierarchy of '\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma'
<name>	The name of the term.
<synonym_cd>	Indicates whether or not the term is a synonym of another term. The value can be "Y" or "N".
<basecode>	Code representation of the term.

<modify_child>	Container for term modify request
incl_synonyms	Flag to indicate if synonyms of this node should be modified also
<level>	Also known as hierarchy level. 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 hierarchy of '\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma'
<name>	The name of the term.
<synonym_cd>	Indicates whether or not the term is a synonym of another term. The value can be "Y" or "N".
<visualattributes>	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 term items found. (not typically used)
<basecode>	Code representation of the term.
<metadataxml>	Additional xml that may be associated with the term (optional).
<facttablecolumn>	Observation fact table column associated with this term.
<tablename>	Dimension table associated with this term.
<columnname>	Dimension table column associated with this term.
<columndatatype>	Dimension table column data type associated with this term where ‘T’ = text ‘N’ = number
<operator>	SQL comparison operator associated with this term (‘LIKE’ or ‘=’).
<dimcode>	Dimension code that is associated with the term.
<comment>	Optional comment associated with the term.
<tooltip>	Tooltip that is associated with the term.
<valuetype_cd>	Indicates the term type. ‘DOC’ = term represents a document or note. ‘LAB’ = term is associated with a lab result ‘KEY’ = keyword associated with term (may be used for NLP)

<update_crc_concept>	Container for synchronize dimension table request
operation_type	Indicates synchronization operation request type. ‘update_only’ = only add new terms from metadata to dimension table ‘synchronize_all’ = perform a full synchronization between metadata and dimension table

<get_ont_process_status>	Container for synchronization process status request

<process_id>	Id of the process we are requesting status for

<get_dirty_state>	Container for dirty state request

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 hierarchy level. 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 hierarchy of '\RPDR\Diagnoses\Respiratory system (460-519)\Chronic obstructive diseases (490-496)\(493) Asthma'
<name>	The name of the term.
<synonym_cd>	Indicates whether or not the term is a synonym of another term. The value can be "Y" or "N".

<visualattributes>	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>	Code representation of the term.
<metadataxml>	Additional xml that may be associated with the term (optional).
<facttablecolumn>	Observation fact table column associated with this term.
<tablename>	Dimension table associated with this term.
<columnname>	Dimension table column associated with this term.
<columndatatype>	Dimension table column data type associated with this term where ‘T’ = text ‘N’ = number
<operator>	SQL comparison operator associated with this term (‘LIKE’ or ‘=’).
<dimcode>	Dimension code that is associated with the term.
<comment>	Optional comment associated with the term.
<tooltip>	Tooltip that is associated with the term.
<valuetype_cd>	Indicates the term type. ‘DOC’ = term represents a document or note. ‘LAB’ = term is associated with a lab result

<ontology_process_status>	Container for requested synchronization process status
<process_id>	Id of the process we requested status for
<process_type_cd>	Provides information about the type of process that is currently in progress. The following codes are in use: 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: ONT_BUILD_PDO_START start process ONT_SENTTO_FRC file sent to FRC ONT_SENTTO_CRCLOADER instruct CRC to load file
<start_date>	Date the process was requested
<end_date>	Date the process was completed
<process_status_cd>	Provides information about the process step. ERROR an error occurred PROCESSING process step in progress COMPLETED process has completed.
<crc_upload_id>	Corresponding CRC upload id for the ONT_SENTTO_CRC_LOADER process step
<message>	Miscellaneous status message

<dirty_state>	Container for dirty state information
	The following **dirty state** codes are in use: 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

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>	Version associated with this data
<CreationDateTime>	Timestamp associated with this data
<DataType>	Identifies the data type. Possible values are: 'PosFloat', 'Float', 'PosInteger', 'Integer' (numeric) 'Enum' (pre-determined text) 'String' (free-form text)
<Flagstouse>	Represents the type of flag to use. Example values: 'HL' = High/Low (does not apply to String types) 'A' = abnormal
<Oktousevalues>	Indicates if it is OK to use values (Y/N). Y indicates value is a number (numeric types) N or empty indicates value is not a number
<MaxStringLength>	The maximum length associated with a String data type.
The next 6 fields 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>	Low end of toxic value range.
<HighofToxicValue>	High end of toxic value range.
This field applies to Enum types	

<EnumValues>	Container of values associated with enum data type.
<Val description>	enum data type value.
This field applies to any type	associated with a unit of measurement
<UnitValues>	Container for unit of measurement value information.
<NormalUnits>	Unit of measurement associated with value.
<EqualUnits>	An equivalent unit of measurement associated with value.
<ConvertingUnits>	Container for unit conversion information.
<Units>	Conversion unit of measurement.
<MultiplyingFactor>	Conversion unit multiplication factor.