Calyx RIM 7.0 Data Exchange

CALYX™

Contents

Calyx RIM Data Exchange 7.0 1

Prepare to use Calyx RIM Data Exchange for Importing Data 1

Assign the Data Exchange User in Calyx RIM 2

Use Data Exchange to Import Data into Calyx RIM 2

Data Supported by Data Exchange Import 2

Prepare Data for Import 4

Include Special Characters in Fields 5

Data Exchange Web Service 5

Import Data with the Data Exchange Web Service 6

Set Security to Access Import Functionality through the Web Service 7

Data Exchange ADD Message 9

Data Exchange UPDATE Message 11

Data Exchange UPSERT Message 14

Import Data with the Data Exchange API 15

Confirm and Modify Imported Data in Calyx RIM 16

Import Failures 16

Transaction Logs 16

Use Data Exchange to Export Data from Calyx RIM 17

Entities Supported by Data Exchange Export 18

Set Notifications in XML Format 18

Export Application Assumptions 20

Transaction Logs 20

Post a Message to the Data Exchange Web Service 22

UPSERT Message Components 23

Data Exchange API 25

Data Exchange API Document 25

Calyx RIM 7.0 Data Exchange Updates 25

Data Exchange 7.0 Metadoc Document 27

Index **28**

Calyx RIM Data Exchange 7.0

Calyx RIM Data Exchange 7.0 is designed to import certain entity information into the Calyx RIM database from multiple sources and to export certain entity information stored in Calyx RIM to other systems using Data Exchange Import and Export functionality.

The topics included in this guide describe how to import data into Calyx RIM, and export data, using the Calyx RIM Data Exchange 7.0 API.

Data Exchange Web Service extends the existing data exchange capabilities. This web service enables you to transmit data in the accepted data exchange message format directly from your system without the need for a custom data exchange client.

Data Exchange Export functionality is not currently supported by the Data Exchange Web Service.

Prepare to use Calyx RIM Data Exchange for Importing Data

Calyx RIM Data Exchange uses a programming interface, not a graphical user interface (GUI). To import records, an application uses the Calyx RIM Data Exchange Web service.
If your application is going to use the Data Exchange client that Calyx RIM Data Exchange provides for posting messages, see the examples in the `DataExchanger-examples.jar` file included with Calyx RIM.

If your application is going to use Message Orientated Middleware (MOM) software for posting messages, your application must construct the messages.

To modify data so that it conforms to the proper constraints for import, specific Calyx RIM privileges must be established for the user and certain rules must be applied:

- To enable the creation of entities, you must have Administrator privileges for the entities to be imported.
- Calyx RIM Data Exchange fields must be configured and active in Calyx RIM before importing data with Calyx RIM Data Exchange.
- All data entry rules and constraints concerning uniqueness, field lengths, and predefined values apply.
- All required fields must be populated.
- Before importing the data, values must exist for all drop-down fields defined in Data Administration.
- Date field limitations apply to newly created entities. The following format is required:

```
yyyy-MM-dd'T'HH:mm:ss.SSSXXX
```

For instructions on setting Calyx RIM user privileges in Data Administration, refer to the Data Administration topics in the online help.

Assign the Data Exchange User in Calyx RIM

To be aware of all transactions made using Calyx RIM Data Exchange, you must assign the Data Exchange User in Calyx RIM.

When the Calyx RIM Data Exchange license is activated in an Calyx RIM installation, the Calyx RIM Data Exchange user is automatically created in the system. The Calyx RIM Data Exchange user is assigned a unique identifier in Calyx RIM called data exchange that identifies all transactions made using Calyx RIM Data Exchange.

This Calyx RIM Data Exchange user ID appears in the Calyx RIM Data Exchange log file, `data_exchanger.log`, and also in the Calyx RIM audit trail to identify transactions that are performed using Calyx RIM Data Exchange.

Administrative privileges in Calyx RIM for the specified Calyx RIM Data Exchange user *(data exchange)* must be assigned during Calyx RIM installation. You will need Calyx RIM administrative privileges that include the ability to add or edit entity attributes.

