Informatics for Integrating Biology and the Bedside

i2b2 Cell Messaging

Data Repository (CRC) Cell

Document Version: *1.7.08-006*

i2b2 Software Version: *1.7.08*

TABLE OF CONTENTS

DOCUMENT MANAGEMENT

Revision Number	Date	Author	Description of change
1.7.0	09/24/12	Janice Donahoe	Created 1.7 version of the document
1.7.1	11/27/12	Mike Mendis	Added selection filter information
1.7.00-002	08/03/2015	Janice Donahoe	Fixed minor spelling and grammatical errors.
1.7.00-003	09/18/2015	Janice Donahoe	Updated Case 2 in section 2.5 Data Upload Messages. Removed statement that observation facts have to be unique and referenced section 2.5.1 where the rules for appending facts are located.
1.7.00-004	01/05/2016	Janice Donahoe	Fixed an error with the xml message examples. Changed <modifier_path> to the correct tag which is <modifier_key>
1.7.08-005	07/26/2016	S. Wayne Chan	Added the 4 DBlookup Messages.
1.7.08-006	10/06/2016	Janice Donahoe	Updated the information on the 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 **Data Repository (CRC) 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.nih.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 application 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 of 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.

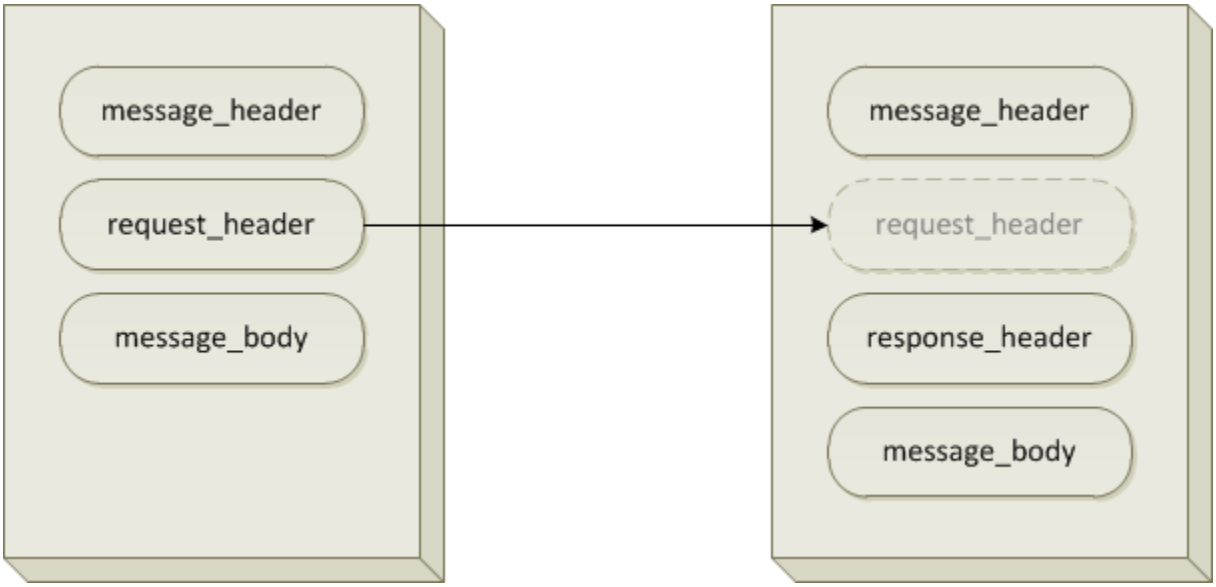
1.2 i2b2 Messaging Overview

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.

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. The <*request_header*> is used for requests but may be echoed back (optionally) in the response. The response must include a <*response_header*>.

The XSD specification of the i2b2 message permits individual cells to add cell-specific XML in 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.

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



1.2.1 XSD Files for i2b2 XML Schema

The i2b2 XML schema consists of three XSD files:

1.2.1.1 i2b2.xsd

This schema is not used directly to create i2b2 messages, but is included in the *i2b2_request.xsd* and the *i2b2_response.xsd*.

It defines the <i2b2:request> tag, which includes the <message_header> tag.

1.2.1.2 i2b2_request.xsd

This schema is used for validating i2b2 request messages.

It defines the <i2b2:request> tag, which includes the <message_header> tag.

1.2.1.3 i2b2_response.xsd

This schema is used for validating i2b2 response messages.

It defines the <i2b2:response> tag, which includes the <message_header> tag.

2 DATA REPOSITORY (CRC) CELL MESSAGING DETAIL

The Data Repository Cell is one of the core cells in the i2b2 hive. Since much of the data in the repository is clinical in nature, it has also come to be known as the Clinical Research Chart (CRC) and the terms "**data repository**" and "**CRC**" are used interchangeably. The data repository is a warehouse of patient phenotypic and genotypic data that interacts with other cells to provide information for users.

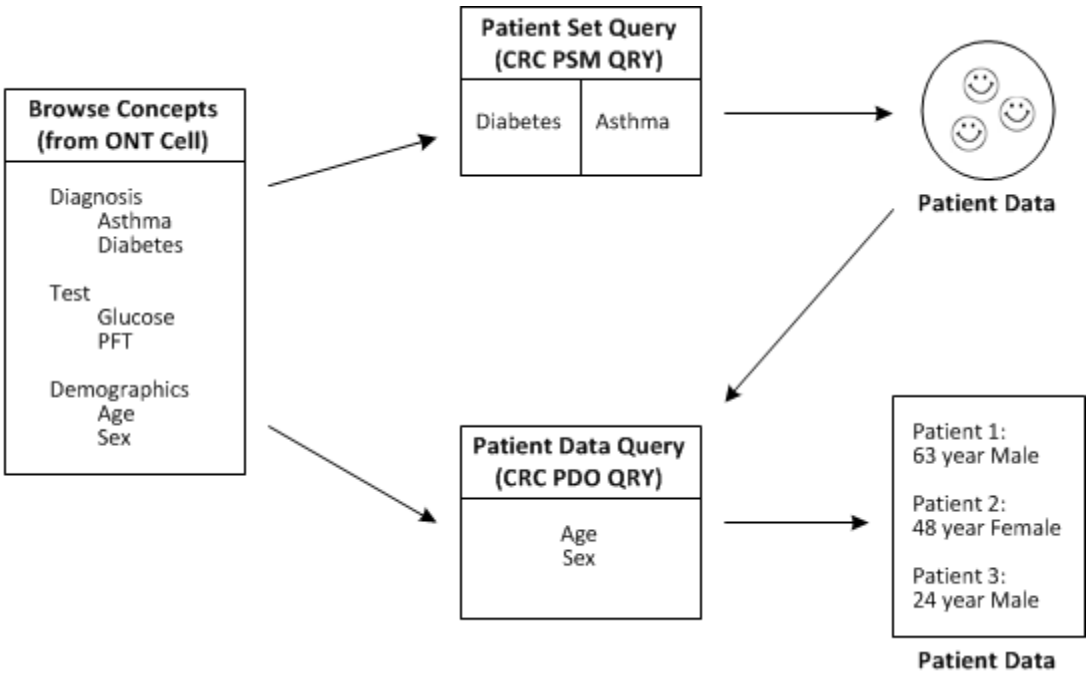
Communication with the CRC Cell, like all cells in the i2b2 hive, is handled via standardized XML web services. These XML messages conform to the i2b2 messaging standard described above, which allows cell-specific XML within the <*message_body*> tag. The remainder of this document describes **CRC-specific web services and the XML formats** that encode them and illustrates how these XML messages are used to accomplish a set of interactions that correspond to typical CRC use cases.

A typical CRC client may want to define a patient set and then request patient data on that set. Both of these tasks require the user to first interact with another cell called the Ontology Management (ONT) Cell in order to choose concept codes to define in the CRC request. Although the specific interactions with the ONT Cell are not described in this document, the following diagram shows the basic flow of information.

Scenario:

The diagram describes how a user may get the age and gender for all patients who have either diabetes or asthma.

- The user starts by selecting the diagnoses "Diabetes" and "Asthma" from the ONT cell; these define the Patient Set Query, which creates a patient set in the data repository.
- Then the user selects the demographic concepts "Age" and "Sex" from the ONT cell to define the Patient Data Query.
- The patient data query returns the age and gender for all patients in the data set, those with diabetes or asthma.



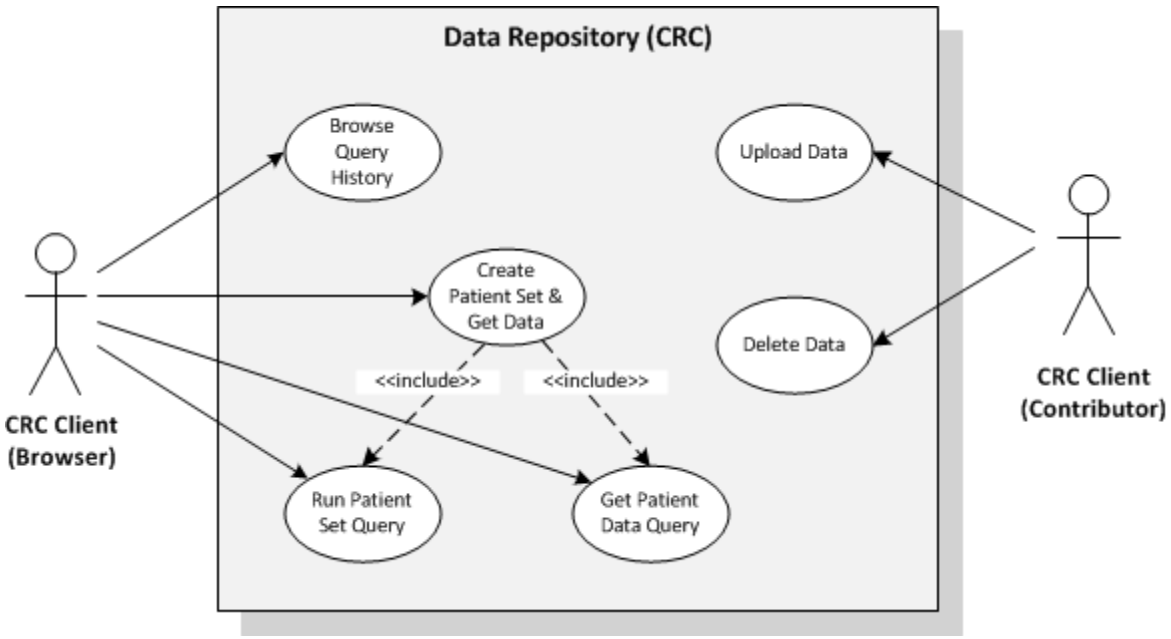
2.1 Use Cases

The CRC Cell is a repository of clinical data and has a set of services that respond to requests for patient data. A request might be issued by a client cell which is used by a researcher conducting a clinical trial in order to help gather a cohort.

There are two types of clients or users; the **contributing client** and the **browsing client**.

The *contributing client* adds content to the CRC by uploading patient data or deleting data by removing previous uploads.

The *browsing client* has four possible interactions with the repository cell. The user may create queries that define patient sets, browse previous queries, rerun existing patient set queries and get specific patient data from a patient set.



2.2 Services / Messages

The CRC Cell provides services that support the interactions necessary for each of the use cases described in the *Use Cases* section of this document. The services expect different message request types for each specific behavior or request.

2.2.1 Browse Query History

Get a list of saved query definitions	Provides a list of all prior queries created by a client / user
Get a list of saved query results	Provides query results for given query run / instance
Get and XML definition of a defined query	Returns the definition of the query

2.2.2 Run Patient Set Query

Run patient set query	Runs the query and returns the result and its status

2.2.3 Get Patient Data Query

Get patient data from a patient set	Returns the patient data object for the given patient set

2.2.4 Database Connection Query

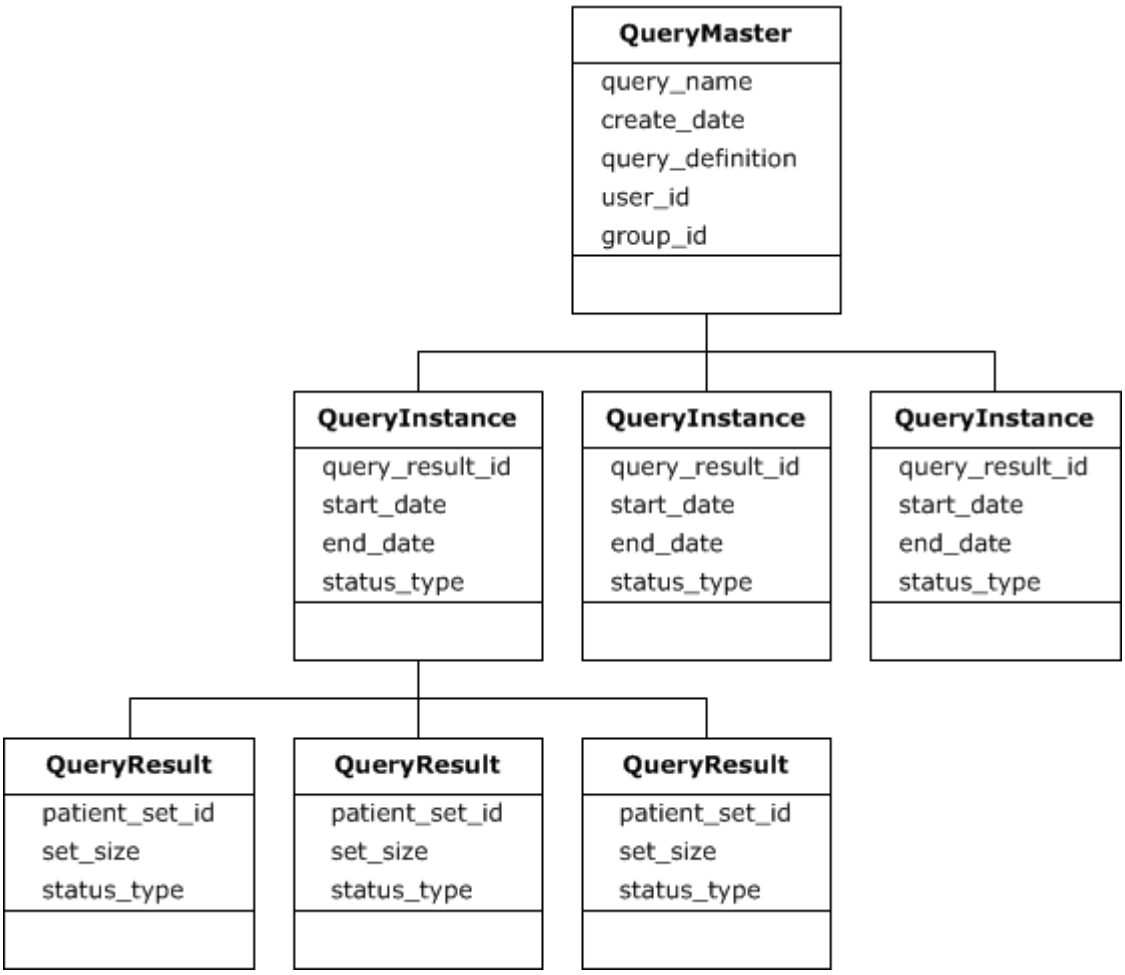
get_all_dblookups	Returns a list of all the entries (rows) in the CRC_DB_LOOKUP table.
set_dblookup	Adds or updates a specific entry (row) in the CRC_DB_LOOKUP table.
get_dblookup	Returns all the data for a specific entry (row) in the CRC_DB_LOOKUP table.
delete_dblookup	Deletes a specific entry (row) in the CRC_DB_LOOKUP database table.

