Informatics for Integrating Biology and the Bedside

i2b2 Cell Messaging

Workflow Framework (WORK) 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 **Workflow 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.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.

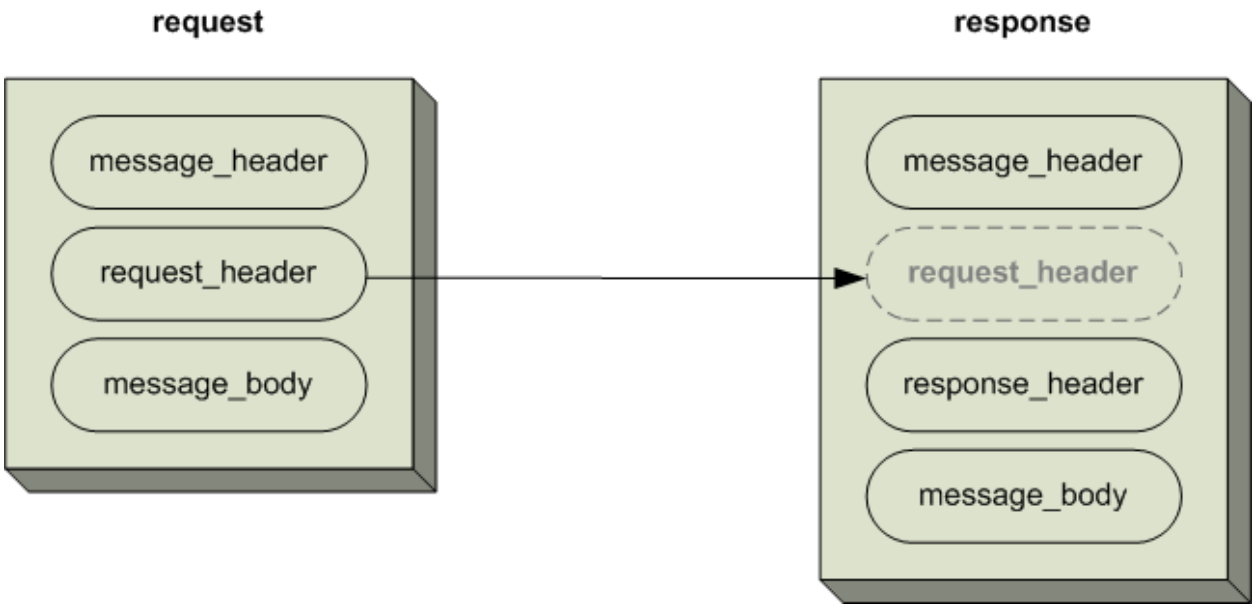


Figure 1: The basic structure of a request and response message. The request_header in the request can be echoed in the response.

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. WORKFLOW FRAMEWORK (WORK) CELL MESSAGING DETAIL

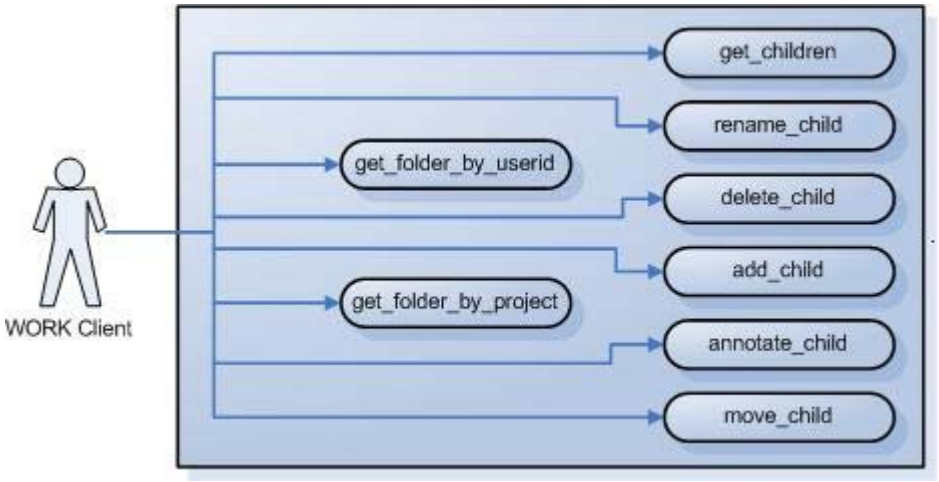
The **Workflow Framework (WORK) Cell** is an optional i2b2 Hive cell. This cell manages project specific XML data objects for users of a given project. The project specific XML data objects originate in other views or cells, such as Ontology or Previous Query and are stored in the WORK cell as a convenience.