Note: *Do not edit or delete the* data exchange *user or group permissions.* data exchange *is the user ID used by Calyx RIM and the Calyx RIM Data Exchange API for importing data.*

For instructions on assigning users in Calyx RIM, see the topic *Assign User or Group Security Privileges*.

Use Data Exchange to Import Data into Calyx RIM

Importing data into Calyx RIM involves the following activities:

— Preparing data for import
— Importing data using Calyx RIM Data Exchange
— Confirming and modifying imported data in Calyx RIM Data Exchange

Data Supported by Data Exchange Import

Data Administration Values

Calyx RIM Data Exchange provides the ability to import the following kinds of records into the Calyx RIM Data Exchange database.

- ATC
- ATC RMS Values
- Attachment
- Change Status
- Change Type
- Compendial Designation
- Compendial Designation/Source RMS Values
- Concentration Measure Type
- Concentration Measure Type RMS Values
- Concentration Unit
- Country
- Country RMS Values
- Custom List Name
- Custom List Value
- Denominator Unit
- Denominator/Numerator Unit RMS Values
- Device Allergenicity
- Device Nomenclature
- Device Sterility Indicator
- Device Type
- Device Usage
- Dosage/Pharmaceutical Form (including XEVMPD information)
- Dosage/Pharmaceutical Form RMS Values
- EMA Authorisation Status
- EMA Authorisation Status RMS Values
- Event-Change Detail Status
- Indications/Intended Use
- Leaf Status
- Language
- Language RMS Values
- Legal Basis
- Legal Basis RMS Values
- Legal Status
- Length Unit Of Measure
- MAH/Development Sponsor/Organization (including XEVMPD Information)
- Manufacturer Global Product Details for Active Ingredients, Excipients, Packaging, Material, Manufacturing Functions
- Manufacturer
- Master File Location (MFL)
- Medical Device
- Medical Device RMS Values
- Medical Device Part Code
- Medicinal Product Type
- Medicinal Product Type RMS Values
- Message Sender
- Numerator Unit
- Previous EV Code
- Procedure Type
- Procedure Type RMS Values
- Product Characteristics
- Product Material
- Product Strength Unit
- Quantity Unit of Measure
- Reason for Termination/Withdrawal
- Region
- Region Type
- Registration Status
- Route of Administration (including XEVMPD Information)
- Route of Administration RMS Values
- Shelf Life Type
- SME Status
- Special Measure
- Sterilization Requirement
- Substance - including Substance Alias, Substance Alias Translations, Substance International Codes, Substance Translations, Substance Attachments, Substance XEVMPD Information
- Substance Class
- Substance Type
- Trade Names
- Unit of Measure Prefix
- Unit of Measure Prefix RMS Values
- Unit Of Presentation
- Volume Unit of Measure
- Weight Unit of Measure

Entities

- Application
- CT Shared Data
- Change
- Change Details
- Event
- Event-Change Details(s)
- Event-Product(s)
- Event Country
- Event Country Status
- Full Product Presentation
- FPP Previous EV Code
- License Package Set Country
- Package Set Registration
- Product Detail Set (PDS)
- PDS Active Ingredient Component Detail
- PDS Country Detail
- PDS Flu Strain Detail
- PDS Indications/Intended Use Detail
- PDS Package Set Detail
- PDS Reference Strain Detail
- PDS Species Indication Detail
- PDS Substance Component Detail
- Product Component
- Product Family
- Product
- Pharmaceutical Product
- Product Component Active Ingredient
- Product Component Reference Active Ingredient
- Registration FPP Attachment
- Registration Attachment
- Sequence
- Task

Prepare Data for Import

When you create or update a sequence through Data Exchange, ensure the event name, if provided, is unique within an application. A duplicate event name will result in an error. When you associate a change detail to an event through Data Exchange, ensure that the change detail name is unique within an the system and the event name is unique within the application. A duplicate change detail name or duplicate event name will result in an error.