Note

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

2.3 Patient Set Query Service

2.3.1 Conceptual Model



The **QueryMaster** holds the master information about the query like the *query name*, *query_definition*, *user_id*, and *group_id*.

The **query_defintion** element in the QueryMaster holds the xml representation of the query constraint; its details are described below.

The run information of the query is recorded in the **QueryInstance** and is created whenever the query is executed.

QueryResult typically holds the result information of the query, such as the *id* and the *set size* of the patient set, as well as the result status.

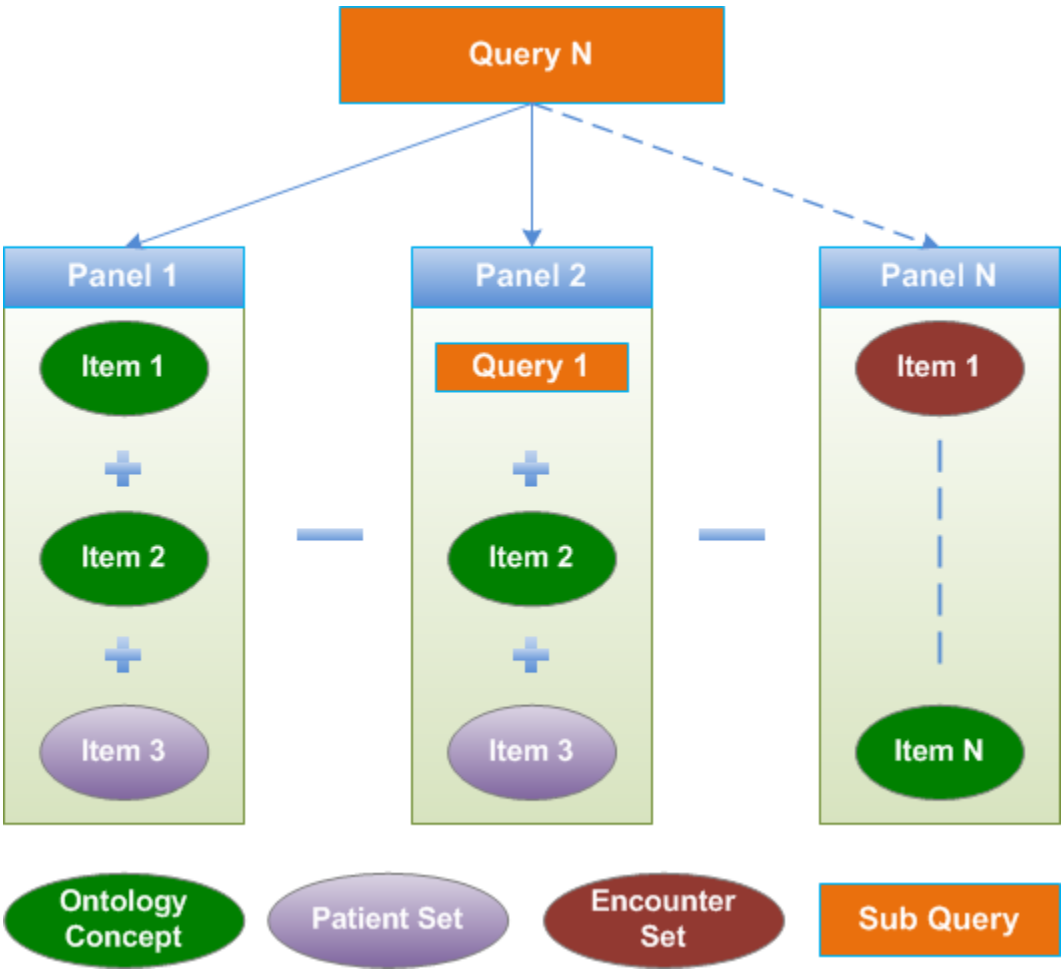
There can be multiple QueryInstances for one QueryMaster and one QueryInstance can have multiple QueryResults.

2.3.2 Panel Definition Details

The **panel** is the common definition used to query the data mart. The panel holds one or more <*items*>. These items can be one of the following:

1. concept
2. patient set id
3. patients encounter set id
4. another query

The <*item*> constraints are OR-ed and the <*panel*> constraints are AND-ed in the query. Both the setfinder and PDO request share this panel definition.



Example:

Italicized and *Gray* are optional for both setfinder and PDO schema.

Gray items (no italics) are required only if the parent container is being used.

Italicized items are optional but **only** for setfinder schema.

All others are required.

```
<panel name="name0">
    <panel_number># of panel</panel_number>
    <panel_timing>ANY|SAMEVISIT|SAMEINSTANCENUM</panel_timing>
    <panel_date_from time="start_date" inclusive="yes"></panel_date_from>
    <panel_date_to time="start_date" inclusive="yes"></panel_date_to>
    <panel_accuracy_scale>1</panel_accuracy_scale>
    <invert>0</invert>
```

```
<total_item_occurrences operator="EQ/NE/GT/GE/LT/LE">1</total_item_occurrences>
<item>
    <hlevel>3</hlevel>
    <item_name>item_name0</item_name>
    <item_table>
    <item_key>
        item_key0| patient_set_coll_id:NNN |
        patient_set_enc_id:NNN |masterid:NNN
    </item_key>
    <item_icon>item_icon0</item_icon>
    <tooltip>tooltip0</tooltip>
    <class>ENC</class>
    <constrain_by_value>
        <value_operator>
            EQ/NE/GT/GE/LT/LE/IN/BETWEEN/LIKE[exact]/LIKE[begin]/
LIKE[end]/LIKE[contains]/Contains[database]
        </value_operator>
        <value_constraint>value_constraint0</value_constraint>
        <value_unit_of_measure>unit</value_unit_of_measure>
        <value_type>TEXT/LARGETEX/NUMBER/FLAG/MODIFIER</value_type>
    </constrain_by_value>
    <constrain_by_date>
        <date_from time="start_date" inclusive="yes">2006-05-04</date_from>
        <date_to time="start_date" inclusive="yes">2006-05-04</date_to>
    </constrain_by_date>
    <constrain_by_modifier>
        <applied_path>\i2b2\Medications\%</applied_path>
        <modifier_key>\\i2b2_DEMO\Dose\</modifier_key>
        <constrain_by_value>
            <value_operator>
                EQ/NE/GT/GE/LT/LE/IN/BETWEEN/LIKE[exact]/LIKE[begin]/
LIKE[end]/LIKE[contains]/Contains[database]
            </value_operator>
            <value_constraint> value_constraint0</value_constraint>
            <value_unit_of_measure>unit_of_measure</value_unit_of_measure>
            <value_type>TEXT/NUMBER/FLAG/MODIFIER</value_type>
        </constrain_by_value>
    </constrain_by_modifier>
    <dim_tablename>dim_tablename0</dim_tablename>
    <dim_columnname>dim_columnname0</dim_columnname>
    <dim_dimcode>dim_dimcode0</dim_dimcode>
    <dim_columndatatype>dim_columndatatype0</dim_columndatatype>
    <dim_operator>dim_operator0</dim_operator> LIKE
    <facttablecolumn>facttablecolumn0</facttablecolumn>
    <item_color/>
    <item_shape/>
    <item_row_number/>
```

```
        <item_is_synonym/>
      </item>
</panel>
```

Element Name	Description
panel	A concept used to group items. The set of observation facts for each item filter are "unioned" at the panel level. The panel has the attribute "name", which is the key field for the panel and it is unique.
panel_timing	The values of the panel timing are the same as the values of the <query_timing> option in the query definition. The panel timing will override the query timing value if both have values specified. **Value** / **Description** ANY — ANY is the default value. The query result doesn't depend on the patient's encounter / visit value SAME, SAMEVISIT — The patients selected within the panel will have the same Encounter / Visit. SAMEINSTANCENUM — The patients selected within the panel will have the same Encounter / Visit and INSTANCE_NUM value in the fact table.
panel_number	A serial number starting with 1.
panel_date_from	Apply the observation fact's start date condition at the panel level.
panel_date_to	Apply the observation fact's end date condition at the panel level.
invert	The invert value could by "1" or "0". If the value is "1" then the query applies "NOT" condition for the entire panel.
total_item_occurrences	Select the events only if the total number of occurrence is greater or equal to this value.
item	Item contains the filter and query building information, like the item key, dimension table column name, data type, etc.
hlevel	Hierarchy level not required for this implementation
item_name	Nam of the item. It is not a required element and mostly for UI purposes.
item_table	The name of the dimension table.
item_key	This is the key field for the query and it is of four types, each one distinguished by

<table>
<tr><td></td><td>the key format.

<table><tr><th>Key Type</th><th>Example</th></tr><tr><td>Dimension Path Key</td><td>Item key representing the unique path of concepts available in metadata schema or the Ontology Cell. The format of the item_key is [\\Dimension\concept path].
<item_key>\\rpdr\RPDR\Diagnoses</item_key></td></tr><tr><td>Patient Set</td><td>Providing the patient set id.
<item_key>patient_set_coll_id:NNN</item_key></td></tr><tr><td>Encounter Set</td><td>Providing the patient’s encounter set id.
<item_key>patient_set_enc_id:NNN</item_key></td></tr><tr><td>Query Master Id</td><td>Providing another query’s master id, aka query-in-query.
<item_key>masterid:NNN</item_key></td></tr></table></td></tr>
<tr><td>item_icon</td><td>Not currently used.</td></tr>
<tr><td>tooltip</td><td>This is not a required element. It is used by the i2b2 clients.</td></tr>
<tr><td>class</td><td>This is not currently used. It has been added to the specification and could be used to classify the data. For example, whether we need a fact’s image data, text data, etc.</td></tr>
<tr><td>constrain_by_value</td><td>To constrain the observation value of a concept. More information can be found in the CRC Design Document.</td></tr>
<tr><td>value_operator</td><td>The conditional operator for filtering. The values are:
EQ, NE, GT, GE, LT, LE, IN, BETWEEN, LIKE[exact], LIKE[begin], LIKE, LIKE[end], and LIKE[contains].</td></tr>
<tr><td>value_constraint</td><td>Example ‘GT’ Operator:
<constrain_by_value>
<value_operator>GT</value_operator>
<value_constraint>99.9</value_constraint>
<value_type>NUMBER</value_type>
</constrain_by_value>

Example ‘IN’ Operator:
<value_constraint>(‘NEG’,’NEGATIVE’)</value_constraint>

Example ‘BETWEEN’ Operator:
<value_constraint>100 and 200</value_constraint></td></tr>
<tr><td>value_unit_of_measure</td><td></td></tr>
</table>

<table>
<tr><td>value_type</td><td>TEXT, LARGETEXT, NUMBER, FLAG

The following shows how the SQL will be built for the different operators:

<table>
<tr><th>VALUE_ TYPE</th><th>VALUETYPE_ CD</th><th>TVAL_C HAR</th><th>VALUEFLAG _CD</th><th>OBSERVATION_ BLOB</th><th>NVAL_ NUM</th></tr>
<tr><td>TEXT</td><td>T</td><td>*</td><td></td><td></td><td></td></tr>
<tr><td>LARGET EXT</td><td>B</td><td></td><td></td><td>*</td><td></td></tr>
<tr><td>NUMBER</td><td>N</td><td>*</td><td></td><td></td><td>*</td></tr>
<tr><td>FLAG</td><td></td><td></td><td>*</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>constrain_by_date</td><td>Apply start and end date constraint for the item</td></tr>
<tr><td>date_from</td><td><date_from time="start_date/end_date" inclusive="yes"></date_from></td></tr>
<tr><td>date_to</td><td><date_to time="start_date/end_date" inclusive="yes"></date_to></td></tr>
<tr><td>constrain_by_modifier</td><td>Apply modifier constraint for the item. The MODIFIER_CD column in the fact table is used to apply this constraint to the fact.</td></tr>
<tr><td>applied_path</td><td>The applied path for the modifier.
<applied_path>\i2b2\Medications\%</applied_path></td></tr>
<tr><td>modifier_key</td><td>The modifier’s path.
<modifier_key>\\i2b2_DEMO\Dose\</modifier_key></td></tr>
<tr><td>constrain_by_value</td><td>Same as the <constrain_by_value> section at the item level.</td></tr>
<tr><td>dim_tablename</td><td>The name of the dimension table to join with the fact table. i.e. ‘CONCEPT_DIMENSION’, ‘PROVIDER_DIMENSION’, etc.

This information is used to construct the dimension filter SQL.

Example:

SELECT *
FROM OBSERVATION_FACT
WHERE facttablecolumn IN
(SELECT dim_columnname
FROM dim_tablename
WHERE dim_columnname LIKE dim_dimcode
)</td></tr>
<tr><td>dim_columnname</td><td>The name of the column in the dimension table.</td></tr>
<tr><td>dim_dimcode</td><td>The is the same as the concept path i.e. ‘\i2b2\Diagnoses’</td></tr>
</table>

dim_columndatatype	The data type of the dimension table's filter column. The default is string.
dim_operator	The conditional operator for filtering. The default is the 'LIKE' operator. Other values are 'LE', 'GE', and 'EQ'
facttablecolumn	This is the name of the column in the OBSERVATION_FACT table to join the dimension table.
item_color	UI rendering attribute.
item_shape	UI rendering attribute.
item_row_number	UI rendering attribute.
item_is_synonym	UI rendering attribute.

2.3.3 Request and Response Message Structure

The patient set request / response message structure is divided into three parts:

1. PSMHeader
2. Request
3. Response

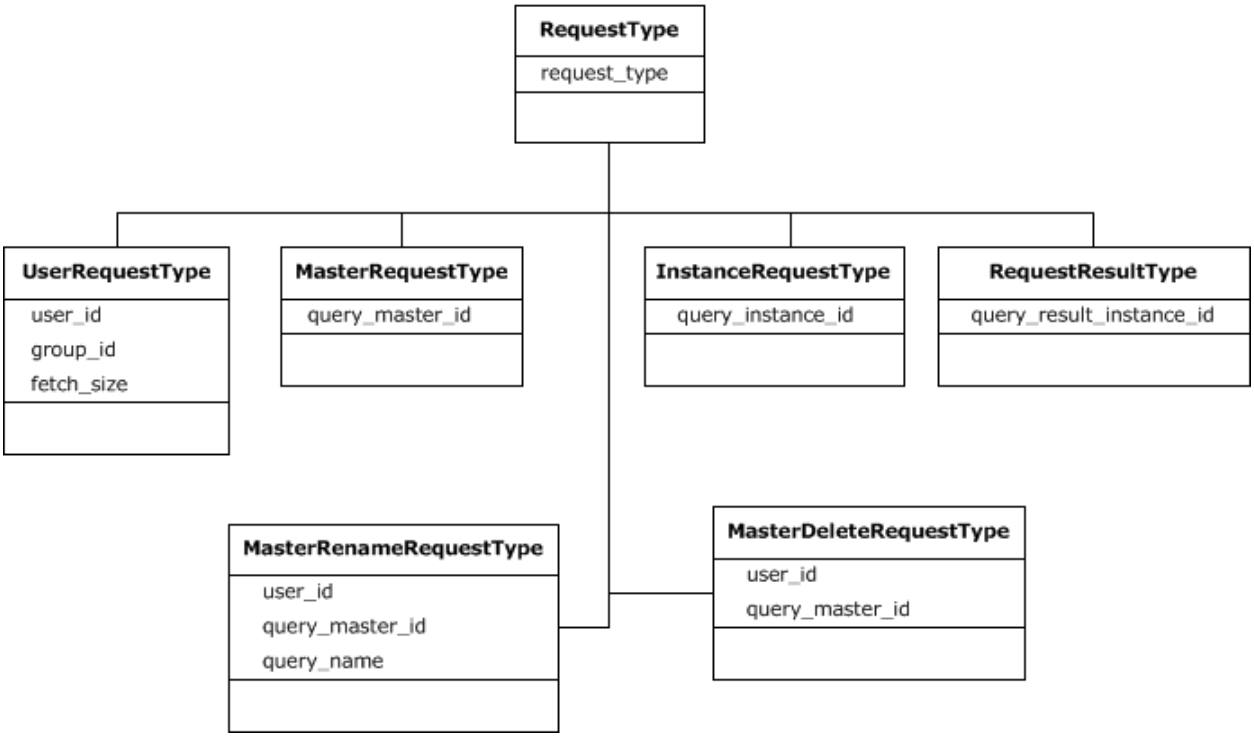
For a request message, the <*psmheader*> and <*request*> sections are required; while a response message requires only the <*response*> section.

```
<i2b2:message_body>
    <crc:psmheader>
        <request_type></request_type>
    </crc:psmheader>
    <crc:request>
        …
    </crc:request>
    <crc:response>
        …
    </crc:response>
</i2b2:message_body>
```

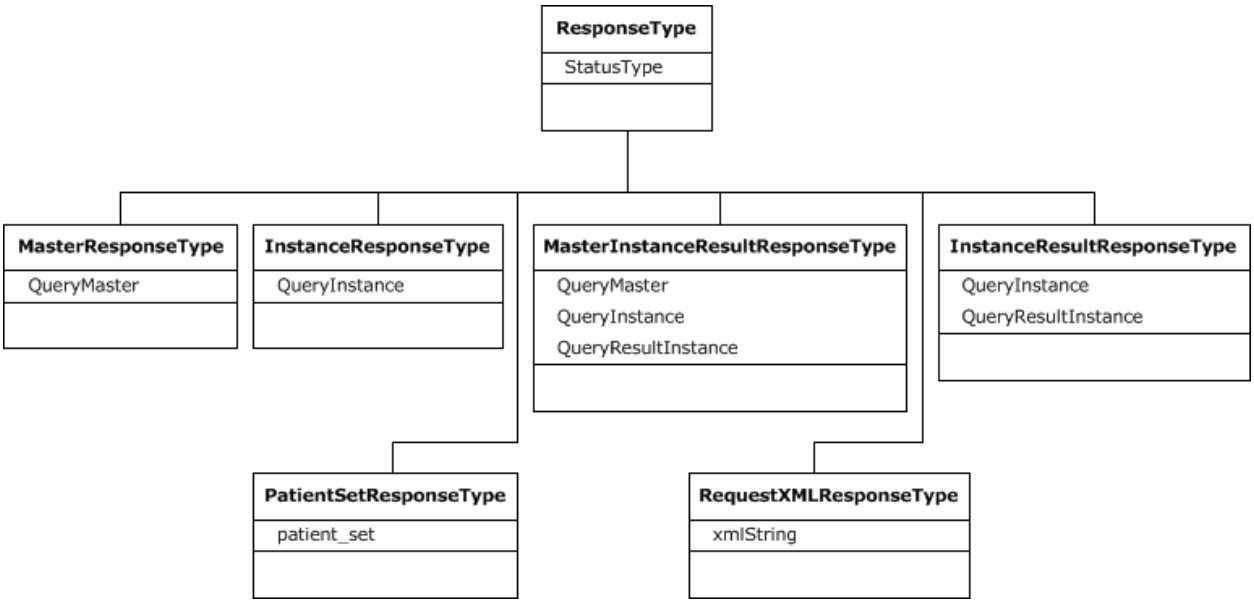
Element Name	Description
psmheader	The header has a <request_type> element, which will carry the operation name. Each operation name has a specific request / response combination. The following is a list of supported operation names: • CRC_QRY_getRequestXml_fromQueryMasterId • CRC_QRY_getQueryMasterList_fromUserId • CRC_QRY_runQueryInstance_fromQueryDefinition • CRC_QRY_getQueryMasterList_fromGroupId • CRC_QRY_getQueryResultInstanceList_fromQueryInstanceId • CRC_QRY_getQueryInstanceList_fromQueryMasterId • CRC_QRY_deleteQueryMaster • CRC_QRY_renameQueryMaster • CRC_QRY_runQueryInstance_fromQueryMasterId • CRC_QRY_getResultDocument_fromResultInstanceId • CRC_QRY_getResultType • CRC_QRY_runQueryInstance_fromAnalysisDefinition • CRC_QRY_cancelQuery • CRC_QRY_updateResultInstanceDescription
request	The <request> is modeled as an object using a polymorphic approach. All operation specific request objects inherit a base **RequestType *object*** containing a **request_type *attribute*** as shown in the object model diagram in the next section.
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*** as shown in the object model diagram in the next section.

2.3.4 Request and Response Object Model

Request Object Model:



Response Object Model:



The following chart shows the different request and response types for each service type listed above. The **RequestType** column describes what input is expected and the **ResponseType** column describes what output is expected.

Operation	RequestType						ResponseType					
	User	Master (Query)	Instance (Query Run)	QueryDefinition	PatientSet	ObservationFact	Master (Query)	Instance (Query Run)	Result	RequestXml	MasterInstanceResult	PatientData
Get a List of Saved Query Definitions	x						x					
Get a List of Saved Query Runs		x						x				
Get a List of Saved Query Results			x						x			
Get XML Definition of a Defined Query		x								x		
Run (New) Patient Set Query				x							x	
Run (Existing) Patient Set Query		x									x	
Get Patient Data from a Patient Set					x							x
Get Patient Data from Observation Fact						x						x

2.3.5 Use Case Scenarios

2.3.5.1 Execute a Query and Get Its Results

The service queries the data mart using the query definition and generates output based on the result option. Each query request and its results will be recorded under the given user id and project id.

The server will read the value <*result_waittime_ms*> from the <*request_header*> and if the query did not complete before the wait time specified in the request, it will send a response to the client with a "**PENDING**" status. The client can later send a query instance request to see if the query has completed and retrieve the query result information.

The query definition option:

<table>
<tr><th>Element Name</th><th>Description</th></tr>
<tr><td>query_name</td><td>Name of the query</td></tr>
<tr><td>query_description</td><td>Description of the query</td></tr>
<tr><td>query_timing</td><td>The values of the query timing are the same as the values of the <panel_timing> option in the panel definition. The panel timing will override the query timing value if both have values specified.

<table><tr><th>Value</th><th>Description</th></tr><tr><td>ANY</td><td>The query result doesn’t depend on the patient’s encounter / visit value</td></tr><tr><td>SAME
SAMEVISIT</td><td>The patients selected within the panel will have the same Encounter / Visit.</td></tr><tr><td>SAMEINSTANCENUM</td><td>The patients selected across the panels will have the same Encounter / Visit, CONCEPT_CD, START_DATE, PROVIDER_ID, and INSTANCE_NUM value in the fact table.</td></tr></table></td></tr>
<tr><td>specificity_scale</td><td>Not implemented</td></tr>
<tr><td>panel</td><td>The panel option is defined in the previous section called Panel Definition Details.</td></tr>
</table>

The service supports the following result options.

- PATIENTSET
- PATIENT_ENCOUNTER_SET
- PATIENT_COUNT_XML
- PATIENT_GENDER_COUNT_XML
- PATIENT_AGE_COUNT_XML
- PATIENT_VITALSTATUS_COUNT_XML
- PATIENT_RACE_COUNT_XML

Note

If the result option (<result_output_list>) is not specified in the request, then the default result option is “PATIENTSET”.

CRC_ENABLE_UNITCD_CONVERSION=ON|OFF

(https://community.i2b2.org/wiki/display/DevForum/Metadata+XML+for+Medication+Modifiers)

To use LARGETEXT constraint in the query, the minimum user role of DATA_DEID is required.

Notes:

1. The CRC will take advantage of simple query by not using the temp table. As a default it is advisable to set the flag <query_mode> optimize_without_temp_table</query_mode> in the <psmheader>.
2. To speed up the query, the CRC selects which panel SQL to run first. The selection of panel will be based on the individual item / concept’s total_num value specified on the Ontology’s Metadata table. It is not necessary to populate total_num for each of the Metadata’s concepts, but just populating some of the commonly used concepts like the ‘\Demographics\Gender\Male’ would help. Please refer to the Ontology cell if you would like to automatically populate the total_num column for all the Ontology’s metadata concept.
3. If the concepts numerical facts value is not stored in normalized unit_cd, then the user can enable the unit conversion in the query before applying the value constraint by setting the PM project param (CRC_ENABLE_UNITCD_CONVERSION=ON|OFF). The fact’s nval_num column will be converted based on the metadata xml (<ConvertingUnits/>, <MultiplyingFactor/> exists in the Ontology cell before applying the query’s value constraint. For better query performance do not enable this option, make sure numerical fact values are in the normalized units.
4. To enable the process timing information in the <query_instance> section of the response set the PM project param (PM_ENABLE_PROCESS_TIMING=INFO|DEBUG).

Result Output Name	Description	Output Value

PATIENTSET	The patient set from the query will be persisted in the database	Not applicable
PATIENT_ENCOUNTER_SET	The set of patients and the corresponding encounter will be persisted in the database.	Not applicable
PATIENT_COUNT_XML	Returns the patient count in the i2b2 result xml format. The i2b2 result format is similar to the name-value pair. (i2b2_result_msg.xsd)	<i2b2_result_envelope xmlns:ns10="http://www.i2b2.org/xsd/hive/msg/result/1.1/"> <body> <result name="PATIENT_COUNT_XML"> <data type="int" column="patient_count">1000</data> </result> </body> </i2b2_result_envelope>
PATIENT_GENDER_COUNT_XML	Returns the patient gender count in the i2b2 result xml format. Refer to i2b2_result_msg.xsd for the i2b2 result format.	<i2b2_result_envelope xmlns:ns10="http://www.i2b2.org/xsd/hive/msg/result/1.1/"> <body> <result name="PATIENT_GENDER_COUNT_XML"> <data type="int" column="Female">100</data> <data type="int" column="Male">200</data> <data type="int" column="Unknown">9</data> </result> </body> </i2b2_result_envelope>
PATIENT_AGE_COUNT_XML	Returns the patient age count in the i2b2 result xml format. Refer to i2b2_result_msg.xsd for the i2b2 result format.	<i2b2_result_envelope xmlns:ns10="http://www.i2b2.org/xsd/hive/msg/result/1.1/"> <body> <result name="PATIENT_AGE_COUNT_XML"> <data type="int" column=">= 65 years old">83</data> <data type="int" column=">= 85 years old">6</data> <data type="int" column="0-9 years old">1</data> <data type="int" column="10-17 years old">0</data> <data type="int" column="18-34 years old">16</data> <data type="int" column="35-44 years old">31</data> <data type="int" column="45-54 years old">32</data> <data type="int" column="55-64 years old">44</data> <data type="int" column="65-74 years old">42</data> <data type="int" column="75-84 years old">35</data>

		<data type="int" column="zz not recorded">0</data> </result> </body> </i2b2_result_envelope>
PATIENT_VITALSTATUS_COUNT_XML	Returns the patient vital status count in the i2b2 result xml format. Refer to i2b2_result_msg.xsd for the i2b2 result format.	<i2b2_result_envelope xmlns:ns10="http://www.i2b2.org/xsd/hive/msg/result/1.1/"> <body> <result name="PATIENT_VITALSTATUS_COUNT_XML"> <data type="int" Column="Deceased">0</data> <data type="int" column="Deferred">0</data> <data type="int" column="Living">123</data> <data type="int" column="Not recorded">0</data> </result> </body> </i2b2_result_envelope>
PATIENT_RACE_COUNT_XML	Returns the patient race count in the i2b2 result xml format. Refer to i2b2_result_msg.xsd for the i2b2 result format.	<i2b2_result_envelope xmlns:ns10="http://www.i2b2.org/xsd/hive/msg/result/1.1/"> <body> <result name="PATIENT_RACE_COUNT_XML"> <data type="int" column="American Indian">30</data> <data type="int" column="Asian">40</data> <data type="int" column="Black">18</data> <data type="int" column="Hispanic">19</data> <data type="int" column="Indian">2</data> <data type="int" column="Not recorded">1866</data> <data type="int" column="White">100</data> </result> </body> </i2b2_result_envelope>

2.3.5.2 Message Request and Response

Request Type	Request	Response

CRC_QRY_runQueryInstance_from QueryDefinition	query_definition_requestType	master_instance_result_responseType

Example:

```
<request_header>
    <result_waittime_ms>90000</result_waittime_ms>
</request_header>

<message_body>
    <crc:psmheader>
        <query_mode>optimize_without_temp_table</query_mode>
        <request_type>
            CRC_QRY_runQueryInstance_fromQueryDefinition
        </request_type>
    </crc:psmheader>

    <crc:request xsi:type="crc:query_definition_requestType">
        <query_definition>
            <query_name/>
            <query_description/>
            <query_timing>ANY</query_timing>
            <panel>
                <panel_number>1</panel_number>
                <panel_date_from>2000-12-30T00:00:00</panel_date_from>
                <panel_date_to>2000-12-30T00:00:00</panel_date_to>
                <invert>0</invert>
                <total_item_occurrences>1</total_item_occurrences>
                <item>
                    <item_key>\\rpdr\RPDR\Diagnoses</item_key>
                    <!-- OR  pass patient set id
<item_key>patient_set_coll_id:123</item_key>
                    -->
                    <!-- OR  pass encounter set id
<item_key>patient_set_enc_id:123</item_key>
                    -->
                    <constrain_by_value>
                        <value_operator> </value_operator>
                        <value_constraint>value_constraint0</value_constraint>
                        <value_unit_of_measure>unit</value_unit_of_measure>
                        <value_type></value_type>
                    </constrain_by_value>
                    <constrain_by_modifier>
                        <applied_path>\i2b2\Medications\%</applied_path>
```

```
                    <modifier_key>\\i2b2_DEMO\Dose\</modifier_key>
                    <constrain_by_value>
                        <value_operator></value_operator>
                        <value_constraint> </value_constraint>
                        <value_unit_of_measure></value_unit_of_measure>
                        <value_type></value_type>
                    </constrain_by_value>
                </constrain_by_modifier>
                <constrain_by_date>
                </constrain_by_date>
            </item>
        </panel>
    </query_definition>
    <result_output_list>
        <result_output name="PATIENT_COUNT_XML"/>
        <result_output name="PATIENT_GENDER_COUNT_XML "/>
        <result_output name="PATIENT_AGE_COUNT_XML "/>
        <result_output name="PATIENT_VITALSTATUS_COUNT_XML "/>
        <result_output name="PATIENT_RACE_COUNT_XML "/>
        <result_output name="PATIENTSET"/>
    </result_output_list>
</crc:request>

<crc:response xsi:type="crc:master_instance_result_responseType">
    <query_master>
        <query_master_id>0</query_master_id>
        <name/>
        <user_id/>
        <group_id/>
        <create_date>2000-12-30T00:00:00</create_date>
        <request_xml/>
    </query_master>
    <query_instance>
        <query_instance_id>0</query_instance_id>
        <query_master_id>0</query_master_id>
        <user_id/>
        <group_id/>
        <batch_mode/>
        <message>
            <!--  optional process status message -->
            <process_step_timing>
                <name/>
                <start_date/>
                <end_date/>
                <total_time_second/>
                <message/>
```

```
            </process_step_timing>
            . . .
        </message>
        <start_date>2000-12-30T00:00:00</start_date>
        <end_date>2000-12-30T00:00:00</end_date>
        <query_status_type>
            <status_type_id>6</status_type_id>
            <name>COMPLETED</name>
            <description/>
        </query_status_type>
    </query_instance>
    <query_result_instance>
        <result_instance_id>0</result_instance_id>
        <query_instance_id>0</query_instance_id>
        <query_result_type>
            <result_type_id>1</result_type_id>
            <name>PATIENTSET</name>
            <description/>
        </query_result_type>
        <set_size>0</set_size>
        <obfuscate_method>OBTOTAL</obfuscate_method>
        <start_date>2000-12-30T00:00:00</start_date>
        <end_date>2000-12-30T00:00:00</end_date>
        <query_status_type>
            <status_type_id>3</status_type_id>
            <name>FINISHED</name>
            <description/>
        </query_status_type>
    </query_result_instance>
    <query_result_instance>
        <result_instance_id>0</result_instance_id>
        <query_instance_id>0</query_instance_id>
        <query_result_type>
            <result_type_id>4</result_type_id>
            <name> PATIENT_COUNT_XML </name>
            <description/>
        </query_result_type>
        <set_size>0</set_size>
        <obfuscate_method>OBSUBTOTAL</obfuscate_method>
        <start_date>2000-12-30T00:00:00</start_date>
        <end_date>2000-12-30T00:00:00</end_date>
        <query_status_type>
            <status_type_id>3</status_type_id>
            <name>FINISHED</name>
            <description/>
        </query_status_type>
```

```
            </query_result_instance>
        </crc:response>
    </message_body>
```

2.3.5.3 Scenario: Check if the query is completed and get its results

This request accepts a **query_instance_id** and returns the query status information. i.e. COMPLETED, RUNNING, ERROR, etc.

Request Type	Request	Response
CRC_QRY_getQueryResultInstanceList_fromQueryInstanceId	instance_requestType	result_responseType

Example:

```
<message_body>
    <psmheader>
        <user login="demo">demo</user>
        <patient_set_limit>0</patient_set_limit>
        <estimated_time>0</estimated_time>
        <request_type>
            CRC_QRY_getQueryResultInstanceList_fromQueryInstanceId
        </request_type>
    </psmheader>

    <request xsi:type="ns4:instance_requestType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <query_instance_id>6280</query_instance_id>
    </request>

    <response xsi:type="ns5:result_responseType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
        <query_result_instance>
            <result_instance_id>6280</result_instance_id>
            <query_instance_id>6280</query_instance_id>
            <query_result_type>
                <result_type_id>1</result_type_id>
                <name>PATIENTSET</name>
```

```
            </query_result_type>
            <set_size>2000</set_size>
            <start_date>2007-09-06T10:42:14.000-04:00</start_date>
            <end_date>2007-09-06T10:42:15.000-04:00</end_date>
            <query_status_type>
                <status_type_id>3</status_type_id>
                <name>FINISHED</name>
            </query_status_type>
        </query_result_instance>
    </response>
</message_body>
```

2.3.5.4 Scenario: Get the query result (XML output) by result instance

Pass the query result instance id and get the query result. If the result instance has xml output, then the xml output will be passed in the <*xml_value*> and the <xml_value> element will contain an i2b2 result xml document.

Request Type	Request	Response
CRC_QRY_getResultDocument_fromResultInstanceId	result_requestType	crc_xml_result_responseType

Example:

```
<message_body>
    <psmheader>
        <request_type>
            CRC_QRY_getResultDocument_fromResultInstanceId
        </request_type>
    </psmheader>

    <ns4:request xsi:type="ns4:result_requestType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <query_result_instance_id>7953</query_result_instance_id>
    </ns4:request>

    <ns4:response xsi:type="ns4:crc_xml_result_responseType">
        <status>
```

```
            <condition type="DONE">DONE</condition>
        </status>
        <query_result_instance>
            <result_instance_id>7953</result_instance_id>
            <query_instance_id>8192</query_instance_id>
            <query_result_type>
                <result_type_id>4</result_type_id>
                <name>PATIENT_GENDER_COUNT_XML</name>
                <description>PATIENT GENDER COUNT XML</description>
            </query_result_type>
            <set_size>5815</set_size>
            <start_date>2008-07-02T11:07:05.000-04:00</start_date>
            <end_date>2008-07-02T11:07:05.000-04:00</end_date>
            <query_status_type>
                <status_type_id>3</status_type_id>
                <name>FINISHED</name>
                <description>FINISHED</description>
            </query_status_type>
        </query_result_instance>
        <crc_xml_result>
            <xml_result_id>804</xml_result_id>
            <result_instance_id>7953</result_instance_id>
            <xml_value><?xml version="1.0" encoding="UTF-8" standalone="yes"?>
                <ns9:i2b2_result_envelope
                xmlns:ns9="http://www.i2b2.org/xsd/hive/msg/result/1.1/">
                    <body>
                        <ns9:result name="patient_gender_count">
                            <data type="int" column="male_count">1559</data>
                            <data type="int" column="female_count">4256</data>
                        </ns9:result>
                    </body>
                </ns9:i2b2_result_envelope>
            </xml_value>
        </crc_xml_result>
    </ns4:response>
</message_body>
```

2.3.5.5 Scenario: Run an analysis plug-in and get the status

Run an Analysis plug-in by passing the plug-in name and its parameters. The analysis name and parameter are usually part of the individual Analysis plug-in document. The response message for this request is similar to the Setfinder's run query request.

Request Type	Request	Response
CRC_QRY_runQueryInstance_fromAnalysisDefinition	analysis_definitionType	master_instance_result_responseType

Example:

```
<request_header>
    <result_waittime_ms>90000</result_waittime_ms>
</request_header>

<message_body>
    <crc:psmheader>
        <request_type>
            CRC_QRY_runQueryInstance_fromAnalysisDefinition
        </request_type>
    </crc:psmheader>

    <crc:request  xsi:type="crc:analysis_definition_requestType">
        <analysis_definition>
            <analysis_plugin_name>
                CALCULATE_PATIENTCOUNT_FROM_CONCEPTPATH
            </analysis_plugin_name>
            <crc_analysis_input_param name="ONT request">
                <param type="int" column="item_key">
                    \\rpdr\RPDR\Diagnoses\Circulatory system (390-459)\
                </param>
            </crc_analysis_input_param>
            <crc_analysis_result_list>
                <result_output full_name="XML" priority_index="1" name="XML"/>
            </crc_analysis_result_list>
        </analysis_definition>
    </crc:request>

    <crc:response xsi:type="crc:master_instance_result_responseType">
        <query_master>
            <query_master_id>0</query_master_id>
            <name>CALCULATE_PATIENTCOUNT_FROM_CONCEPTPATH</name>
            <user_id/>
            <group_id/>
            <create_date>2000-12-30T00:00:00</create_date>
            <request_xml/>
        </query_master>
```

```
            <query_instance>
                <query_instance_id>0</query_instance_id>
                <query_master_id>0</query_master_id>
                <user_id/>
                <group_id/>
                <batch_mode/>
                <start_date>2000-12-30T00:00:00</start_date>
                <end_date>2000-12-30T00:00:00</end_date>
                <query_status_type>
                    <status_type_id>6</status_type_id>
                    <name>COMPLETED</name>
                    <description/>
                </query_status_type>
            </query_instance>
            <query_result_instance>
                <result_instance_id>0</result_instance_id>
                <query_instance_id>0</query_instance_id>
                <query_result_type>
                    <query_result_type>
                        <result_type_id>3</result_type_id>
                        <name>XML</name>
                        <display_type>CATNUM</display_type>
                        <visual_attribute_type>LH</visual_attribute_type>
                        <description>Generic query result</description>
                    </query_result_type>
                    <set_size>0</set_size>
                    <obfuscate_method/>
                    <start_date>2000-12-30T00:00:00</start_date>
                    <end_date>2000-12-30T00:00:00</end_date>
                    <query_status_type>
                        <status_type_id>6</status_type_id>
                        <name>COMPLETED</name>
                        <description>COMPLETED</description>
                    </query_status_type>
                </query_result_instance>
        </crc:response>
    </message_body>
```

2.3.5.6 Scenario: Get a list of queries by user id

This request fetches a list of query master information for the given user id. The client can also specify how many query master items to return from the server using the *<fetch_size>* element. The server returns the query master items in descending order of query creation time.

Request Type	Request	Response
CRC_QRY_getQueryMasterList_fromUserId	user_requestType	master_responseType

Example:

```
<message_body>
    <psmheader>
        <request_type>
            CRC_QRY_getQueryMasterList_fromUserId
        </request_type>
    </psmheader>

    <request xsi:type="ns3:user_requestType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <user_id>user1</user_id>
        <fetch_size>100</fetch_size>
    </request>

    <response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ns5:master_responseType">
        <status>
            <condition type="DONE">DONE</condition>
        </status>
        <query_master>
            <query_master_id>6302</query_master_id>
            <name>1 y-Femal-Rheum@10:17:55</name>
            <user_id>demo</user_id>
            <group_id>Asthma</group_id>
            <create_date>2007-09-06T22:17:57.000-04:00</create_date>
        </query_master>
        <query_master>
            <query_master_id>6301</query_master_id>
            <name>10 ye-Female@10:42:41</name>
            <user_id>demo</user_id>
            <group_id>Asthma</group_id>
            <create_date>2007-09-06T10:42:42.000-04:00</create_date>
        </query_master>
    </response>
</message_body>
```

2.3.5.7 Scenario: Get metadata of analysis plug-in

This request will return the analysis plug-in's metadata by <*plugin_name*>. The <plugin_name> value of "ALL" is used to return all the plug-in's metadata.

Request Type	Request	Response
CRC_QRY_getAnalysisPluginMetadata	analysis_plugin_metadata_requestType	analysis_plugin_metadata_responseType

Example:

```
<request_header>
    <result_waittime_ms>90000</result_waittime_ms>
</request_header>

<message_body>
    <crc:psmheader>
        <request_type>
            CRC_QRY_getAnalysisPluginMetadata
        </request_type>
    </crc:psmheader>

    <crc:request  xsi:type="crc:analysis_plugin_metadata_requestType ">
        <plugin_name>ALL</plugin_name>
    </crc:request>

    <crc:response xsi:type="crc:analysis_plugin_metadata_responseType">
        <analysis_plugin_metadata_type>
            <plugin_id>1</plugin_id>
            <plugin_name>
                CALCULATE_PATIENTCOUNT_FROM_CONCEPTPATH
            </plugin_name>
            <description>Plugin description</description>
            <version_cd>1.0</version_cd>
            <parameter_info>
                <crc_analysis_input_param name="name0">
                    <param column="item_key" type="string">some value</param>
                </crc_analysis_input_param>
            </parameter_info>
```

```
            <parameter_info_xsd />
            <status_cd>A</status_cd>
            <command_line>
                /opt/jboss/server/default/analysis_commons_launcher/bin/co.sh
            </command_line>
            <working_folder>
                /opt/jboss/server/default/analysis_commons_launcher/bin
            </working_folder>
            <group_id>project_id</group_id>
        </analysis_plugin_metadata_type>
    </crc:response>
</message_body>
```

2.3.5.8 Scenario: Get query definition from master id

This request will return the *<query_definition>* information for the given query master id.

Request Type	Request	Response
CRC_QRY_getRequestXml_fromQueryMasterId	master_requestType	request_xml_responseType

Example:

```
<message_body>
    <psmheader>
        <request_type>
            CRC_QRY_getRequestXml_fromQueryMasterId
        </request_type>
    </psmheader>

    <request xsi:type="ns4:master_requestType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <query_master_id>6300</query_master_id>
    </request>

    <response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="request_xml_responseType ">
        <status>
            <condition type="DONE">DONE</condition>
```

```
</status>
<request_xml>
    <![CDATA[
    <query_definition>
        <query_name/>
        <query_description/>
        <query_timing>ANY</query_timing>
        <specificity_scale>0</specificity_scale>
        <query_date_from>2000-12-30T00:00:00</query_date_from>
        <query_date_to>2000-12-30T00:00:00</query_date_to>
        <panel>
            <panel_number>0</panel_number>
            <panel_date_from>2000-12-30T00:00:00</panel_date_from>
            <panel_date_to>2000-12-30T00:00:00</panel_date_to>
            <invert>0</invert>
            <total_item_occurrences>0</total_item_occurrences>
            <item>
                <hlevel>0</hlevel>
                <item_name/>
                <item_table/>
                <item_key/>
                <item_icon/>
                <tooltip/>
                <class/>
            </item>
        </panel>
    </query_definition>]]>
</request_xml>
</response>
</message_body>
```

2.3.5.9 Scenario: Rename a query

Use this request to change the name of the query. If the same user already has a query with the specified name, then the server will return an error in the *<status>* tag.

Request Type	Request	Response
CRC_QRY_renameQueryMaster	master_rename_requestType	master_responseType

Example:

```
<message_body>
    <psmheader>
        <user login="demo">demo</user>
        <patient_set_limit>0</patient_set_limit>
        <estimated_time>0</estimated_time>
        <request_type>CRC_QRY_renameQueryMaster</request_type>
    </psmheader>

    <request xsi:type="ns4:master_rename_requestType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <user_id>demo</user_id>
        <query_master_id>5997</query_master_id>
        <query_name>Demographics@03:21:10 -n[07-20-2007 ]</query_name>
    </request>

    <response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ns5:master_responseType">
        <status>
            <condition type="DONE">DONE</condition>
        </status>
        <query_master>
            <query_master_id>5997</query_master_id>
            <name>Demographics@03:21:10 -n[07-20-2007 ]</name>
            <user_id>demo</user_id>
        </query_master>
    </response>
</message_body>
```

2.3.5.10 Scenario: Delete a query

Use this request to remove a query and its results. Deleting it will not permanently remove the query; it will just set the delete flag to true.

Request Type	Request	Response
CRC_QRY_deleteQueryMaster	master_delete_requestType	master_responseType

Example:

```
<message_body>
    <psmheader>
        <user login="demo">demo</user>
        <patient_set_limit>0</patient_set_limit>
        <estimated_time>0</estimated_time>
        <request_type>CRC_QRY_deleteQueryMaster</request_type>
    </psmheader>

    <request xsi:type="ns4:master_delete_requestType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <user_id>demo</user_id>
        <query_master_id>5997</query_master_id>
    </request>

    <response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="ns5:master_responseType">
        <status>
            <condition type="DONE">DONE</condition>
        </status>
        <query_master>
            <query_master_id>5997</query_master_id>
        </query_master>
    </response>
</message_body>
```

2.3.5.11 Scenario: Cancel a query

The process of cancelling a query simply stops it from moving to the next queue. It does not stop the execution if the query has already started.

Request Type	Request	Response
CRC_QRY_cancelQuery	instance_requestType	instance_result_responseType

Example:

```
<message_body>
    <psmheader>
        <request_type>CRC_QRY_cancelQuery</request_type>
    </psmheader>

    <request xsi:type="ns4:instance_requestType">
        <query_instance_id>1013</query_instance_id>
    </request>

    <crc:response xsi:type="crc:instance_result_responseType">
        <query_instance>
            <query_instance_id>0</query_instance_id>
            <query_master_id>0</query_master_id>
            <user_id/>
            <group_id/>
            <batch_mode/>
            <start_date>2000-12-30T00:00:00</start_date>
            <end_date>2000-12-30T00:00:00</end_date>
            <query_status_type>
                <status_type_id>9</status_type_id>
                <name>CANCELLED</name>
                <description/>
            </query_status_type>
        </query_instance>
        <query_result_instance>
            <result_instance_id>0</result_instance_id>
            <query_instance_id>0</query_instance_id>
            <query_result_type>
                <query_result_type>
                    <result_type_id>3</result_type_id>
                    <name>XML</name>
                    <display_type>CATNUM</display_type>
                    <visual_attribute_type>LH</visual_attribute_type>
                    <description>Generic query result</description>
                </query_result_type>
                <set_size>0</set_size>
                <obfuscate_method/>
                <start_date>2000-12-30T00:00:00</start_date>
                <end_date>2000-12-30T00:00:00</end_date>
                <query_status_type>
                    <status_type_id>9</status_type_id>
                    <name>CANCELLED</name>
                    <description>CANCELLED</description>
                </query_status_type>
            </query_result_instance>
    </crc:response>
```

```
</message_body>
```

2.4 Patient Data Object Query Service

As the name suggests, these queries return patient data objects (PDO) in the response message as specified by the patient set and filter criteria defined in the request message.

2.4.1 Request and Response Message Structure

The PDO request / response message structure is divided into three parts:

1. pdoheader
2. Request
3. Response

For the request message, the <*pdoheader*> and <*request*> sections are required, while for the response message, only the <*response*> section is required.

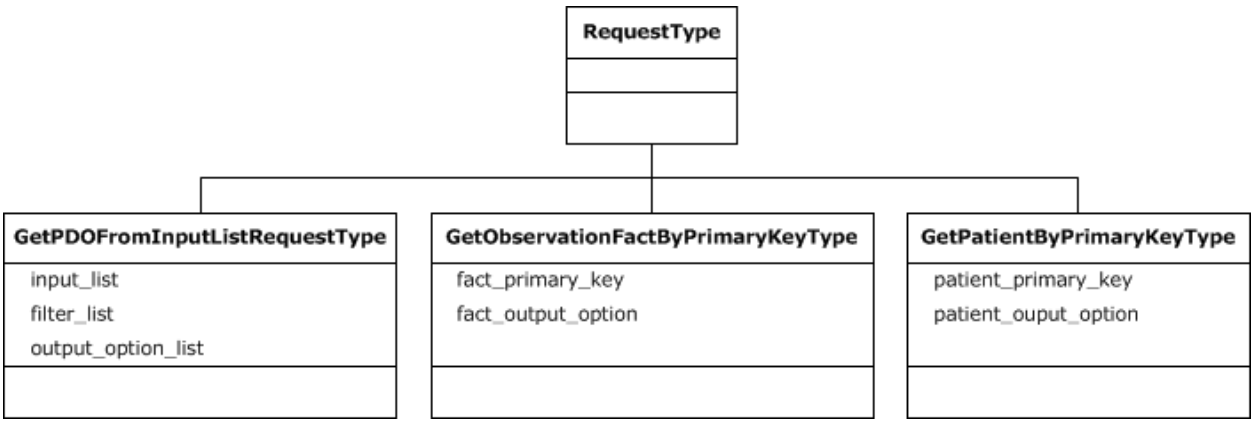
```
<i2b2:message_body>
    <crc:pdoheader>
        <request_type>GetPDOFromInputList_requestType</request_type>
        . . .
    </crc:pdoheader>
    <crc:request>
        . . .
    </crc:request>
    <crc:response>
        . . .
    </crc:response>
</i2b2:message_body>
```

Element Name	Description
pdoheader	The header contains a <request_type> element, which will carry the operation name. Each operation name has a specific request / response combination.

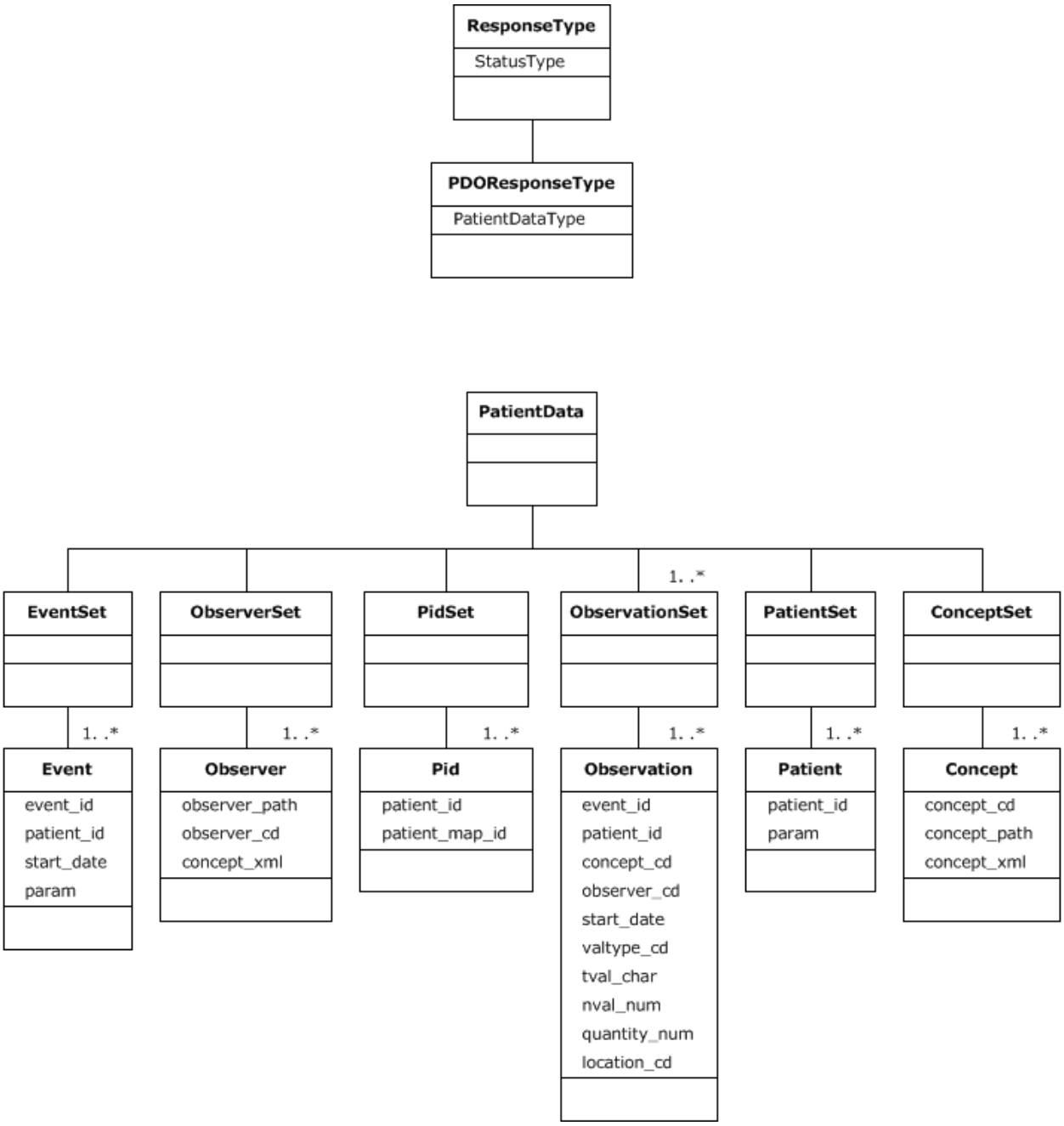
	The following is a list of supported operation names: • getPDO_fromInputList • get_observationfact_by_primary_key • get_patient_by_primary_key • get_event_by_primary_key • get_concept_by_primary_key • get_observer_by_primary_key
request	The <request> is modeled as an object using a polymorphic approach. All operation specific request objects inherit a base **RequestType *object*** as shown in the object model diagram in the next section.
response	The <response> is also modeled as an object using a polymorphic approach. All operation specific response objects inherit a base **PDOResponseType *object*** containing a **StatusType *attribute*** as shown in the object model diagram in the next section.

2.4.2 Request and Response Object Model

Request Object Model:



Response Object Model:



2.4.3 Use Case Scenarios

2.4.3.1 Scenario: Get patient data from a patient set id

This request is divided into three parts:

1. input_list

   The input_list accepts either the id of the patient set or a list of patient ids.

2. filter_list

   The filter_list holds a list of panels. The panels have the item details that are used in constructing a PDO query.

3. output_option

   The output_option specifies which set of patient data to return.

Each of the patient data sections in the *output_option* has attributes to specify the level of detail data that is expected in the response.

Request Type	Request	Response
getPDO_fromInputList	GetPDOFromInputList_requestType	master_responseType

2.4.3.1.1 INPUT OPTION LIST TYPE

The input list to the PDO request can be in one of the following formats. i.e. the patient_list, encounter_list, pid_list, and eid_list.

Element Name	Description
patient_list	The patient_list input can be in one of these three forms. 1. The patient set id (<patient_set_coll_id>) 2. An enumeration list of the patient id (<patient_id>) 3. The entire patients of datamart (<entire_patient_set>) ***Example for <patient_list>:*** <patient_list max="10" min="0"> <patient_id index="0">86850</patient_id> <!--OR-->

	<patient_set_coll_id>4741</patient_set_coll_id> <!--OR--> <entire_patient_set >true</entire_patient_set> </patient_list>
encounter_list	The encounter_list input format is similar to the patient_list. This list can be in one of three forms. 1. The encounter set id (<encounter_set_coll_id>) 2. An enumeration list of the encounter id (<encounter_id>) 3. The entire encounters of datamart (<entire_encounter_set>)
pid_list	The pid_list consists of a list of enumerated patient ids (patient_ide) and their source value. This input allows the user to query the data via the patient_ide and its source. The PDO service first maps the patient_ide to the patient_num before querying the datamart. ***Example for <pid_list>:*** <pid_list> <pid index="1" source="MGH_E">PAT_MGH_001</pid> <pid index="2" source="BWH_E">PAT_BWH_001</pid> </pid_list>
eid_list	The eid_list consists of a list of enumerated encounter ids (encounter_ide) and their source value. This input allows the user to query the data via the encounter_ide and its source. The PDO service first maps the encounter_ide to the encounter_num before querying the datamart. ***Example for <pid_list>:*** <eid_list> <eid index="1" source="MGH_E">ENC_MGH_001</pid> <eid index="2" source="BWH_E">ENC_BWH_001</pid> </eid_list>

2.4.3.1.2 FILTER LIST TYPE

Please refer to the earlier section titled *Panel Definition Detail*.

The LARGETEXT search in the panel definition requires the minimum of DATA_DEID user role.

2.4.3.1.3 OUTPUT OPTION LIST TYPE

This option specifies the list of sections that is required in the PDO response.

Attribute Name	Description	Implemented
name = ”asattributes | none”	This option specifies to return the names for the codes when returning the PDO data. The names for the codes are available in the CODE_LOOKUP table.	Yes
phi = ”encrypted | unencrypted”		No
time = ”nozone | withzone”		No

Example:

```
<!-- output options -->
<output_option name="asattributes">
    <patient_set select="using_filter_list" onlykeys="true"/>
    <concept_set_using_filter_list select="using_filter_list" onlykeys="true"/>
    <modifier_set_using_filter_list select="using_filter_list" onlykeys="true"/>
    <observation_set selectionfilter="min_value" withmodifiers="true" blob="false"
onlykeys="false"/>
    <event_set select="using_filter_list" onlykeys="true"/>
    <pid_set select="using_filter_list" onlykeys="true"/>
    <eid_set select="using_filter_list" onlykeys="true"/>
    <observer_set_using_filter_list onlykeys="true"/>
</output_option>
```

The following are the available options for each element in the <*output_option*>.

Attribute Name	Description	Implemented
onlykeys = ”true | false”	If this option is “true” then only the primary keys will be returned. If it is “false” then all the fields except the blob and tech data fields will be returned.	
blob = ”true | false”	This option specifies whether the blob field is required in the output. The blob option requires the minimum of	

	DATA_DEID user role.	
techdata = ”true | false”	This option specifies whether the blob field is required in the output. The blob option requires the minimum of DATA_DEID user role.	
select = ”using_filter_list | using_input_list”	There are a couple of ways to select the PDO data. If the select option is “*using_input_list*” then the returned data will be based on the <*input_list*> and the fact table will not be used to return the data. If it is “*using_filter_list*” then return the data that satisfies the <*filter_list*> constraints when applied on the fact table.	
selectionfilter = “min_value | max_value | first_value | last_value | single_observation | last_n_values”	This option will retrieve specific results depending on which one is defined in the xml message. If *min_value* is specified, then the minimum for the NVAL_NUM will be returned for each patient in the concept group. If *max_value* is specified, then the maximum for the NVAL_NUM will be returned for each patient in the concept group. If *first_value* is specified, then the minimum for the START_DATE will be returned for each patient in the concept group. If *last_value* is specified, then the maximum for the START_DATE will be returned for each patient in the concept group. If *single_observation* is specified, then the will be return just one record even if it contains more than one due to different values in the MODIFIER_CD column for each patient in the concept group. If *last_n_values* is specified then the last number of records in the concept group, where the value contains the number of records	

Element Name	**Description**
patient_set	Return the set of patient dimension data either for the input list or from the patient present in the observation set.

<table>
<tr><td></td><td></td></tr>
<tr><td>observation_set</td><td>Return the observation set of the patient data object. There could be a multiple number of <observation_set> returned and the number of <observation_set> returned will be equal to the number of panels defined in the filter list.

Observation set has the attribute “panel_name” which corresponds to “name” attribute defined in the <panel>.

For backward compatibility to 1.5 clients, which expects a unique observation fact not including the modifier option of the fact namely the INSTANCE_NUM column, the sending application version should have 1.5 in the PDO request.

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td>withmodifiers = “true | false”</td><td>If the value is “false” then the service will return the observation where the modifier_cd = ‘@’.

The default value is “true”, returns the observation irrespective of the modifier_cd value.</td></tr>
</table>
</td></tr>
<tr><td>ue</td><td>Return the set containing event / visit dimension data occurring in the observation set or from the input list.</td></tr>
<tr><td>pid_set</td><td>Return the patient mapping table’s information for the given input list or from the observation set.

<table>
<tr><th>patient_ide</th><th>patient_ide_source</th><th>patient_num</th><th>patient_ide_status</th></tr>
<tr><td>1000000</td><td>HIVE</td><td>1000000</td><td>A</td></tr>
<tr><td>1000001</td><td>HIVE</td><td>1000001</td><td>A</td></tr>
<tr><td>123</td><td>MGH</td><td>1000001</td><td>A</td></tr>
<tr><td>777</td><td>BWH</td><td>1000001</td><td>A</td></tr>
</table>

<pre>
<pid_set>
    <!--pid with only the hive number -->
    <pid>
        <patient_id status="A" source="HIVE">1000000</patient_id>
    </pid>
    <!--pidwithhivenumberandmappingpatient_ide’s-->
    <pid>
        <patient_id status="A" source="HIVE">1000001</patient_id>
        <patient_map_id status="A" source="MGH">123</patient_map_id>
        <patient_map_id status="A" source="BWH">777</patient_map_id>
    </pid>
</pid_set>
</pre>
</td></tr>
<tr><td>eid_set</td><td>Return the encounter mapping table’s information for the given input list or</td></tr>
</table>

	from the observation set.
observer_set_using_filter_list	Return the set containing observer / provider dimension data occurring in the observation set.
concept_set_using_filter_list	Return the set of concept dimension data occurring in the observation set.
modifier_set_using_filter_list	Return the set of modifier dimension data occurring in the observation set.

Example:

```
<message_body>

    <crc:pdoheader>
        <request_type>getPDO_fromInputList</request_type>
    </crc:pdoheader>

    <crc:request xsi:type="ns2:GetPDOFromInputList_requestType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

        <!-- inputlist -->
        <input_list>
            <patient_list max="300" min="1">
                <patient_set_coll_id>100</patient_set_coll_id>
            </patient_list>
            <!--or-->
            <event_list/>
            <!--or-->
            <pid_list min="1" max="3">
                <pid index="1" source="HIVE">9876</pid>
                <pid index="3" source="HIVE">1</pid>
            </pid_list>
            <!--or-->
            <eid_list min="1" max="5">
                <eid index="1" source="HIVE">12222313</eid>
            </eid_list>
        </input_list>

        <!-- filterlist -->
        <filter_list>
            <panel name="panel1">
                <panel_invert>0</panel_invert>
                <panel_accuracy_scale></panel_accuracy_scale>
                <panel_start_date></panel_start_date>
```

```
            <panel_end_date></panel_end_date>
            <item>
                <item_name></item_name>
                <item_key></item_key>
                <constrain_by_value>
                    <value_operator></value_operator>
                    <value_constraint>value_constraint0</value_constraint>
                    <value_unit_of_measure>unit</value_unit_of_measure>
                    <value_type></value_type>
                </constrain_by_value>
                <constrain_by_modifier>
                    <applied_path>\i2b2\Medications\%</applied_path>
                    <modifier_key>\\i2b2_DEMO\Dose\</modifier_key>
                    <constrain_by_value>
                        <value_operator></value_operator>
                        <value_constraint> </value_constraint>
                        <value_unit_of_measure></value_unit_of_measure>
                        <value_type></value_type>
                    </constrain_by_value>
                </constrain_by_modifier>
                <constrain_by_date>
                </constrain_by_date>
            </item>
            . . .
        </panel>

        <panel name="panel2">
            <panel_invert>0</panel_invert>
            <panel_accuracy_scale></panel_accuracy_scale>
            <panel_start_date></panel_start_date>
            <panel_end_date></panel_end_date>
            <item>
                <item_name> </item_name>
                <item_key> </item_key>
                <constrain_by_value>
                    <value_type></value_type>
                    <value_operator></value_operator>
                    <value_unitofmeasure></value_unitofmeasure>
                    <value></value>
                </constrain_by_value>
                <constrain_by_modifier></constrain_by_modifier>
                <constrain_by_date></constrain_by_date>
            </item>
        </panel>
        . . .
    </filter_list>
```

```
        <!-- output options -->
        <output_option>
            <patient_set select="using_filter_list" onlykeys="true"/>
            <concept_set_using_filter_list select="using_filter_list" onlykeys="true"/>
            <modifier_set_using_filter_list select="using_filter_list" onlykeys="true"/>
            <observation_set blob="false" onlykeys="false"/>
            <event_set select="using_filter_list" onlykeys="true"/>
            <pid_set select="using_filter_list" onlykeys="true"/>
            <eid_set select="using_filter_list" onlykeys="true"/>
            <observer_set_using_filter_list onlykeys="true"/>
            <!-- To specify generalized dimension type -->
            <dimension_set_using_filter_list dimensionname="dimension1"
onlykeys="true"/>
        </output_option>
    </crc:request>

    <!-- response begin -->
    <response>
        <patient_data>
            <!-- patient set section begins -->
            <patient_set>
                <patient>
                    <patient_id>patient_id6</patient_id>
                    <param column="vital_status_cd" type="string"
                        column_descriptor="date interpretation code"
                        name=" ">param3</param>
                    <param column="birth_date" type="dateTime"
                    column_descriptor="birthdate"
                    name="">param3</param>
                    <param column="death_date" type="dateTime"
                    column_descriptor="date of death">param3</param>
                    <param column="sex_cd"  type="string"
                    column_descriptor="gender" name="">param3</param>
                    <param column="age_in_years_num" type="int"
                    column_descriptor="age">param3</param>
                    <param column="language_cd" type="string"
                    column_descriptor="language" name="">param3</param>
                    <param column="race_cd" type="string"
                    column_descriptor ="race" name="">param3</param>
                    <param column="religion_cd" type="string"
                    column_descriptor ="religion" name="">param3</param>
                    <param column="marital_status_cd" type="string"
                    column_descriptor ="marital status" name="">param3</param>
                    <param column="statecityzip_path_char" type="string"
                    column_descriptor="zip code hierarchy"
```

```
        name="" >param3</param>
    </patient>
    . . .
</patient_set>

<!-- concept set section begins -->
<concept_set>
    <concept>
        <concept_path>concept_path0</concept_path>
        <concept_cd>concept_cd0</concept_cd>
        <name_char>name_char0</name_char>
    </concept>
    . . .
</concept_set>

<!-- observation set section begins -->
<observation_set panel_name="panel1">
    <observation>
        <event_id source="source3">event_id3</event_id>
        <patient_id>patient_id9</patient_id>
        <concept_cd name="name0">concept_cd3</concept_cd>
        <observer_cd soruce="soruce0">observer_cd3</observer_cd>
        <start_date>2006-05-04T18:13:51.0Z</start_date>
        <modifier_cd name="name1">modifier_cd0</modifier_cd>
        <instance_num>1</instance_num>
        <valuetype_cd>valuetype_cd0</valuetype_cd>
        <tval_char>tval_char0</tval_char>
        <nval_num units="units0">3.141592653589</nval_num>
        <valueflag_cd name="name2">valueflag_cd0</valueflag_cd>
        <quantity_num>3.141592653589</quantity_num>
        <units_cd>units_cd0</units_cd>
        <end_date>2006-05-04T18:13:51.0Z</end_date>
        <location_cd name="name3">location_cd0</location_cd>
    </observation>
    <observation>
        . . .
    </observation>
</observation_set>

<observation_set panel_name="panel2">
    <observation>
        . . .
    </observation>
    <observation>
        . . .
    </observation>
```

```
            . . .
        </observation_set>

        <!-- event set section begins -->
        <event_set>
            <event>
                <event_id source="source0">event_id0</event_id>
                <patient_id>patient_id0</patient_id>
                <start_date>2006-05-04T18:13:51.0Z</start_date>
                <end_date>2006-05-04T18:13:51.0Z</end_date>
                <param column="inout_cd" type="string" name=""
                column_descriptor="in vs. outpatient code"></param>
                <param column="location_cd" type="string" name=""
                column_descriptor="location code"></param>
                <param column="location_path" type="string" name=""
                column_descriptor="location hierarchy"></param>
                <param column="active_status_cd" type="string" name=""
                column_descriptor="date interpretation code"></param>
            </event>
            . . .
        </event_set>

        <!-- observer / provider set section begins -->
        <observer_set>
            <observer>
                <observer_path>observer_path0</observer_path>
                <observer_cd>observer_cd0</observer_cd>
                <name_char>name_char3</name_char>
            </observer>
            . . .
        </observer_set>

        <!-- pid set section begins -->
        <pid_set>
            <pid>
                <patient_id status="A" source="HIVE">123456</patient_id>
            </pid>
            <pid>
                <patient_id status="A" source="HIVE">234567</patient_id>
                <patient_map_id status="A" source="EMP_E">ABC1</patient_map_id>
                <patient_map_id status="A" source="MGH_E">XYZ1</patient_map_id>
            </pid>
            . . .
        </pid_set>
    </patient_data>
</response>
```

```
    <!-- response end -->
</message_body>
```

2.4.3.2 Scenario: Get observation blob by primary key

This request returns the observation blob using the observation primary key.

Example:

```
<message_body>
    <pdoheader>
        <request_type>get_observationfact_by_primary_key</request_type>
    </pdoheader>

    <ns5:request xsi:type="ns5:GetObservationFactByPrimaryKey_requestType"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <fact_primary_key>
            <event_id>2004005981</event_id>
            <patient_id>52003</patient_id>
            <concept_cd>LCS-I2B2:c1009c</concept_cd>
            <observer_id>03840261</observer_id>
            <start_date>1995-08-24T00:00:00.179-05:00</start_date>
            <instance_num>1</instance_num>
        </fact_primary_key>
        <fact_output_option select="using_filter_list" onlykeys="false"/>
    </ns5:request>

    <response>
        <patient_data>
            <observation_set>
                <observation>
                    <event_id source="source3">event_id3</event_id>
                    <patient_id>patient_id9</patient_id>
                    <concept_cd name="name0">concept_cd3</concept_cd>
                    <observer_cd soruce="soruce0">observer_cd3</observer_cd>
                    <start_date>2006-05-04T18:13:51.0Z</start_date>
                    <instance_num>1</instance_num>
                    <observation_blob>
                        <![CDATA[patient notes]]>
                    </observation_blob>
                </observation >
```

```
            </observation_set>
        </patient_data>
    </response>
</message_body>
```

2.5 Data Upload Messages

The data loading process loads the data to three main sections of data mart.

1. Dimension tables
2. Mapping tables
3. Observation fact table

> **Note**
>
> In order to upload data a user must have the role of DATA_DEID.

Except for the OBSERVATION_FACT table, the data load will perform the insert or update operation based on two criteria: (1) whether the data exists in the respective tables and (2) whether the PDO's update date is greater than the existing data's update date.

For example, the following table shows the conditions for both the update and insert operations for the VISIT_DIMENSION table.

Loading VISIT_DIMENSION	Condition			
UPDATE	INPUT.ENCOUNTER_NUM	=	OBSFACT.ENCOUNTER_NUM	AND
	INPUT.PATIENT_NUM	=	OBSFACT.PATIENT_NUM	AND
	INPUT.UPDATE_DATE	>	OBSFACT.UPDATE_DATE	
INSERT	INPUT.ENCOUNTER_NUM	!=	OBSFACT.ENCOUNTER_NUM	OR
	INPUT.PATIENT_NUM	!=	OBSFACT.PATIENT_NUM	

The data load on the OBSERVATION_FACT table is slightly different compared to the other tables and it can be divided into two cases.

Case 1: Refresh the Observation Fact

This case assumes the user is trying to load a fresh set of observation facts and expects to delete any old observation facts that have matching ENCOUNTER_NUM and PATIENT_NUM.

Set append_flag=”false” to support this case.

Case 2: Incremental Observation Fact

In this case the user has the option to just add new observation facts that may match the ENCOUNTER_NUM and PATIENT_NUM of an existing OBSERVATION_FACT entry.

Set append_flag=”true” to support this case.

The next section, *i2b2 Import: Append Flag*, explains the rules for importing observations facts into the OBSERVATION_FACT table.

2.5.1 i2b2 Import: Append Flag

Assumption: *the record(s) in the update file (new record) has the same primary key as a record(s) in the associated table (existing record).*

Primary Key includes:

Encounter number

Patient number

Concept code

Start date

Modifier code

Observer code

Append Flag = True

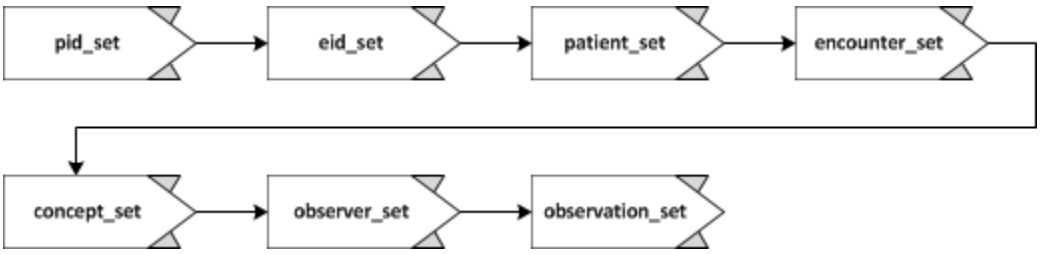
The following conditions will result in the new record **replacing** the existing record:

new record update date	equal to (=)		update date on the existing record	
new record update date	greater than (>)		update date on the existing record	
new record update date	is not null	AND	update date on the existing record	null
new record update date	null	AND	update date on the existing record	null

The following conditions will result in **ignoring** the new record and **not** updating the existing record:

new record update date	less than (<)		update date on the existing record	
new record update date	null	AND	update date on the existing record	is not null

The loader process loads the PDO data in a certain order, which is based on the data dependency of the data mart. The order of the data load is shown in the below diagram.



2.5.2 Use Case Scenarios

2.5.2.1 Scenario: Upload data in patient data object XML

The data load request message can be divided into three sections.

1. The <input_list> specifies the location of the PDO document.
2. The <load_list> specifies the PDO sections to load.
3. The <output_list> specifies the output expected from the upload process.

In addition, certain dependencies exist for loading some section of the PDO. The following table shows the dependencies for loading the individual PDO section.

Load PDO section	Dependent PDO section
event_set	eid_set
patient_set	pid_set
observation_set	eid_set, pid_set, event_set, patient_set
concept_set, modifier_set, observer_set, eid_set, pid_set	NONE

The example shown below contains the sections of the PDO XML that are required to load a single observation fact.

> **Note**
>
> The dependency section in the PDO is optional; if they exist in the data mart

For a more detailed example on loading patient and encounter mapping, please refer to the *CRC Design Document*.

```
<patient_data>
    <observation_set>
        <observation update_date="2006-05-04T18:13:51.0Z"
        sourcesystem_cd="TEST">
            <event_id source="MGHE">MENT001</event_id>
            <patient_id source="MGHP">MPAT001</patient_id>
            <concept_cd name="C">i2b2CD</concept_cd>
            <observer_cd source="TEST">@</observer_cd>
            <start_date>2006-05-04T18:13:51.0Z</start_date>
            <instance_num>1</instance_num>
            <modifier_cd>modifier_cd</modifier_cd>
            <valuetype_cd>valuetype_cd0</valuetype_cd>
            <nval_num units="units0">3.141592653589</nval_num>
            <valueflag_cd name="name8">valueflag_cd0</valueflag_cd>
            <quantity_num>3.141592653589</quantity_num>
            <confidence_num>3.141592653589</confidence_num>
        </observation>
    </observation_set>
    <pid_set>
        <pid>
            <patient_id status="A" update_date="2006-05-04T18:13:51.0Z"
            sourcesystem_cd="TEST" source="MGHP">MPAT001</patient_id>
        </pid>
    </pid_set>
    <eid_set>
        <eid>
            <event_id source="MGHE" patient_id="MPAT001"
            patient_id_source="MGHP" status="A"
            update_date="2006-05-04T18:13:51.0Z"
            sourcesystem_cd="TEST">MENT001</event_id>
        </eid>
    </eid_set>
    <event_set>
        <event update_date="2006-05-04T18:13:51.0Z"
        sourcesystem_cd="TEST">
            <event_id source="MGHE">MENT001</event_id>
            <patient_id source="MGHP">MPAT001</patient_id>
            <start_date>2006-05-04T18:13:51.0Z</start_date>
            <end_date>2006-05-04T18:13:51.0Z</end_date>
        </event>
    </event_set>
</patient_data>
```

Request Message:

```
<message_body>
    <publish_data_request>
        <input_list>
            <data_file>
                <location_uri protocol_name="FR|LOCAL">location_uri0</location_uri>
                <data_format_type>PDO</data_format_type>
                <source_system_cd>source_system_cd0</source_system_cd>
                <load_label>load_label0</load_label>
            </data_file>
        </input_list>
        <load_list commit_flag="true" clear_temp_load_tables="false">
            <load_observation_set ignore_bad_data="true"/>
            <load_event_set ignore_bad_data="true"/>
            <load_patient_set ignore_bad_data="true"/>
            <load_eventid_set ignore_bad_data="true"/>
            <load_pid_set ignore_bad_data="true"/>
            <load_eid_set ignore_bad_data="true"/>
            <load_concept_set encrypt_blob="false" ignore_bad_data="true"
delete_existing_data="true|false"/>
            <load_modifier_set encrypt_blob="false" ignore_bad_data="true"
delete_existing_data="true|false"/>
            <load_observer_set ignore_bad_data="true" delete_existing_data="true|false"/>
        </load_list>
        <output_list detail="true">
            <observation_set onlykeys="true" blob="false" techdata="false"/>
            <patient_set onlykeys="true" blob="false" techdata="false"/>
            <event_set onlykeys="true" blob="false" techdata="false"/>
            <observer_set onlykeys="true" blob="false" techdata="false"/>
            <concept_set onlykeys="true" blob="false" techdata="false"/>
            <modifier_set onlykeys="true" blob="false" techdata="false"/>
            <pid_set onlykeys="true" blob="false" techdata="false"/>
            <eid_set onlykeys="true" blob="false" techdata="false"/>
            <eventid_set onlykeys="true" blob="false" techdata="false"/>
        </output_list>
    </publish_data_request>
</message_body>
```

Element Name	Description
<input_list>	**This section of xml will carry input information such as the location of the PDO file, its data format, etc.**

location_uri	The location of the input PDO file.
protocol_name	Protocol name that specifies the service for how to access the PDO file in the location_uri. FR – File Repository LOCAL – File resides in the server's local folder
data_format_type	PDO (Patient Data Object XML)
source_system_cd	Upload's the code for the source system
load_label	The user's load label
<load_list>	**This section of the request holds the load options.**
commit_flag	The current implementation supports only a value of "true". This is a placeholder for future releases. If the flag is set to false, the system will just run all the load process for the given PDO data and will not perform commit. This is a useful option to check, whether the given PDO data, have the valid information.
clear_temp_load_tables	This flag specifies whether to clean up the staging area. Set this flag to false when you need to debug.
ignore_bad_data	Currently this is not implemented. It will be part of a future release.
load_observation *set, append_flag*	Load or update the fact entry coming from the input PDO data. If the append_flag = 'true' is specified, then the PDO's fact information is just added to the stored fact and will not do any updates.
load_event_set	The information from the PDO's <event_set> is used to load the data to the VISIT_DIMENSION table.
load_patient_set	The information from the PDO's <patient_set> is used to load the data to the PATIENT_DIMENSION table.
load_observer_set *set delete_existing_data*	The information from the PDO's <observer_set> is used to load the data to the PROVIDER_DIMENSION table.
load_pid_set	The information from the PDO's <pid_set> is used to load the data to the PATIENT_MAPPING table.
load_eid_set	The information from the PDO's <eventid_set> is used to load the data to the ENCOUNTER_MAPPING table.
load_concept_set *set delete_existing_data*	The information from the PDO's <concept_set> is used to load the data to the CONCEPT_DIMENSION table. Set the delete_existing_flag to remove the old data before loading the new data and the old data will be available in the back up table.
load_modifier_set *set delete_existing_data*	The information from the PDO's <modifier_set> is used to load the data to the MODIFIER_DIMENSION table. Set the delete_existing_flag to remove the old data before loading the new data and the old data will be available in the back up table.

<output_list>	This section of the request is used for getting the upload process status information.
detail	This attribute is a placeholder for a future release. The attribute specifies whether the response message should have detailed information. If the value is false, then the service will return only the count for the number of inserted and updated records. Otherwise a separate file with the inserted and updated records will be created.
observation_set	This element specifies the process to fetch the details of updated records on the OBSERVATION_FACT table.
onlykeys	Return the primary key fields in the response
blob	Return the blob field in the response
techdata	Return the techdata fields in the response
patient_set	The element specifies the process to fetch the details of updated records on the PATIENT_DIMENSION table.
event_set	The element specifies the process to fetch the details of updated records on the VISIT_DIMENSION table.
observer_set	The element specifies the process to fetch the details of updated records on the PROVIDER_DIMENSION table.
concept_set	The element specifies the process to fetch the details of updated records on the CONCEPT_DIMENSION table.
modifier_set	The element specifies the process to fetch the details of updated records on the MODIFIER_DIMENSION table.
pid_set	The element specifies the process to fetch the details of updated records on the PATIENT_MAPPING table.
eid_set	The element specifies the process to fetch the details of updated records on the ENCOUNTER_MAPPING table.

Response Message:

```
<message_body>
    <load_data_response>
        <status>
            <condition type="ERROR|DONE" coding_system="">
                error message
            </condition>
        </status>
        <upload_id>upload_id0</upload_id>
```

```
            <user_id>user_id0</user_id>
            <data_file_location_uri protocol_name="FR|LOCAL">
                location_uri0
            </data_file_location_uri>
            <load_status>load_status0</load_status>
            <transformer_name>transformer_name0</transformer_name>
            <start_date>2006-05-04T18:13:51.0Z</start_date>
            <end_date>2006-05-04T18:13:51.0Z</end_date>
            <message>message0</message>
            <observation_set inserted_record="0" ignored_record="0" total_record="0">
                <ignored_patient_data_file_uri protocol_name="FR|LOCAL">
                    uri0
                </ignored_patient_data_file_uri>
                <message>message1</message>
            </observation_set>
            <patient_set inserted_record="0" ignored_record="0" total_record="0">
                <ignored_patient_data_file_uri protocol_name="FR|LOCAL">
                </ignored_patient_data_file_uri>
                <message>message2</message>
            </patient_set>
            <event_set inserted_record="0" ignored_record="0" total_record="0">
            </event_set>
            <observer_set inserted_record="0" ignored_record="0" total_record="0">
            </observer_set>
            <concept_set inserted_record="0" ignored_record="0" total_record="0">
            </concept_set>
            <modifier_set inserted_record="0" ignored_record="0" total_record="0">
            </modifier_set>
            <pid_set inserted_record="0" ignored_record="0" total_record="0">
            </pid_set>
            <eventid_set inserted_record="0" ignored_record="0" total_record="0">
            </eventid_set>
        </load_data_response>
    </message_body>
```

2.5.2.2 Scenario: Get upload status info by upload_id and user_id

This message will retrieve one of the following:

1. A list of upload status information based on the user_id

2. A single upload status by upload_id

Request Message:

```
<message_body>
    <get_upload_info_request>
        <user_id>user_id</user_id>
        <upload_id>100</upload_id>
    </get_upload_info_request>
</message_body>
```

Response Message:

```
<message_body>
    <load_data_list_response>
        <load_data_response>
            Please refer section 4.5.1.1 for <load_data_response/>
        </load_data_response>
        . . .
    </load_data_list_response>

</message_body>
```

2.5.2.3 Scenario: Find the unmapped term in the datamart

This service returns a count for the number of concepts, modifiers and providers that considered “unmapped”. A term is “unmapped” when it is found in the fact table and is missing from the corresponding dimension table.

The unmapped term can be requested for a particular upload or for the entire data mart. If detail=”true” then the service will return the list of unmapped terms.

Request Message:

```
<message_body>
    <get_missing_term_request upload_id="NN" detail="true | false">
        <concept_set start="1" end="1000" />
        <modifier_set start="1" end="1000" />
        <observer_set start="1" end="1000" />
    </get_missing_term_request>
</message_body>
```

Response Message:

```
<message_body>
    <missing_term_report_response>
        <missing_term_report upload_id="22">
            <concept_set mapped="41683182" unmapped=""/>
            <modifier_set mapped="41683182" unmapped="" />
            <observation_set mapped="41683182" unmapped=""/>
        </missing_term_report>
        <missing_codes>
            <concept_set>
                <concept missing_total="9">
                    <concept_cd>brine:ses5</concept_cd>
                </concept>
                <concept missing_total="5">
                    <concept_cd>MTP:KGC</concept_cd>
                </concept>
            </concept_set>
            <observer_set>
                <observer/>
            </observer_set>
            <modifier_set>
                <modifier/>
            </modifier_set>
        </missing_codes>
    </missing_term_report_response >
</message_body>
```

2.6 Database Lookup Services / Messages

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

Service	XML Message
QueryToolService/**getAllDblookups**	get_all_dblookups
QueryToolService/**setDblookup**	set_dblookup
QueryToolService/**getDblookup**	get_dblookup
QueryToolService/**deleteDblookup**	delete_dblookup

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

2.6.1 Restrictions

The role of **ADMIN** is required when running any of the *DBLookup messages*. The CRC 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.

2.6.2 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 CRC cell.

2.6.2.1 Processing by the CRC Cell

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

STEP 1: Send Request Message

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

STEP 2: Verify User Access

The CRC 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 CRC 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 **CRC_DB_LOOKUP** table that meet the following criteria.

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

   Example:

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

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

STEP 3: Send Response Message

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

2.6.2.2 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/QueryToolService/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>
```

2.6.2.2.1 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/QueryToolService/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

2.6.2.3 Request Message: get_all_dblookups

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

2.6.2.3.1 ATTRIBUTES

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

Attribute Name	Description
type	Always set to "default".

2.6.2.3.2 XML ELEMENTS

<get_all_dblookups>	Container for get_all_dblookups data request

2.6.2.4 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>
```

2.6.2.4.1 XML ELEMENTS

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

Tip

Please refer to the *CRC_Architecture* document for additional information about the CRC_DB_LOOKUP table in the i2b2 Database.

2.6.2.5 Use Cases

2.6.2.5.1 ADMIN USER REQUESTS ALL DB LOOKUPS

The get_all_dblookups service sends a request message to the CRC 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 CRC_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/QueryToolService/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">CRC 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>
```

2.6.2.5.2 NON-ADMIN USER REQUESTS ALL DB LOOKUPS

The get_all_dblookups service sends a request message to the CRC 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 CRC_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/QueryToolService/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>
```

2.6.3 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.

2.6.3.1 Processing by the CRC Cell

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

STEP 1: Verify User Access

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

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

   Example:

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

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

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

   Example:

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

2.6.3.2 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/QueryToolService/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">CRC processing completed</status>
        </result_status>
    </response_header>
    <message_body>
        ...
    </message_body>
</ns5:response>
```

2.6.3.2.1 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/QueryToolService/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

2.6.3.3 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>
```

2.6.3.3.1 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 CRC_DB_LOOKUP table.

2.6.3.3.2 XML ELEMENTS

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

Tip

Please refer to the *CRC_Architecture* document for additional information about the CRC_DB_LOOKUP table in the i2b2 Database.

2.6.3.3.3 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 CRC.

- project_path (attribute)
- domain_id
- owner_id
- db_fullschema

- db_datasource
- db_servertype
- db_nicename

2.6.3.4 Response Message: set_dblookup

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

2.6.3.5 Use Cases

2.6.3.5.1 ADMIN USER REQUESTS TO ADD A NEW OR EDIT AN EXISTING RECORD

The **set_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/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">CRC processing completed</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.6.3.5.2 NON-ADMIN USER REQUESTS TO ADD A NEW OR EDIT AN EXISTING RECORD

The **set_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/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">CRC processing completed</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.6.3.5.3 REQUIRED INFORMATION NOT SENT IN REQUEST

The **set_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/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>
```

2.6.4 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.

2.6.4.1 Processing by the CRC Cell

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

STEP 1: Verify User Access

The CRC 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 CRC 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 CRC_DB_LOOKUP table. The criteria to find an existing entry in the CRC_DB_LOOKUP table.

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

   Example:

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

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

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

   Example:

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

STEP 3: Send Response Message

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

2.6.4.2 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/QueryToolService/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">CRC processing completed</status>
        </result_status>
```

```
    </response_header>
    <message_body>
        ...
    </message_body>
</ns5:response>
```

2.6.4.2.1 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/QueryToolService/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

2.6.4.3 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 CRC Cell.

2.6.4.3.1 ATTRIBUTES

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

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

2.6.4.3.2 XML ELEMENTS

<get_dblookup>	**Container for get_dblookup data request**

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

2.6.4.3.3 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 CRC.

- 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 CRC Cell:

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

2.6.4.4 Response Message: get_dblookup

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

2.6.4.4.1 XML ELEMENTS

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

Tip

Please refer to the *CRC_Architecture* document for additional information about the CRC_DB_LOOKUP table in the i2b2 Database.

2.6.4.5 Use Cases

2.6.4.5.1 ADMIN USER REQUESTS DATA ON A SPECIFIC DATABASE CONNECTION

The **get_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/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">CRC 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>
```

2.6.4.5.2 NON-ADMIN USER REQUESTS DATA ON A SPECIFIC DATABASE CONNECTION

The **get_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/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>
```

2.6.4.5.3 REQUESTED DATA NOT FOUND

The **get_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/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! - CRC processing
completed</status>
    </result_status>
</response_header>
```

2.6.4.5.4 REQUIRED INFORMATION NOT SENT IN REQUEST

The **get_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/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>
```

2.6.5 delete_dblookup

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

2.6.5.1 Processing by the CRC Cell

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

STEP 1: Verify User Access

The CRC 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 CRC 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 CRC_DB_LOOKUP table. The criteria to find an existing entry in the CRC_DB_LOOKUP table.

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

   Example:

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

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

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

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

   Example:

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

STEP 3: Send Response Message

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

2.6.5.2 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/QueryToolService/dele
teDblookup</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">CRC processing completed</status>
        </result_status>
    </response_header>
    <message_body>
        ...
    </message_body>
</ns5:response>
```

2.6.5.2.1 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/QueryToolService/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

2.6.5.3 Request Message: delete_dblookup

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

2.6.5.3.1 ATTRIBUTES

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

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

2.6.5.3.2 XML ELEMENTS

<delete_dblookup>	Container for delete_dblookup data request

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

2.6.5.3.3 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 CRC.

- project_path
- domain_id
- owner_id

2.6.5.4 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>
```

2.6.5.5 Use Cases

2.6.5.5.1 ADMIN USER REQUESTS DELETION OF RECORD

The **delete_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/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">CRC processing completed</status>
        </result_status>
    </response_header>
</ns5:response>
```

2.6.5.5.2 NON-ADMIN USER REQUESTS DELETION OF RECORD

The **delete_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/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>
```

2.6.5.5.3 REQUESTED RECORD DOESN'T EXIST

The **delete_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/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)! - CRC processing completed</status>
    </result_status>
</response_header>
```

2.6.5.5.4 REQUIRED INFORMATION NOT SENT IN REQUEST

The **delete_dblookup** service sends a request message to the CRC 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 CRC_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/QueryToolService/deleteDblookup
</redirect_url>
        </proxy>
        ...
    </message_header>
    <message_body>
        <pm:delete_dblookup domain_id="i2b2demo" owner_id="@"/>
    </message_body>
</i2b2:request>
```

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

Response Message:

```
<ns5:response>
```

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

2.7 Message Explanations

This section defines the message elements in the CRC namespace (http://www.i2b2.org/xsd/cell/crc/psm/1.1/ and http://www.i2b2.org/xsd/cell/crc/pdo/1.1/

Each element defined will have an implied prefix of **crc:** unless another namespace is explicitly stated. Elements from other namespaces that are included within the CRC elements will be listed but not expanded or defined in this document. Please refer to the documents for the other cells to get specific details on those elements.

2.7.1 Header

The <header> is the first CRC element within an i2b2 <message_body>. This section defines the elements shown below in the example header.

```
<crc:header>
    <user login=""></user>
    <data_source></data_source>
    <patient_set_limit>0</patient_set_limit>
    <estimated_time>0</estimated_time>
    <create_date>2002-12-23T00:00:00</create_date>
    <submit_date>2002-12-23T00:00:00</submit_date>
    <complete_date>2002-12-23T00:00:00</complete_date>
    <request_type>getPDO_fromObservationFact</request_type>
</crc:header>
```

Element Name	Description
header	The container for generic information useful for any CRC message
user	User information used for authentication and login
data_source	Information about the source of the data
patient_set_limit	Limit the size of the patient set returned in a query
estimated_time	The time estimated for the query to be completed.
create_date	The date the query was created.
submit_date	The data the query was submitted to be executed or run.
complete_date	The date a query finished executing.
request_type	A code that tells the service what type of request to expect, which tells it what kind of xml to expect in the rest of the message.

2.7.2 Requests

2.7.2.1 xsi:type=”crc:user_requestType”

```
<crc:request xsi:type="crc:user_requestType">
    <user_id>some_user_id</user_id>
    <group_id>some_group_id</group_id>
    <fetch_size>5000</fetch_size>
</crc:request>
```

2.7.2.2 xsi:type=”crc:master_requestType”

```
<crc:request xsi:type="crc:master_requestType">
    <query_master_id>0</query_master_id>
</crc:request>
```

2.7.2.3 xsi:type=”crc:instance_requestType”

```
<crc:request xsi:type="crc:instance_requestType">
    <query_instance_id>0</query_instance_id>
</crc:request>
```

2.7.2.4 xsi:type=”crc:query_definition_requestType”

```
<crc:request xsi:type="crc:query_definition_requestType">
    <query_definition>
        <query_name/>
        <query_description/>
        <query_timing>SAME</query_timing>
        <specificity_scale>0</specificity_scale>
        <query_date_from>2000-12-30T00:00:00</query_date_from>
        <query_date_to>2000-12-30T00:00:00</query_date_to>
        <panel>
            <panel_number>0</panel_number>
            <panel_date_from>2000-12-30T00:00:00</panel_date_from>
            <panel_date_to>2000-12-30T00:00:00</panel_date_to>
            <invert>0</invert>
            <total_item_occurrences>0</total_item_occurrences>
            <item>
                <hlevel>0</hlevel>
                <item_name/>
                <item_table/>
                <item_key/>
                <item_icon/>
                <tooltip/>
                <class/>
            </item>
        </panel>
    </query_definition>
</crc:request>
```

2.7.2.5 xsi:type=”crc:getPDO_FromInputList”

```
<crc:request xsi:type="crc:patient_set_requestType">
    <select_option_list>
        <observation_fact blob="true" before="2005-12-30T00:00:00" after="2003-12-
30T00:00:00"/>
        <patient_dimension fact_related="false"/>
        <provider_dimension/>
        <visit_dimension detail="false"/>
        <concept_dimension status="true"/>
    </select_option_list>
    <filter_list>
        <panel name="panel1">
            <panel_invert>0</panel_invert>
            <panel_accuracy_scale></panel_accuracy_scale>
            <panel_start_date>
                <panel_start_date>
                    <panel_end_date></panel_end_date>
                    <item>
                        <item_name></item_name>
                        <item_key></item_key>
                        <item_icon></item_icon>
                        <item_tooltip></item_tooltip>
                        <dim_tablename></dim_tablename>
                        <dim_columnname></dim_columnname>
                        <dim_dimcode></dim_dimcode>
                        <dim_columndatatype></dim_columndatatype>
                        <dim_operator></dim_operator>
                        <facttablecolumn></facttablecolumn>
                        <value_constraint>
                            <value_type></value_type>
                            <value_operator></value_operator>
                            <value_unitofmeasure></value_unitofmeasure>
                            <value></value>
                        </value_constraint>
                    </item>
                    . . .

        </panel>
    </filter_list>
    <patient_list min="1" max="10">
        <patient_num index="1">50</patient_num>
        <patient_num index="2">24</patient_num>
        <patient_num index="3">78</patient_num>
        <!--
<entire_patient_set/>
<patient_set_coll_id>0</patient_set_coll_id>
-->
```

```
    </patient_list>
</crc:request>
```

2.7.2.6 xsi:type=”crc:GetObservationFactByPrimaryKey_requestType”

```
<crc:request xsi:type="crc:observation_fact_set_requestType">
    </event_id>
    </patient_id>
    </concept_cd/>
    </observer_id/>
    </start_date/>
    </modifier_cd/>
</crc:request>
```

2.7.3 Responses

2.7.3.1 xsi:type=”crc:master_responseType”

```
<crc:response xsi:type="crc:master_responseType">
    <query_master>
        <query_master_id>0</query_master_id>
        <name/>
        <user_id/>
        <group_id/>
        <create_date>2000-12-30T00:00:00</create_date>
        <delete_date>2000-12-30T00:00:00</delete_date>
        <request_xml/>
        <generated_sql/>
    </query_master>
    <query_master>
        <query_master_id>1</query_master_id>
        <name/>
        <user_id/>
        <group_id/>
        <create_date>2000-12-30T00:00:00</create_date>
        <delete_date>2000-12-30T00:00:00</delete_date>
        <request_xml/>
```

```
        <generated_sql/>
    </query_master>
</crc:response>
```

2.7.3.2 xsi:type=”crc:instance_responseType”

```
<crc:response xsi:type="crc:instance_responseType">
    <query_instance>
        <query_instance_id>0</query_instance_id>
        <query_master_id>0</query_master_id>
        <user_id/>
        <group_id/>
        <batch_mode/>
        <start_date>2000-12-30T00:00:00</start_date>
        <end_date>2000-12-30T00:00:00</end_date>
        <query_status_type>
            <status_type_id>0</status_type_id>
            <name>finished</name>
            <description/>
        </query_status_type>
    </query_instance>
    <query_instance>
        <query_instance_id>1</query_instance_id>
        <query_master_id>0</query_master_id>
        <user_id/>
        <group_id/>
        <batch_mode/>
        <start_date>2000-12-30T00:00:00</start_date>
        <end_date>2000-12-30T00:00:00</end_date>
        <query_status_type>
            <status_type_id>0</status_type_id>
            <name>finished</name>
            <description/>
        </query_status_type>
    </query_instance>
</crc:response>
```

2.7.3.3 xsi:type=”crc:query_result_instanceTYPE”

```
<crc:response xsi:type="crc:result_responseType">
    <query_result_instance>
        <result_instance_id>0</result_instance_id>
        <query_instance_id>0</query_instance_id>
        <query_result_type>
            <result_type_id>0</result_type_id>
            <name>PATIENT_SET</name>
            <description/>
        </query_result_type>
        <set_size>0</set_size>
        <start_date>2000-12-30T00:00:00</start_date>
        <end_date>2000-12-30T00:00:00</end_date>
        <query_status_type>
            <status_type_id>0</status_type_id>
            <name>finished</name>
            <description/>
        </query_status_type>
    </query_result_instance>
    <query_result_instance>
        <result_instance_id>1</result_instance_id>
        <query_instance_id>0</query_instance_id>
        <query_result_type>
            <result_type_id>1</result_type_id>
            <name>ENCOUNTER_SET</name>
            <description/>
        </query_result_type>
        <set_size>0</set_size>
        <start_date>2000-12-30T00:00:00</start_date>
        <end_date>2000-12-30T00:00:00</end_date>
        <query_status_type>
            <status_type_id>0</status_type_id>
            <name>finished</name>
            <description/>
        </query_status_type>
    </query_result_instance>
</crc:response>
```

2.7.3.4 xsi:type=”crc:request_xml_responseType”

```
<crc:response xsi:type="crc:request_xml_responseType">
    <xml_string>
        <![CDATA[
```

```
            <query_definition>
                <query_name/>
                <query_description/>
                <query_timing>ANY</query_timing>
                <specificity_scale>0</specificity_scale>
                <query_date_from>2000-12-30T00:00:00</query_date_from>
                <query_date_to>2000-12-30T00:00:00</query_date_to>
                <panel>
                    <panel_number>0</panel_number>
                    <panel_date_from>2000-12-30T00:00:00</panel_date_from>
                    <panel_date_to>2000-12-30T00:00:00</panel_date_to>
                    <invert>0</invert>
                    <total_item_occurrences>0</total_item_occurrences>
                    <item>
                        <hlevel>0</hlevel>
                        <item_name/>
                        <item_table/>
                        <item_key/>
                        <item_icon/>
                        <tooltip/>
                        <class/>
                    </item>
                </panel>
            </query_definition>
                ]]>
        </xml_string>
    </crc:response>
```

2.7.3.5 xsi:type=”crc:master_instance_result_responseType”

```
<crc:response xsi:type="crc:master_instance_result_responseType">
    <query_master>
        <query_master_id>0</query_master_id>
        <name/>
        <user_id/>
        <group_id/>
        <create_date>2000-12-30T00:00:00</create_date>
        <delete_date>2000-12-30T00:00:00</delete_date>
        <request_xml/>
        <generated_sql/>
    </query_master>
    <query_instance>
```

```
    <query_instance_id>0</query_instance_id>
    <query_master_id>0</query_master_id>
    <user_id/>
    <group_id/>
    <batch_mode/>
    <start_date>2000-12-30T00:00:00</start_date>
    <end_date>2000-12-30T00:00:00</end_date>
    <query_status_type>
        <status_type_id>0</status_type_id>
        <name>finished</name>
        <description/>
    </query_status_type>
</query_instance>
<query_result_instance>
    <result_instance_id>0</result_instance_id>
    <query_instance_id>0</query_instance_id>
    <query_result_type>
        <result_type_id>0</result_type_id>
        <name>PATIENT_SET</name>
        <description/>
    </query_result_type>
    <set_size>0</set_size>
    <start_date>2000-12-30T00:00:00</start_date>
    <end_date>2000-12-30T00:00:00</end_date>
    <query_status_type>
        <status_type_id>0</status_type_id>
        <name>finished</name>
        <description/>
    </query_status_type>
</query_result_instance>
<query_result_instance>
    <result_instance_id>1</result_instance_id>
    <query_instance_id>0</query_instance_id>
    <query_result_type>
        <result_type_id>1</result_type_id>
        <name>ENCOUNTER_SET</name>
        <description/>
    </query_result_type>
    <set_size>0</set_size>
    <start_date>2000-12-30T00:00:00</start_date>
    <end_date>2000-12-30T00:00:00</end_date>
    <query_status_type>
        <status_type_id>0</status_type_id>
        <name>finished</name>
        <description/>
    </query_status_type>
```

```
    </query_result_instance>
</crc:response>
```

2.7.3.6 xsi:type=”crc:instance_result_responseType”

```
<crc:response xsi:type="crc:instance_result_responseType">
    <query_instance>
        <query_instance_id>0</query_instance_id>
        <query_master_id>0</query_master_id>
        <user_id/>
        <group_id/>
        <batch_mode/>
        <start_date>2000-12-30T00:00:00</start_date>
        <end_date>2000-12-30T00:00:00</end_date>
        <query_status_type>
            <status_type_id>0</status_type_id>
            <name>finished</name>
            <description/>
        </query_status_type>
    </query_instance>
    <query_result_instance>
        <result_instance_id>0</result_instance_id>
        <query_instance_id>0</query_instance_id>
        <query_result_type>
            <result_type_id>0</result_type_id>
            <name>PATIENT_SET</name>
            <description/>
        </query_result_type>
        <set_size>0</set_size>
        <start_date>2000-12-30T00:00:00</start_date>
        <end_date>2000-12-30T00:00:00</end_date>
        <query_status_type>
            <status_type_id>0</status_type_id>
            <name>finished</name>
            <description/>
        </query_status_type>
    </query_result_instance>
    <query_result_instance>
        <result_instance_id>1</result_instance_id>
        <query_instance_id>0</query_instance_id>
        <query_result_type>
            <result_type_id>1</result_type_id>
```

```
            <name>ENCOUNTER_SET</name>
            <description/>
        </query_result_type>
        <set_size>0</set_size>
        <start_date>2000-12-30T00:00:00</start_date>
        <end_date>2000-12-30T00:00:00</end_date>
        <query_status_type>
            <status_type_id>0</status_type_id>
            <name>finished</name>
            <description/>
        </query_status_type>
    </query_result_instance>
</crc:response>
```

2.7.3.7 xsi:type=”crc:patient_data_responseType”

```
<crc:response xsi:type="crc:patient_data_responseType">
    <crc:patient_data>
        <!-- see PDO cell messaging document -->
    </crc:patient_data>
</crc:response>
```

2.8 CRC XML Schema Definitions

The CRC XML schema consists of the following XSD files:

2.8.1 CRC.xsd

This schema is not used directly to create the CRC messages.

It is included in *CRC_PDO_QRY.xsd* and *CRC_PSM_QRY.xsd*.

2.8.2 CRC_PDO_QRY.xsd

This schema is used for validating CRC patient data queries.

It defines the <*crc:header*> and <*crc:sql*> tags.

2.8.3 CRC_PDO_QRY_request.xsd

This schema is not used directly.

It is included in *CRC_PDO_QRY.xsd*.

2.8.4 CRC_PDO_QRY_response.xsd

This schema is not used directly

It is included in *CRC_PDO_QRY.xsd*.

2.8.5 CRC_PSM_OBJ.xsd

This schema defines the data objects that hold information about patient set queries.

2.8.6 CRC_PSM_QRY.xsd

This schema is used for validating CRC patient set queries.

It defines the <*crc:header*> and <*crc:sql*> tags.

2.8.7 CRC_PSM_QRY_request.xsd

This schema is not used directly

It is included in *CRC_PSM_QRY.xsd*.

2.8.8 CRC_PSM_QRY_response.xsd

This schema is not used directly

It is included in *CRC_PSM_QRY.xsd*.

2.8.9 CRC_PSM_QRY_query_definition.xsd

This schema validates the xml format that defines a patient set query.

2.8.10 CRC_PSM_QRY_analysis_definition.xsd

This schema validates the XML format that defines an analysis request.

2.8.11 CRC_PANEL.xsd

This schema defines the panel definition.

2.8.12 CRC_UPLOADER_QRY.xsd

This schema defines the upload request and response message.

2.8.13 i2b2_result_msg.xsd

This schema defines the result format in name-value pairs.