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 Cases

The diagram below depicts common use cases a user may perform with the WORK 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 as 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 (folder) in the workplace tree and reassigns all its children.

2.1.2 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 are provided by the Project Management (PM) cell.

2.1.2.1 GET A LIST OF FOLDERS ASSOCIATED WITH A USER

The **get_folder_by_userid** message is sent by the workplace tool to populate the root nodes available to a user.

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

1. Client requests list of folders 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.

2.1.2.2 GET_FOLDER_BY_USERID REQUEST MESSAGE

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

2.1.2.2.1 Possible "type" settings

core = Return all data except system/date information

2.1.2.3 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>
```

2.1.3 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. Project information is provided in the *message_header*; roles are provided by the **Project Management (PM) cell**.

2.1.3.1 GET A LIST OF FOLDERS ASSOCIATED WITH A PROJECT

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

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

1. Client requests list of folders for a given project (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.

2.1.3.2 GET_FOLDER_BY_PROJECT REQUEST MESSAGE

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

2.1.3.2.1 Possible "type" settings

core = Return all data except system/date information

2.1.3.3 GET_FOLDER_BY_PROJECT 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>
```

2.1.4 get_children

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

2.1.4.1 POPULATING CHILDREN OF TREE NODES

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

The sequence of events is as follows:

1. Client sends message with “type” = core “blob” = true
2. WORK server performs following steps:
    a. Parses <parent> to obtain the tabld_cd and index. Queries workplace_access table for table name associated with table_cd.
    b. Query workplace table returned above for all entries with parent_index = index.
3. Client receives list of children and populates tree.

2.1.4.2 GET_CHILDREN REQUEST MESSAGE

A **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 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>
```

2.1.4.2.1 Possible “blob” settings

false = Do not return data stored as a blob or clob (xml, comments)

true = Return xml and comments. Must be set to true for this feature.

2.1.4.3 GET_CHILDREN RESPONSE MESSAGE

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>
```

2.1.5 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.

2.1.5.1 RENAME A NODE IN THE TREE

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

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

2.1.5.2 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>
```

2.1.5.3 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.

2.1.6 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.

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

2.1.6.2 DELETE_CHILD REQUEST MESSAGE

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

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

2.1.6.3 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.

2.1.7 add_child

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

2.1.7.1 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 folder.
2. Workplace server performs following steps:
    a. Parses node index to obtain key/table_cd
    b. Queries workplace_access table for table name associated with table_cd.
    c. Generates a new index for the new leaf or folder.
    d. Inserts new leaf or folder into the workplace table
3. Client populates selected folder with new node.

2.1.7.2 ADD_CHILD REQUEST MESSAGE

This message requires the user to specify the node 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>
```

2.1.7.3 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.

2.1.8 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 folder.

2.1.8.1 ANNOTATE A NODE IN THE TREE

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

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

2.1.8.2 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>
```

2.1.8.3 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.

2.1.9 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.

2.1.9.1 MOVE A FOLDER IN THE TREE

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

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

2.1.9.2 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>
```

2.1.9.3 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.

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 WORK cell:

3.1 WORK.xsd

Describes 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 workplace 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.

<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.

4.1.2 WORK_QRY.xsd

<request>	Container for work data request.

<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 concept. Example: String XYZ, where X can be “C” (container), “F” (folder), “L” (leaf), “M” (muli) Y can be “A” (active), “I” (inactive), “H” (hidden) Z can be “O” (open) or null
<tooltip>	The 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.
<workXml>	The drag and drop XML associated with the workplace item (is null for folders or containers).
<workXmlSchema>	The schema associated with <workXml>.
<workmlI2B2Type>	The i2b2 type associated with the drag and drop XML.