Note: *Comments and Comment threads are not supported for import by Calyx RIM Data Exchange.*

Before using Calyx RIM Data Exchange, perform the following tasks to prepare your data for import to Calyx RIM.

1. Remove duplicate data.
2. Export data from your corporate repositories and collect the data in a common location for import into Calyx RIM.
3. Format the data according to the schema that has been provided in the API.
   See the *Data Exchange API Document*.
4. Connect Calyx RIM Data Exchange to the database.

Include Special Characters in Fields

When you use Calyx RIM Data Exchange to add an entity that includes special characters in the name or code fields, use a CDATA block to add the special characters.

In the following example, an asterisk is added in a CDATA block in order to include the asterisk symbol (*) in a Product Family name or code.

```
<value><![CDATA[VKP *]]></value>
```

The following special characters can be used in entity name or code fields only if they are included within a CDATA block:

! @ # $ % ^ & * ( ) _ + | \ / < >

This example shows how all of the special characters can be included for use in the entity names or codes.

```
<value><![CDATA[VKPF !@#$%^&*()_+|\/<>}?{]]></value>
```

Note: *Although Calyx RIM Data Exchange can accept special characters when performing import functionality the Calyx RIM User Interface might not display data containing special characters as expected.* It is strongly recommended that special characters be removed from data before it is imported into Calyx RIM. *(For more information, see Special Characters in your Calyx RIM online help or user guide.)*

Data Exchange Web Service

Calyx RIM Data Exchange 7.0 includes the Data Exchange Web Service. The Data Exchange Web Service provides a convenient way for an application to import data into the Calyx RIM Data Exchange database.

An application can still import data by using the Calyx RIM Data Exchange client API to format messages and by using the Data Exchange Web Service to post the messages to a JMS queue. Posting to the queue directly requires `dataExchange-examples.jar` Implementation-Version: 1.0.0.141 or earlier and is not supported in Calyx RIM 7.0.

Benefits of Using the Data Exchange Web Service

Using the Data Exchange Web Service has the following benefits:

- RESTful Web Service - The Data Exchange functionality is exposed as a RESTful Web service. An application can use an external system such as Message-Orientated Middleware (MOM) to post messages to the Data Exchange Web Service.
- Exposed Message Format - To post a message to the web service, messages can be constructed and sent via the Data Exchange client API or use the provided grammar to construct the messages manually.
- HTTP/HTTPS - An application can use HTTP to send messages to the Data Exchange Web Service. Calyx RIM supports SSL, enabling an application to send messages by using HTTPS, which provides more security and control.

— Status Reporting - If an application posts messages to a JMS queue via a legacy Data Exchange client API, the application writes errors to a log file on the Calyx RIM server. If an application posts messages to the Data Exchange Web Service, the application writes log messages to a transaction ledger. The application can issue GET requests remotely to retrieve these messages.

Same Data Exchange Functionality

Calyx RIM Data Exchange import functionality is the same whether the application posts to the Data Exchange Web Service or to a JMS queue. The Data Exchange Web Service does not currently support Data Exchange export functionality.

Import Data with the Data Exchange Web Service

An application can import data by constructing a message and posting the message to the Calyx RIM Data Exchange Web Service on the Calyx RIM server. The application can get the status of the transaction from the Data Exchange Web Service.

Constructing a Message

For each class that Data Exchange supports, you can easily identify the fields required for each type of message used to import data. See the *Calyx RIM Data Exchange 7.0 Metadoc Document*.

Posting a Message to the Data Exchange Web Service

The Data Exchange Web Service consumes text/xml posted to the Calyx RIM 7.0 server on the following path of the Calyx RIM context:

```
rest/dataexchange/process
```

Example

In this example, Calyx RIM 7.0 is running on the server InSight6 on port 8080.

To send the message, you can use an application to post the message to the following path as text/xml:

```
http://InSight6:8080/InSight/rest/dataexchange/process
```

You receive a 200 OK response with a URL similar to:

```
http://InSight6:8080/InSight/rest/dataexchange/status/275003
```

Getting the Status of a Data Exchange Transaction from the Data Exchange Web Service

After you post a message to the Data Exchange Web Service as text/xml, you can paste the returned URL into a GET request. After you send this request, the Data Exchange Web Service returns the status of the transaction as application/xml on the following path of the InSight context:

```
rest/dataexchange/status/<transaction ID>
```

Example

In this example, you send a GET request containing the URL received in the example of posting a message:

```
http:// InSight6:8080/InSight/rest/dataexchange/status/275003
```

The Data Exchange Web Service returns the following transaction status message:

```
<dataExchangeStatusMessage>
        <received>2015-03-04T17:17:02.855-05:00</received>
        <processed>2015-03-04T17:17:02.918-05:00</processed>
        <parsemessages>
               <errors>
                      <errorText>Unable to convert value for

                       field: countryId on
                       object: ID1</errorText>
               </errors>
               <errors>
                      <errorText>Unable to convert value for

                       field: healthAuthorities on
                       object: D1</errorText>
               </errors>
               <errors>
                      <errorText>Health Authority is

                       required.</errorText>
               </errors>
        </parsemessages>
        <storemessages>
               <errors>
                      <errorText>Unable to ADD object with
                       id: ID1</errorText>
               </errors>
        </storemessages>
</dataExchangeStatusMessage>
```

Set Security to Access Import Functionality through the Web Service

Access to the Data Exchange import through the Web Service must be enabled by defining the IP addresses or host names in the `DataExchangeIPConfig.xml` file.

To define IP addresses or host names in the `DataExchangeConfig.xml` file, do the following.

1. Navigate to the `C:\InsightManager\server\all\conf\insight` folder and open the `DataExchangeConfig.xml` file in Notepad.
2. To add an IP range, locate the following section:

   Example

```
<constructor-arg index="0">
   <list>
    <!--<value>0.0.0.0-0.0.0.1</value>-->
      </list>
```

```
  </constructor-arg>
```

3. Enter an IP range between the value attribute.

   Example

```
For example, 123.123.123.25-123.123.123.50
```

4. Uncomment the code line by removing the comment tags (<!-- and -->).
5. To add an IP address, locate the following section:

   Example

```
        <constructor-arg index="1">
            <list>
                <!--<value>0.0.0.0</value>-->
                <!--<value>0.0.0.1</value>-->
               </list>
```

6. Enter an IP address between the value attributes.
   a) To add more IP addresses, enter IP addresses between the successive attributes.
7. Uncomment the code line by removing the comment tags (<!-- and -->).
8. To add a user host name, locate the following section:

   Example

```
<constructor-arg index="2">
            <list>
                <!--<value>my.domain.com</value>-->
                <!--<value>machine.domain.name.com</value>-->
            </list>
```

9. Enter a host name between the value attributes.
   a) To add more hosts, enter host names between the successive attributes.
10. Uncomment the code line by removing the comment tags (<!-- and -->).
11. To authenticate users with Data Exchange, proceed as follows:
    a) Locate the following section:

```
<!-- User authentication required to call data exchange -->
        <constructor-arg index="0" value="false"/>
```

    b) Set value to `true`.

```
<constructor-arg index="0" value="true"/>
```

    c) Add a user. Locate the following section:

```
<constructor-arg index="1">
            <list>
                <!--<value>user1</value>-->
            </list>
        </constructor-arg>
```

d) Enter a user between the value attributes.

e) Uncomment the code line by removing the comment tags (`<!--` and `-->`).

12. Save and close the `DataExchangeConfig.xml` file.

Data Exchange ADD Message

If your application imports data by using the Calyx RIM Data Exchange Web Service, the application constructs the messages that the applications posts to the Data Exchange Web Service.

This topic describes the ADD message.

ADD Message Example

```
<dataexchange version="3">
    <method type="ADD">
        <object id="ID1" type="com::liquent::InSight::manager::
                              regintel::domain::RICountryDetail">
            <field id="owner">
                <value>John Admin</value>
            </field>
            <field id="keywords">
            <value>keywords</value>
            </field>
            <field id="productTypes">
                <values>
                    <value>Pharmaceutical</value>
                    <value>Medical Device</value>
                </values>
            </field>
            <field id="description">
                <value>description</value>
            </field>
            <field id="countryId">
                <value>unit test country</value>
            </field>
            <field id="healthAuthorities">
                <values>
                    <value>authorities</value>
                </values>
            </field>
            <field id="comments">
                <value>comments</value>
            </field>
        </object>
    </method>
</dataexchange>
```

ADD Message Components

Data Exchange Header

```
<dataexchange version=”3”>
```

The recommended version is the current version. For Calyx RIM 7.0, the current version is version 3.

Method

```
<method type=”ADD”>
```

Must be ADD for all ADD messages.

Object

```
<object id="ID1" type="

com::liquent::InSight::manager::regintel::domain::RICountryDetail ">
```

id must be unique for all objects in the message. A message can include multiple objects.

type must reference a metaclass unique type.

Field

```
<field id=" productTypes ">
```

id must reference the field name in the appropriate metaclass defined by type in the object component.

Values

```
<values>
```

A <values> tag has multiple values for a field. The values are usually for a multi-select field in the user interface.

Value

```
 <value> Pharmaceutical </value>
```

If the referenced field is a drop-down list in the user interface UI, the value must be a valid value in the list. For free-text fields, normal validation rules apply.

ADD Message Grammar

```
Message: <Data Exchange header> Data Exchange Header: (<method>)+ Method:
(<object>)+ Object: (<field>)+ Field: <value> | <values> Values: (<value>)+
Value: text
```

Character	Description
+	Indicates one or more of the immediately preceding grouping.
|	Indicates the preceding or following (not both) term or grouping.
()	Indicates a grouping of the enclosed terms.
<>	Indicates that the enclosed text is a term.

Data Exchange UPDATE Message

If your application imports data by using the Calyx RIM Data Exchange Data Exchange Web Service, the application constructs the messages that the applications posts to the Data Exchange Web Service.

This topic describes the UPDATE message.

UPDATE Message Example

Note that this example is identical to the example in *Data Exchange UPSERT Message*, except for <method type=”UPDATE”> in the following example.

```
<?xml version="1.0" encoding="UTF-8"?>
<dataexchange version="3">
    <method type="UPDATE">
        <object id="updateID1" type="com::liquent::

                                InSight::manager::
                                registration::domain::

                                Application">
            <field id="name">
                <value>UpdatedTESTAPPNAME</value>
            </field>
            <field id="code">
                <value>UpdatedTESTAPPCODE</value>
            </field>
            <field id="clinicalTrialsNumber">
                <value>CTN #2</value>
            </field>
            <field id="procTypeId">
                <value>National US</value>
            </field>
            <field id="relatedSet">
                <value>RelatedApp2</value>
            </field>
            <field id="progCode">
                <value>UpdatedTESTAPPCODE</value>
            </field>
            <field id="arbitrationFlag">
                <value>No</value>
            </field>
            <field id="internalCode">
```

```
                <value>INTERNALCODE</value>
            </field>
            <field id="submittorId">
                <value>United States</value>
            </field>
            <field id="legalDistTypeId">
                <value>Prescription-Only</value>
            </field>
            <field id="legalBasisId">
                <value>New Legal Basis</value>
            </field>
            <field id="statusDate">
                <value>2011-12-03</value>
            </field>
            <field id="statusId">
                <value>Planned</value>
            </field>
            <field id="impid">
                <value>Some updated impi ID</value>
            </field>
            <field id="regionId">
                <value>United States</value>
            </field>
                <query id="query for updateID1" type="com::

                                                liquent::
                                                InSight::
                                                manager::
                                                registration::
                                                domain::
                                                Application">

            <field id="code">
                <value>SOMETESTAPPNAME</value>
            </field>
        </query>
    </object>
</method>
</dataexchange>
```

UPDATE Message Components

Data Exchange Header

```
<dataexchange version=”3”>
```

The recommended version is the current version. For Calyx RIM Data Exchange 7.0, the current version is version 3.

Method

```
<method type=”UPDATE”>
```

Must be UPDATE for all UPDATE operations.

Object

```
<object id="ID1"

type="com::liquent::InSight::manager::registration::domain::Application">
```

id must be unique for all objects in the message. A message can include multiple objects.

type must reference a metaclass unique type.

Field

```
<field id="statusId">
```

id must reference the field name in the appropriate metaclass defined by type in the object component.

Query

```
<query id="query for updateID1"
type="com::liquent::InSight::manager::registration::domain::Application">
```

id must be unique for all queries in the Data Exchange message. A message can include multiple queries.

type must reference a metaclass unique type that you are querying against.

Values

```
<values>
```

A <values> tag has multiple values for a field. The values are usually for a multi-select field in the user interface.

Value

```
<value>Planned</value>
```

If the referenced field is a drop-down list in the user interface UI, the value must be a valid value in the list. For free-text fields, normal validation rules apply.

UPDATE Message Grammar

```
Message: <Data Exchange header> Data Exchange Header: (<method>)+ Method:
(<object>)+ Object: (<field>)+ <query> Query: (<field>)+ Field: <value> |
<values> Values: (<value>)+ Value: text
```

Character	Description
+	Indicates one or more of the immediately preceding grouping.

Character	Description
|	Indicates the preceding or following (not both) term or grouping.
()	Indicates a grouping of the enclosed terms.
<>	Indicates that the enclosed text is a term.

Data Exchange UPSERT Message

If your application imports data by using the Calyx RIM Data Exchange Data Exchange Web Service, the application constructs the messages that the applications posts to the Data Exchange Web Service.

This topic describes the UPSERT message.

UPSERT Message Example

Note that this example is identical to the example in *Data Exchange UPDATE Message*, except for <method type=”UPSERT”> in the following example.

```
<?xml version="1.0" encoding="UTF-8"?>
<dataexchange version="3">
    <method type="UPSERT">
        <object id="upsertID1" type="com::liquent::
                                InSight::manager::
                                registration::domain::
                                Application">
            <field id="name">
                <value>UpdatedTESTAPPNAME</value>
            </field>
            <field id="code">
                <value>UpdatedTESTAPPCODE</value>
            </field>
            <field id="clinicalTrialsNumber">
                <value>CTN #2</value>
            </field>
            <field id="procTypeId">
                <value>National US</value>
            </field>
            <field id="relatedSet">
                <value>RelatedApp2</value>
            </field>
            <field id="progCode">
                <value>UpdatedTESTAPPCODE</value>
            </field>
            <field id="arbitrationFlag">
                <value>No</value>
            </field>
            <field id="internalCode">
                <value>INTERNALCODE</value>
            </field>
            <field id="submittorId">
                <value>United States</value>
```

```
            </field>
            <field id="legalDistTypeId">
                <value>Prescription-Only</value>
            </field>
            <field id="legalBasisId">
                <value>New Legal Basis</value>
            </field>
            <field id="statusDate">
                <value>2011-12-03</value>
            </field>
            <field id="statusId">
                <value>Planned</value>
            </field>
            <field id="impid">
                <value>an updated impi ID</value>
            </field>
            <field id="regionId">
                <value>United States</value>
            </field>
            <query id="query for upsertID1" type="com::
                                                liquent::
                                                InSight::
                                                manager::
                                                registration::
                                                domain::
                                                Application">
                <field id="code">
                    <value>test app name</value>
                </field>
            </query>
        </object>
    </method>
</dataexchange>
```

Import Data with the Data Exchange API

When the data in your system has been exported to a common location and is properly formatted according to the schema provided in the Calyx RIM Data Exchange API, it is ready to be imported into the Calyx RIM database. Calyx RIM Data Exchange requires *`read only`* access to the data to be imported.

For the following procedure, refer to the Calyx RIM Data Exchange API documentation and `APIUsageExample.java` for more detailed information and examples.

To import data:

1. Import the necessary Calyx RIM Data Exchange code.
2. Use the API to specify the action for the data: `add`, `update`, or `upsert`. On the first message, specify only one request.
3. Assign a unique ID to each entity.
4. Set the values for all fields on the entity by using the API.

5. Use the DataExchangeClient to send the data to Calyx RIM Data Exchange by posting a message to the Data Exchange Web Service.
   The Web service returns a transaction ID in a URL.
6. Use the returned transaction ID to check the import transaction status in the logs.

Confirm and Modify Imported Data in Calyx RIM

After your data is imported to the Calyx RIM database, you must confirm or add the necessary values in Calyx RIM. For information about viewing and modifying entities, and for a table describing entity attributes, please refer to the Calyx RIM user documentation at help.liquent.com.

Import Failures

After importing data to the Calyx RIM database using Data Exchange, there may be system messages that describe problems that may have prevented certain data from importing.

When there is a transaction failure for specific data, that data is not imported to the Calyx RIM database.

The following conditions will cause transaction failures during the import process:
— All required fields must be present or the transaction will fail.
— All required fields must have data that meets the data entry requirements for the field or the transaction will fail.
— Incorrect date formats will cause the transaction to fail.
— Improperly formatted data will cause the transaction to fail.

Import failures are logged in transaction logs.

Transaction Logs

All successful import and export transactions are logged, and failed transactions are logged with an error message that can be used to identify the error condition and remediate the problem.

Calyx RIM Data Exchange Log

Check import transaction statuses in the Calyx RIM Data Exchange Data Exchange log. Your program can identify Calyx RIM Data Exchange Data Exchange transactions by using the transaction IDs (URLs) that the Calyx RIM Data Exchange Data Exchange Web service returns after your program posts messages.

The Calyx RIM Data Exchange Data Exchange Web service periodically purges this log. The default period is every 24 hours, but the Calyx RIM Config wizard can configure this period. (The InSight Config wizard is used in the installation of Calyx RIM.)

Calyx RIM Data Exchange Data Exchange Log Examples

The following is an example of a log entry for an import that succeeded:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dataExchangeStatusMessage>
    <received>2014-12-15T17:33:19.851-05:00</received>
    <processed>2014-12-15T17:33:20.867-05:00</processed>
    <parsemessages>
```

```
        <successes>
            <successText>Successfully performed parse for ADD
of object with id: my family</successText>
        </successes>
    </parsemessages>
    <storemessages>
        <successes>
            <successText>Successfully performed store operation
for ADD of object with id: my family</successText>
        </successes>
    </storemessages>
</dataExchangeStatusMessage>
```

The following is an example of a log entry for an import that failed:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dataExchangeStatusMessage>
    <received>2014-12-15T17:34:41.634-05:00</received>
    <processed>2014-12-15T17:34:41.743-05:00</processed>
   <parsemessages>
        <successes>
            <successText>Successfully performed parse for ADD
of object with id: my family</successText>
        </successes>
    </parsemessages>
    <storemessages>
        <errors>
            <errorText>Family Code, Family Name must be unique
in the system.</errorText>
        </errors>
    </storemessages>
</dataExchangeStatusMessage>
```

InSight Audit Trail

Calyx RIM audit trail queries result in detailed information about the database actions that occurred during an Calyx RIM Data Exchange import or export.

Use Data Exchange to Export Data from Calyx RIM

Calyx RIM Data Exchange has a mechanism to define notification specifications at the xml level without the need to use the Calyx RIM notification wizard.

Define InSight Data Exchange Export Criteria

The definitions are created in draagon meta instance xml notation. The `EcanNotificationService` reads in the (optional) file specified by the parameter name `NotificationsFile` on the `EcanNotificationService` object in the `framework.xml` file at application start-up. By default, this value is set to `${ENV.InSight-props}/notifications.xml`.

Currently, only update and new record notifications are supported. This is the only way to define notifications that are sent using the Calyx RIM Data Exchange export feature.

When a Calyx RIM Data Exchange export is triggered, a subdirectory of the `InSight-props` directory is created called `DataExchangeExport`. A new file is created for each sent notification that is named as follows:

If the notification was created with a name, it is used. If no name exists, the xmlIdentifier value is used instead. In both cases, the current date and time in the format `yyyy-MM-dd_hh_mm_ss` using the server time zone is appended.

Entities Supported by Data Exchange Export

You can export data for the following entities that can have associated notifications using the Calyx RIM Data Exchange Export function:

— Application
— Change
— Change Detail
— CT Shared Data
— Event
— Event Country
— Package Set Registration
— PDS Country
— Product Family
— Product
— Product Component
— Project
— Reference
— Registration
— Sequence
— Task

Set Notifications in XML Format

The following code provides an example of how to create a notification that is issued when an entity is created.

Note that any object-based element (denoted by a '`type`' attribute) must use an '`id`' attribute value that is guaranteed to be unique within the file.

The `<value>` of the '`xmlIdentifier`' field must refer to a unique section of `<exportcriteria>` xml.

If no `<exportcriteria>` exists with the specified name, the default behavior is to export every field on the notified entity. In some cases, this may provide unexpected results, because the default meta-view is used to convert the field values to strings.

```
<notifications>
```

```
 <object id="createFamilyExample"
type="com::liquent::InSight::manager::ecan::domain::EcanNotificationSpec">
  <field id="entityTypeId">
    <value>PRODFAM</value>
  </field> <field id="notificationTypeId">
    <value>NEWRECORD</value>
  </field>
  <field id="notificationLevel">
    <value>O</value>
  </field>
  <field id="notificationName">
    <value>XML Based trigger on product family creation</value>

  </field>
  <field id="deliveryMechanism">
    <value id="exportScenario1"
type="com::liquent::InSight::manager::ecan::domain::DEExportDeliveryMechani
sm">
      <field id="xmlIdentifier">
        <value>exportCriteriaScenario1</value>
      </field>
    </value>
  </field>
</object>
<exportcriteria id="exportCriteriaScenario1">
 <exportobject id="exportScenarioLicense"
type="com::liquent::InSight::manager::registration::domain::License">

  <!- -export these fields from the Registration that triggered the
export-->
  <fieldGroup>
   <fieldRef name="effectiveDate"/>
   <fieldRef name="code"/>
   <fieldRef name="authorityId"/>
   <fieldRef name="marketingAuthority"/>
  </fieldGroup>
  <grid name="packageSets">
   <!- -export all data from the 'packageSets' 'grid' (tab in
                    Calyx RIM UI) tied to the Registration that
triggered the export- ->
   <fieldGroup>
    <fieldRef name="productDetailSetId" exportObject="true">
     <!- -export these fields from the PDS associated to this particular
'packageSet' from the grid tied to the Registration that triggered the
export-->
     <fieldGroup>
      <fieldRef name="code"/>
     </fieldGroup>
    </fieldRef>
    <!- -export this field from this particular 'packageSet' from the grid
tied to the Registration that triggered the export- ->
    <fieldRef name="productId"/>
```

```
   </fieldGroup>
  </grid>
 </exportobject>
 <exportobject id="exportScenarioApplication"
type="com::liquent::InSightjames
may ::manager::registration::domain::Application">
  <!- -export these fields from the Application associated to the
Registration that triggered the export- ->
  <fieldGroup>
   <fieldRef name="appTypeId" />
   <fieldRef name="procTypeId"/>
   <fieldRef name="code"/>
   <fieldRef name="regionId" view="myFavoriteView"/>
  </fieldGroup>
 </exportobject>
</exportcriteria>
</notifications>
```

Export Application Assumptions

Calyx RIM currently makes no effort to clean up the InSight-props directory, so it is assumed that individuals who receive these messages account for this when finished with them.

Transaction Logs

All successful import and export transactions are logged, and failed transactions are logged with an error message that can be used to identify the error condition and remediate the problem.

Calyx RIM Data Exchange Log

Check import transaction statuses in the Calyx RIM Data Exchange Data Exchange log. