Informatics for Integrating Biology and the Bedside

i2b2 Cell Messaging

Workplace Framework (WORK) Cell

Document Version: *1.7.08-001*

i2b2 Software Version: *1.7.08*

TABLE OF CONTENTS

DOCUMENT MANAGEMENT

Revision Number	Date	Author	Description of change
1.7.0	03/19/13	Janice Donahoe	Created 1.7 version of document.
1.7.00-001	08/14/2015	Janice Donahoe	Fixed minor spelling and grammar issues. Also fixed the issue in the Introduction section where it was stating this document was about the Ontology Management cell. This has been updated to accurately reflect the Workplace Management cell.
1.7.08-002	07/25/2016	S. Wayne Chan	Added 4 DBlookup Messages. Corrected several minor misnamed ('get_name_info' to 'set_protected_access', 'WORK_QRY.xsd' to 'WORK_RESP.xsd', etc.) section titles.
1.7.08-003	10/06/2016	Janice Donahoe	Expanded the documentation 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 **Workplace Framework (WORK) 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.

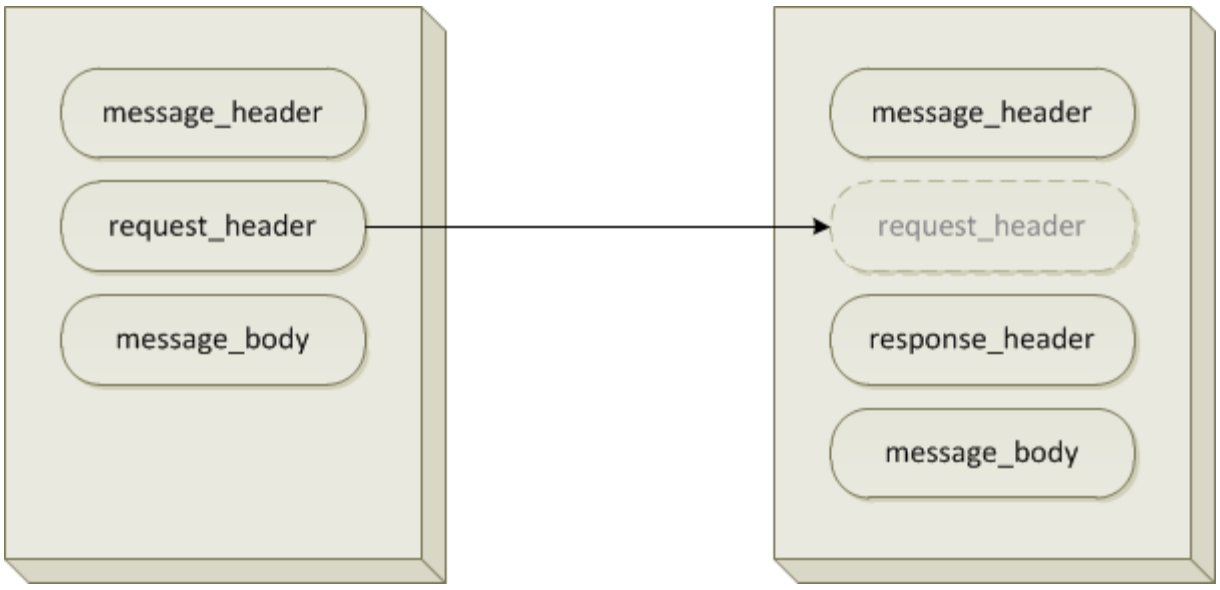
1.2 i2b2 Messaging Overview

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 Identity Management (IM) Cell.

2 WORKPLACE (WORK) CELL MESSAGING DETAIL

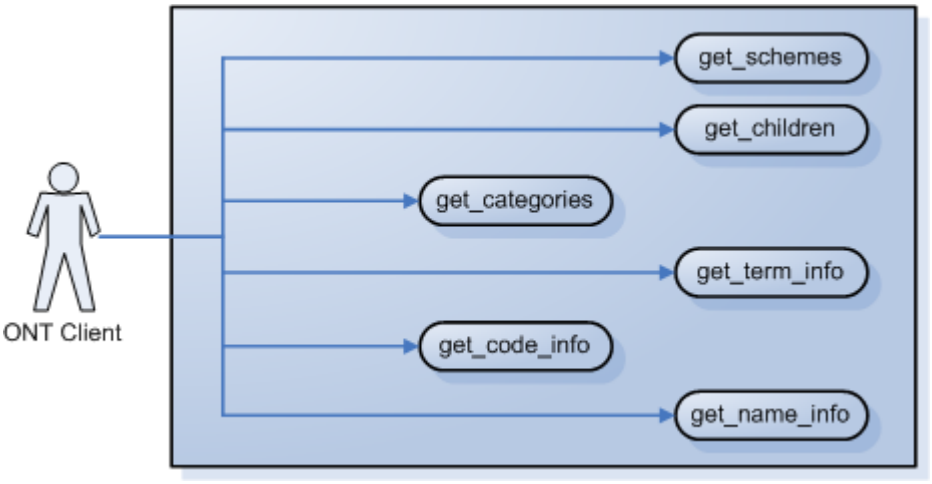
The Workplace Framework (WORK) Cell is an optional i2b2 Hive cell. This cell manages project specific XML data objects for users of a given project. The project specific XML data objects originate in other views or cells, such as Ontology or Previous Query and are stored in the WORK cell as a convenience.

Project specific XML data objects in the WORK cell are organized in hierarchical structures that represent the relationships between elements. 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*".

The WORK cell both accepts new XML data objects for storage and provides a listing of those items previously stored. It also allows the user to organize, label and annotate the stored data objects however they choose to.

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

Service	Description
get_folder_by_userid	Returns a list of root folders available for a given user. These folders are displayed in a tree format. The top level of the tree consists of all the folders a particular user has permission to see, which is determined by his / her role.
get_folder_by_project	Returns a list of root folders available for a given project across all users in the project. These folders are displayed in a tree format. A user must have manager permission and be configured as such to see all folders in a project.
get_children	Expands any level of a folder, providing information about its contents, for a given user.
rename_child	Renames a work item (leaf or folder) in the workplace tree.
delete_child	Deletes a work item (leaf or folder) in the workplace tree.
add_child	Adds a work item (leaf or folder) in the workplace tree.
annotate_child	Annotates a work item (leaf or folder) in the workplace tree by modifying or adding a tooltip.
move_child	Moves a work item (leaf or folder) in the workplace tree and reassigns all its children.
export_child	Export a work item (leaf or folder) as XML to a local file.
get_name_info	Returns information needed about all work item related to a given search keyword or name.
set_protected_access	This webservice will let users set the protected access on any folder or any contents of the folder. If user sets protected access to a folder, all the contents of the folder will recursively get protected access.
get_all_dblookups	Returns a list of all the entries (rows) in the WORK_DB_LOOKUP table.
set_dblookup	Adds or updates a specific entry (row) in the WORK_DB_LOOKUP table.
get_dblookup	Returns all the data for a specific entry (row) in the WORK_DB_LOOKUP table.
delete_dblookup	Deletes a specific entry (row) in the WORK_DB_LOOKUP database table.

Note

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

2.1.2 Database Lookup Services / Messages

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

Service	XML Message
WorkplaceService/**getAllDblookups**	get_all_dblookups
WorkplaceService/**setDblookup**	set_dblookup
WorkplaceService/**getDblookup**	get_dblookup
WorkplaceService/**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 Workplace 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_folder_by_userid

A **get_folder_by_userid** message returns a list of folders that will be displayed as roots of the Workplace tree. 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 Folders Associated with a User

The **get_folder_by_userid** message is sent by the workplace view to populate the root nodes 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_folder_by_userid)

   Request type = core

2. The WORK 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 WORKPLACE_ACCESS table for the list of folders associated with this user for the project they are logged into.

3. The client maps the list of folders to the Workplace root node.

2.2.1.2 get_folder_by_userid Request Message

```
<message_body>
    <work:get_folders_by_userId type="core"/>
</message_body>
```

2.2.1.2.1 Possible “type” Settings

Value	Description
core	Return all data except system / date information

2.2.1.3 get_folder_by_userid Response Message

Response Message:

```
<message_body>
    <folders>
        <folder>
            <name>LCP</name>
            <index>\\asthma\1</index>
            <parentIndex/>
            <visualAttributes>CA</visualAttributes>
            <groupId>asthma</groupId>
            <shareId/>
            <statusCd/>
            <userId>lcp</userId>
            <workXml/>
            <workXmlSchema/>
            <workXmlI2B2Type/>
        </folder>
    </folders>
</message_body>
```

2.2.2 get_folder_by_project

A **get_folder_by_project** message returns a list of folders that will be displayed as roots of the Workplace tree. 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**.

2.2.2.1 Get a List of Folders Associated with a Project

The **get_folder_by_project** message is sent by the workplace view to populate the root nodes of all users in a project. This feature is accessible by users with the role of *MANAGER*.

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_folder_by_project)

   Request type = core

2. The WORK 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 WORKPLACE_ACCESS table for the list of folders associated with this project.

3. The client maps the list of folders to the Workplace root node.

2.2.2.2 get_folder_by_project Request Message

```
<message_body>
    <work:get_folders_by_project type="core"/>
</message_body>
```

2.2.2.2.1 Possible “type” Settings

Value	Description
core	Return all data except system / date information

2.2.2.3 get_folder_by_userid Response Message

Response Message:

```
<folders>
    <folder>
        <name>LCP</name>
        <index>\\asthma\1</index>
        <parentIndex/>
        <visualAttributes>CA</visualAttributes>
        <tooltip>\LCP</tooltip>
        <groupId>Asthma</groupId>
        <shareId/>
        <statusCd/>
        <userId>lcp</userId>
        <workXml/>
        <workXmlSchema/>
        <workXmlI2B2Type/>
    </folder>
    <folder>
        <name>MEM</name>
        <index>\\asthma\10</index>
        <parentIndex/>
        <visualAttributes>CA</visualAttributes>
        <tooltip>\MEM</tooltip>
        <groupId>Asthma</groupId>
        <shareId/>
        <statusCd/>
        <userId>mem</userId>
        <workXml/>
        <workXmlSchema/>
        <workXmlI2B2Type/>
    </folder>
</folders>
```

2.2.3 get_children

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

2.2.3.1 Populating Children of Tree Nodes

The **get_children** message is used to populate tree nodes in the Workplace view. In both of these cases the table / index (root) to search are known.

The sequence of events is as follows:

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

   type = core

   blob = true

2. WORK server performs the following steps:

   a. Parses <*parent*> to obtain the table_cd and index. Queries WORKPLACE_ACCESS table for the table name associated with the table_cd.

   b. Query WORKPLACE table returned above for all entries with parent_index = index.

3. Client receives a list of children and populates the tree.

2.2.3.2 get_children Request Message

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

\\table_key\index

So <parent>\\asthma\22</parent> equates to the metadata table that maps to the key "*asthma*" plus the index "*22*" to search for.

Attributes provide information about the results to be returned. The "*blob*" attribute indicates whether or not to return the blob and for this feature must be set to true.

```
<message_body>
    <get_children blob="true">
        <parent>\\asthma\22</parent>
    </get_children>
</message_body>
```

2.2.3.2.1 Possible "blob" Settings

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

> **Important**
>
> The "blob" attribute must be set to **true** for this feature.

2.2.3.3 get_children Response Message

The request has the following settings:

blob=true

Response Message:

```
<message_body>
    <folders>
        <folder>
            <name>CONCEPTS</name>
            <index>\\asthma\2</index>
            <parentIndex>1</parentIndex>
            <visualAttributes>FA</visualAttributes>
            <tooltip>FOLDER:Concepts</tooltip>
            <groupId>Demo</groupId>
            <shareId/>
            <statusCd/>
            <userId>lcp</userId>
            <workXml/>
            <workXmlSchema/>
            <workXmlI2B2Type/>
        </folder>
        <folder>
            <name>Patient Sets</name>
            <index>\\asthma\8</index>
            <parentIndex>1</parentIndex>
            <visualAttributes>FA</visualAttributes>
            <tooltip>FOLDER:Patient Sets</tooltip>
            <groupId>Demo</groupId>
            <shareId/>
            <statusCd/>
            <userId>lcp</userId>
            <workXml/>
            <workXmlSchema/>
            <workXmlI2B2Type/>
        </folder>
    </folders>
</message_body>
```

2.2.4 rename_child

The **rename_child** message is sent to rename a leaf or folder in a tree. This message requires the user to pass a string that represents the new name for the leaf or folder.

2.2.4.1 Rename a Node in the Tree

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

1. The client provides a new name for a given leaf or folder
2. WORK server performs the following steps:
    a. Parses the leaf or folder index to obtain the key / table_cd
    b. Query WORKPLACE_ACCESS table for table name associated with the table_cd.
    c. Update the workplace table with the new name for the leaf or folder.
3. Client renames the leaf or folder.

2.2.4.2 rename_child Request Message

This message requires the user to pass a string that represents the new name for the leaf or folder. No additional attribute settings are necessary.

```
<message_body>
    <work:rename_child>
        <node>\\asthma\55 </node>
        <name>Juvenile Asthma Set 1</name>
    </work:rename_child>
</message_body>
```

2.2.4.3 rename_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.

2.2.5 delete_child

The **delete_child** message is sent to mark a leaf in a folder or the folder itself for deletion. (c_status_cd = 'D') Deleting a folder will cause its children to be marked for deletion as well.

2.2.5.1 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 or folder to be marked for deletion.
2. WORK server performs the following steps:
    a. Parses the leaf or folder index to obtain the key / table_cd
    b. Query WORKPLACE_ACCESS table for the table name associated with the table_cd.
    c. Update the WORKPLACE table with to mark node corresponding to specified index for deletion.
3. Client removes the leaf or folder.

2.2.5.2 delete_child Request Message

This message requires the user to specify the leaf or folder to be marked for deletion. No additional attribute settings are necessary.

```
<message_body>
    <work:delete_child>
        <node>\\asthma\55 </node>
    </work:delete_child>
</message_body>
```

2.2.5.3 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.

2.2.6 add_child

The **add_child** message is sent to add a new leaf or folder to a given folder.

2.2.6.1 Add a Node in 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 folder.
2. WORK server performs the following steps:
    a. Parses the leaf or folder index to obtain the key / table_cd
    b. Query WORKPLACE_ACCESS table for the table name associated with the table_cd.
    c. Generates a new index for the new leaf or folder.
    d. Insert the new leaf or folder into the WORKPLACE table.
3. Client populates the selected folder with the new leaf or folder.

2.2.6.2 add_child Request Message

This message requires the user to specify the leaf or folder to be added. No additional attribute settings are necessary.

```
<message_body>
```

```
    <work:add_child>
        <name>Circulatory system</name>
        <user_id>lcp</user_id>
        <group_id>Asthma</group_id>
        <index>1oBqNGe4mGB8291gNZlg</index>
        <parent_index>\\asthma\25</parent_index>
        <visual_attributes>LA</visual_attributes>
        <tooltip>CONCEPT:Circulatory system</tooltip>
        <work_xml>
            <plugin_drag_drop xmlns:ns4="http://www.i2b2.org/xsd/cell/ont/1.1/">
                <concepts>
                    <concept>
                        <level>2</level>
                        <key>\\rpdr\RPDR\Diagnoses\Circulatory system (390-459)</key>
                        <name>Circulatory system</name>
                        <synonym_cd>N</synonym_cd>
                        <visualattributes>FA</visualattributes>
                        <totalnum>0</totalnum>
                        <basecode>L_!3</basecode>
                        <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)</dimcode>
                        <comment />
                        <tooltip>Diagnoses \ Circulatory system</tooltip>
                    </concept>
                </concepts>
            </plugin_drag_drop>
        </work_xml>
        <work_xml_i2b2_type>CONCEPT</work_xml_i2b2_type>
    </work:add_child>
</message_body>
```

2.2.6.3 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.

2.2.7 annotate_child

The **annotate_child** message is sent to rename a leaf or folder in a tree. This message requires the user to pass a string that represents the new name for the leaf or the folder.

2.2.7.1 Annotate a Node in the Tree

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

1. The client provides a new tooltip for a given leaf or folder.
2. WORK server performs the following steps:
    a. Parses the leaf or folder index to obtain the key / table_cd
    b. Query WORKPLACE_ACCESS table for the table name associated with the table_cd.
    c. Update the WORKPLACE table with the new name for the leaf or folder.
3. Client annotates the leaf or folder with the new tooltip.

2.2.7.2 annotate_child Request Message

This message requires the user to pass a string that represents the new tooltip for the leaf or folder. No additional attribute settings are necessary.

```
<message_body>
    <work:annotate_child>
        <node>\\asthma\55 </node>
        <tooltip>Juvenile Asthma Set 1</tooltip>
    </work:annotate_child>
</message_body>
```

2.2.7.3 annotate_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 move_child

The **move_child** message is sent to move the location of a folder in a tree. This message requires the user to specify the new parent for the leaf or folder.

2.2.8.1 Move a Folder in the Tree

To move a folder in the tree, the sequence of events is as follows:

1. The client provides a new parent for a given folder.
2. WORK server performs the following steps:
    a. Parses the folder index to obtain the key / table_cd
    b. Query WORKPLACE_ACCESS table for the table name associated with the table_cd.
    c. Update the WORKPLACE table with the new parent_index for the folder.
3. Client moves the folder and updates the tree content.

2.2.8.2 move_child Request Message

This message requires the user to specify the new parent for the leaf or folder. No additional attribute settings are necessary.

```
<message_body>
    <work:move_child>
```

```
            <node>\\asthma\55</node>
            <parent>22</parent>
        </work:move_child>
    </message_body>
```

2.2.8.3 move_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 export_child

The **export_child** message is sent to export as xml the content of a node in a tree. This message requires the user to specify the id for the node.

2.2.9.1 Exporting Child in the Tree

The **export_child** message is used to export the request xml or analysis xml from the tree nodes in the Workplace view. In both of these cases the table / index (root to search are known.

To export an item, the sequence of events is as follows:

1. The client sends a message with one of the following types:

   Request type = QM for Query Master or QR for Analysis Breakdown

2. The WORK server performs the following steps:

   a. Queries the CRC cell requesting either the Query Master or the Analysis Breakdown.

3. The client receives the request XML received from the CRC.

2.2.9.2 export_child Request Message

The **export_child** message implies that the user is passing a key / index for a parent and wants the XML returned.

```
<message_body>
    <export_child node="667737" type="QM" />
</message_body>
```

2.2.9.3 export_child Response Message

Response Message:

```
<ns5:response xmlns:ns2="http://www.i2b2.org/xsd/hive/pdo/1.1/"
xmlns:ns4="http://www.i2b2.org/xsd/cell/crc/psm/1.1/"
xmlns:ns3="http://www.i2b2.org/xsd/cell/crc/pdo/1.1/"
xmlns:tns="http://axis2.crc.i2b2.harvard.edu"
xmlns:ns9="http://www.i2b2.org/xsd/cell/pm/1.1/"
xmlns:ns5="http://www.i2b2.org/xsd/hive/msg/1.1/"
xmlns:ns6="http://www.i2b2.org/xsd/hive/msg/result/1.1/"
xmlns:ns7="http://www.i2b2.org/xsd/cell/crc/psm/querydefinition/1.1/"
xmlns:ns10="http://www.i2b2.org/xsd/cell/ont/1.1/"
xmlns:ns8="http://www.i2b2.org/xsd/cell/crc/psm/analysisdefinition/1.1/">
    <message_header>
        <i2b2_version_compatible>1.1</i2b2_version_compatible>
        <hl7_version_compatible>2.4</hl7_version_compatible>
        <sending_application>
            <application_name>CRC Cell</application_name>
            <application_version>1.6</application_version>
        </sending_application>
        <sending_facility>
            <facility_name>i2b2 Hive</facility_name>
        </sending_facility>
        <receiving_application>
            <application_name>Workplace Cell</application_name>
            <application_version>1.6</application_version>
        </receiving_application>
        <receiving_facility>
            <facility_name>i2b2 Hive</facility_name>
        </receiving_facility>
```

```
        <message_control_id>
            <instance_num>1</instance_num>
        </message_control_id>
        <project_id>i2b2demodata</project_id>
    </message_header>
    <response_header>
        <info>Log information</info>
        <result_status>
            <status type="DONE">DONE</status>
            <polling_url interval_ms="100" />
        </result_status>
    </response_header>
    <message_body>
        <ns4:response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns4:master_responseType">
            <status>
                <condition type="DONE">DONE</condition>
            </status>
            <query_master>
                <query_master_id>667737</query_master_id>
                <name>Princip-Severit@16:46:40</name>
                <user_id>demo</user_id>
                <request_xml>
                    <ns3:query_definition
xmlns:ns2="http://www.i2b2.org/xsd/cell/crc/psm/1.1/"
xmlns:ns4="http://www.i2b2.org/xsd/cell/crc/psm/analysisdefinition/1.1/"
xmlns:ns3="http://www.i2b2.org/xsd/cell/crc/psm/querydefinition/1.1/">
                        <query_name>Princip-Severit@16:46:40</query_name>
                        <query_timing>ANY</query_timing>
                        <specificity_scale>0</specificity_scale>
                        <panel>
                            <panel_number>1</panel_number>
                            <panel_timing>ANY</panel_timing>
                            <panel_accuracy_scale>0</panel_accuracy_scale>
                            <invert>0</invert>
                            <total_item_occurrences>1</total_item_occurrences>
                            ....
                        </panel>
                    </ns3:query_definition>
                </request_xml>
            </query_master>
        </ns4:response>
    </message_body>
</ns5:response>


<message_body>
```

```
        <folders>
            <folder>
                <name>CONCEPTS</name>
                <index>\\asthma\2</index>
                <parentIndex>1</parentIndex>
                <visualAttributes>FA</visualAttributes>
                <tooltip>FOLDER:Concepts</tooltip>
                <groupId>Demo</groupId>
                <shareId/>
                <statusCd/>
                <userId>lcp</userId>
                <workXml/>
                <workXmlSchema/>
                <workXmlI2B2Type/>
            </folder>
            <folder>
                <name>Patient Sets</name>
                <index>\\asthma\8</index>
                <parentIndex>1</parentIndex>
                <visualAttributes>FA</visualAttributes>
                <tooltip>FOLDER:Patient Sets</tooltip>
                <groupId>Demo</groupId>
                <shareId/>
                <statusCd/>
                <userId>lcp</userId>
                <workXml/>
                <workXmlSchema/>
                <workXmlI2B2Type/>
            </folder>
        </folders>
    </message_body>
```

2.2.10 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.

2.2.10.1 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 WORK 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.

2.2.10.2 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. 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="demo" max="200""hiddens="true" type="core" blob =
"false">
        <match_str strategy="contains">asthma</match_str>
    </get_name_info>
</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.

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

2.2.10.2.2 Possible “blob” Settings

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

2.2.10.2.3 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

2.2.10.3 get_name_info Response Message

The request has the following settings:

type=core

blob=false

Example:

```
<message_body>
    <ns4:folders>
        <folder>
            <name>11 years old demo</name>
            <user_id>demo</user_id>
            <group_id>Demo</group_id>
            <share_id>N</share_id>
            <index>\\demo\u5eHaJVYD2iaZ5LJu7Tw9</index>
            <parent_index>c6Y5JaJig614lNQNPSob8</parent_index>
            <visual_attributes>ZA</visual_attributes>
            <tooltip>Demo - 11 years old demo</tooltip>
            <work_xml>
                <ns2:plugin_drag_drop>
                    <ns5:concepts>
                        <concept>
                            <level>4</level>
                            <key>\\i2b2_DEMO\demo\Demographics\Age\10-17 years old\11
years old demo\</key>
```

```
                        <name>11 years old demo</name>
                        <synonym_cd>N</synonym_cd>
                        <visualattributes>LA</visualattributes>
                        <totalnum>undefined</totalnum>
                        <basecode />
                        <facttablecolumn>patient_num</facttablecolumn>
                        <tablename>patient_dimension</tablename>
                        <columnname>birth_date</columnname>
                        <columndatatype>N</columndatatype>
                        <operator>BETWEEN</operator>
                        <dimcode>getdate() - (365.25 *12) +1 AND getdate() - (365.25
* 11) + 1</dimcode>
                        <comment />
                        <tooltip>Demographics \ Age \ 10-17 years old \ 11 years old
demo</tooltip>
                    </concept>
                </ns5:concepts>
            </ns2:plugin_drag_drop>
        </work_xml>
        <work_xml_i2b2_type>CONCEPT</work_xml_i2b2_type>
    </folder>
  </ns4:folders>
</message_body>
```

2.2.11 set_protected_access

The **set_protected_access** message will let users set the protected access on any folder or any contents of the folder. If user sets protected access to a folder, all the contents of the folder will recursively get protected access.

2.2.11.1 set_protected_access Request Message

This message requires the user to pass an index that is the parent or item, and if they want the item protected.

Who can set protected access:

1. User should have a role of “DATA_PROT” in order to set protected access
2. If user is a manager then he or she can set protected access on any content as long as they belong to his/her project id

3. If user is not manager then user can set protected to access to either shared content or his own content.

Index: It should have table code (given in workplace access table) and the index of the content

protectedAccess: This element can be set to true or false

```
<ns4:set_protected_access>
    <index>\\demo\u5eHaJVYD2iaZ5LJu7Tu0</index>
    <protectedAccess>true</protectedAccess>
</ns4:set_protected_access>
```

2.2.11.2 set_protected_access Response Message

Client will need to call 'get_children' service once 'setProtectedAccess' returns a response. This is to ensure that client displays the correct colors for the protected access content. Client can set the color based on the get_children service response which should have following:

```
<protected_access>Y</protected_access>
```

Example:

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

2.2.12 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 Workplace cell.

2.2.12.1 Processing by the Workplace Cell

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

STEP 1: Send Request Message

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

STEP 2: Verify User Access

The Workplace 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 Workplace 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 **WORK_DB_LOOKUP** table that meet the following criteria.

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

   Example:

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

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

STEP 3: Send Response Message

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

2.2.12.2 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/WorkplaceService/get
AllDblookups</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>
```

2.2.12.2.1 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/WorkplaceService/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

2.2.12.3 Request Message: get_all_dblookups

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

2.2.12.3.1 Attributes

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

Attribute Name	Description
type	Always set to "default".

2.2.12.3.2 XML Elements

<get_all_dblookups>	Container for get_all_dblookups data request

2.2.12.4 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>
```

2.2.12.4.1 XML Elements

<dblookups>	**Container that wraps the <dblookup> container returned in the response message.** **The service will return all records in the WORK_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 WORK_DB_LOOKUP table.**
project_path	Contains the data stored in the **c_project_path** column in the WORK_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 WORK_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 WORK_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 WORK_DB_LOOKUP table. The name of the Workplace database / schema is stored in this column.
<db_datasource>	Contains the data stored in the **c_datasource** column in the WORK_DB_LOOKUP table. The data source for this project is stored in this column. The value represents the connection configuration for the Workplace Cell to communicate with the database.
<db_servertype>	Contains the data stored in the **c_servertype** column in the WORK_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 WORK_DB_LOOKUP table. A simple name that used to easily identify the database is stored in this column.

Tip

Please refer to the *Workplace_Architecture* document for additional information about the WORK_DB_LOOKUP table in the i2b2 Database.

2.2.12.5 Use Cases

2.2.12.5.1 ADMIN User Requests All DB Lookups

The get_all_dblookups service sends a request message to the Workplace 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 WORK_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/WorkplaceService/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">Workplace 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>
```

2.2.12.5.2 Non-ADMIN User Requests All DB Lookups

The get_all_dblookups service sends a request message to the Workplace 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 WORK_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/WorkplaceService/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>
```

2.2.13 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.

2.2.13.1 Processing by the Workplace Cell

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

STEP 1: Verify User Access

The Workplace 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 Workplace 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 WORK_DB_LOOKUP table or ***add a new row***. The criteria to find an existing entry in the WORK_DB_LOOKUP table.

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

   Example:

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

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

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

   Example:

C_PROJECT_PATH value *(WORK_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 Workplace sends a response message to the Client. The message simply lets the Client know the process has been completed.

2.2.13.2 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/WorkplaceService/set
Dblookup</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">Workplace processing completed</status>
        </result_status>
    </response_header>
    <message_body>
        ...
    </message_body>
</ns5:response>
```

2.2.13.2.1 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/WorkplaceService/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

2.2.13.3 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>
```

2.2.13.3.1 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 WORK_DB_LOOKUP table.

2.2.13.3.2 XML Elements

<dblookup>	**Container that wraps an object which holds the data for a single record (row) in the WORK_DB_LOOKUP table.**
project_path	Contains the data stored in the **c_project_path** column in the WORK_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 WORK_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 WORK_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 WORK_DB_LOOKUP table. The name of the Workplace database / schema is stored in this column.
<db_datasource>	Contains the data stored in the **c_datasource** column in the WORK_DB_LOOKUP table. The data source for this project is stored in this column. The value represents the connection configuration for the Workplace Cell to communicate with the database.
<db_servertype>	Contains the data stored in the **c_servertype** column in the WORK_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 WORK_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 WORK_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 WORK_DB_LOOKUP table.
<status_cd>	Contains the data stored in the **c_status_cd** column in the WORK_DB_LOOKUP table.

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

2.2.13.3.3 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 Workplace.

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

2.2.13.4 Response Message: set_dblookup

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

2.2.13.5 Use Cases

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

The **set_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/setDblookup</r
edirect_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">Workplace processing completed</status>
            </result_status>
        </response_header>
    </ns5:response>
```

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

The **set_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/setDblookup</r
edirect_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">Workplace processing completed</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.13.5.3 Required Information Not Sent in Request

The **set_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/setDblookup</r
edirect_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>
```

2.2.14 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.

2.2.14.1 Processing by the Workplace Cell

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

STEP 1: Verify User Access

The Workplace 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 Workplace 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 WORK_DB_LOOKUP table. The criteria to find an existing entry in the WORK_DB_LOOKUP table.

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

   Example:

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

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

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

   Example:

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

STEP 3: Send Response Message

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

2.2.14.2 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/WorkplaceService/get
Dblookup</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">Workplace processing completed</status>
        </result_status>
    </response_header>
    <message_body>
        ...
    </message_body>
</ns5:response>
```

2.2.14.2.1 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/WorkplaceService/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

2.2.14.3 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 Workplace Cell.

2.2.14.3.1 Attributes

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

Attribute Name	Description
field	The "field" is equivalent to the column name in the WORK_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 WORK_DB_LOOKUP table.
value	The specific data in the specified field to look for when querying the WORK_DB_LOOKUP table.

2.2.14.3.2 XML Elements

<get_dblookup>	Container for get_dblookup data request

Tip

Please refer to the *Workplace_Architecture* document for additional information about the WORK_DB_LOOKUP table in the i2b2 Database.

2.2.14.3.3 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 Workplace.

- 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 Workplace Cell:

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

2.2.14.4 Response Message: get_dblookup

```
<response_header>
    <result_status>
        <status type="DONE">Workplace 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>
```

2.2.14.4.1 XML Elements

<dblookups>	**Container that wraps the <dblookup> container returned in the response message.** **The service will return all records in the WORK_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 WORK_DB_LOOKUP table.**
project_path	Contains the data stored in the **c_project_path** column in the WORK_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 WORK_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 WORK_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 WORK_DB_LOOKUP table. The name of the Workplace database / schema is stored in this column.
<db_datasource>	Contains the data stored in the **c_datasource** column in the WORK_DB_LOOKUP table. The data source for this project is stored in this column. The value represents the connection configuration for the Workplace Cell to communicate with the database.
<db_servertype>	Contains the data stored in the **c_servertype** column in the WORK_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 WORK_DB_LOOKUP table. A simple name that used to easily identify the database is stored in this column.

Tip

Please refer to the *Workplace_Architecture* document for additional information about the WORK_DB_LOOKUP table in the i2b2 Database.

2.2.14.5 Use Cases

2.2.14.5.1 ADMIN User Requests Data on a Specific Database Connection

The **get_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/getDblookup</r
edirect_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">Workplace 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>
```

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

The **get_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/getDblookup</r
edirect_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>
```

2.2.14.5.3 Requested Data Not Found

The **get_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/getDblookup</r
edirect_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! - Workplace processing
completed</status>
    </result_status>
```

```
</response_header>
```

2.2.14.5.4 Required Information Not Sent in Request

The **get_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/getDblookup</r
edirect_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>
```

2.2.15 delete_dblookup

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

2.2.15.1 Processing by the Workplace Cell

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

STEP 1: Verify User Access

The Workplace 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 Workplace 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 WORK_DB_LOOKUP table. The criteria to find an existing entry in the WORK_DB_LOOKUP table.

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

   Example:

C_DOMAIN_ID value		*<delete_dblookup* **domain_id=""> value**

(WORK_DB_LOOKUP Table)		*(request message)*
i2b2demo	**=**	i2b2demo

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

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

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

   Example:

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

STEP 3: Send Response Message

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

2.2.15.2 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/WorkplaceService/del
eteDblookup</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">Workplace processing completed</status>
        </result_status>
    </response_header>
    <message_body>
        ...
    </message_body>
</ns5:response>
```

2.2.15.2.1 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/WorkplaceService/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

2.2.15.3 Request Message: delete_dblookup

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

2.2.15.3.1 Attributes

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

Attribute Name	Description
project_path	Equivalent to the *c_project_path* column in the WORK_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 WORK_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 WORK_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>
```

2.2.15.3.2 XML Elements

<delete_dblookup>	Container for delete_dblookup data request

Tip

Please refer to the *Workplace_Architecture* document for additional information about the WORK_DB_LOOKUP table in the i2b2 Database.

2.2.15.3.3 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 Workplace.

- project_path
- domain_id
- owner_id

2.2.15.4 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>
```

2.2.15.5 Use Cases

2.2.15.5.1 ADMIN User Requests Deletion of Record

The **delete_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/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">Workplace processing completed</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.2.15.5.2 Non-ADMIN User Requests Deletion of Record

The **delete_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/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>
```

2.2.15.5.3 Requested Record Doesn't Exist

The **delete_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/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)! - Workplace processing completed</status>
    </result_status>
</response_header>
```

2.2.15.5.4 Required Information Not Sent in Request

The **delete_dblookup** service sends a request message to the Workplace 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 WORK_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/WorkplaceService/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>
```

3 WORK CELL XML SCHEMA DEFINITIONS

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

3.1 WORK.xsd

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

3.2 WORK_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 workplace tables.

3.3 WORK_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 WORK_QRY.xsd

<request>	Container for request information

<get_folder_by_userID>	Container for work data request
type	Indicates the amount of data to be returned (core).

<get_folder_by_project>	Container for work data request
type	Indicates the amount of data to be returned (core).

<get_children>	Container for work data request
<parent>	The name of the parent node at which the request is targeted.

<rename_child>	Container for work data request
<node>	The index of the node at which the request is targeted.
<name>	The new name for the node.

<export_child>	Container for work data request
<node>	The index of the node at which the request is targeted.
<type>	The new type for the node.

<delete_child>	Container for work data request
<node>	The index of the node at which the request is targeted.

<add_child>	Container for work data request; identifies the information to insert into a row in the WORKPLACE table.

<annotate_child>	Container for work data request
<node>	The index of the node at which the request is targeted.
<tooltip>	The new annotation for the node.

<move_child>	Container for work data request
<node>	The index of the node at which the request is targeted.
<parent>	The new parent index for the node at which the request is targeted.

<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 WORK_DB_LOOKUP table.
project_path	Contains the data stored in the **c_project_path** column in the WORK_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 WORK_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 WORK_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 WORK_DB_LOOKUP table. The name of the Workplace database / schema is stored in this column.
<db_datasource>	Contains the data stored in the **c_datasource** column in the WORK_DB_LOOKUP table.

	The data source for this project is stored in this column. The value represents the connection configuration for the Workplace Cell to communicate with the database.
<db_servertype>	Contains the data stored in the **c_servertype** column in the WORK_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 WORK_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 WORK_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 WORK_DB_LOOKUP table.
<status_cd>	Contains the data stored in the **c_status_cd** column in the WORK_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 WORK_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 WORK_DB_LOOKUP table.
value	The specific data in the specified field to look for when querying the WORK_DB_LOOKUP table.

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

4.1.2 WORK_RESP.xsd

<response>	Container for response information

<folders>	Container list of folders returned in a response

<folder>	Container for a folder returned in a response; an object that holds information from rows in a WORKPLACE table.
<name>	The name of the workplace item.
<index>	A "\\table code\unique index" pair. Example: \\i2b2\1 In the above example, "i2b2" equates to the table code and a unique index of "1".
<parentIndex>	The index of the parent of this workplace item.
<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 "O" (open) or null
<tooltip>	A tooltip that is associated with the workplace item.
<groupID>	The project or group ID associated with the workplace item.
<shareID>	Indicates whether or not the workplace item is shared (Y/N).
<statusCd>	Indicates the status of the workplace item ("D" = deleted).
<userID>	The user ID of the originator of the workplace item.
<workXmlSchema>	The schema associated with <workXml>.
<workXmlI2B2Type>	The i2b2 type associated with the drag and drop XML.

<dblookups>	**Container that wraps the <dblookup> container returned in the response message.** **The service will return all records in the WORK_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 WORK_DB_LOOKUP table.**
project_path	Contains the data stored in the **c_project_path** column in the WORK_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 WORK_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 WORK_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 WORK_DB_LOOKUP table. The name of the Workplace database / schema is stored in this column.
<db_datasource>	Contains the data stored in the **c_datasource** column in the WORK_DB_LOOKUP table. The data source for this project is stored in this column. The value represents the connection configuration for the Workplace Cell to communicate with the database.
<db_servertype>	Contains the data stored in the **c_servertype** column in the WORK_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 WORK_DB_LOOKUP table. A simple name that used to easily identify the database is stored in this column.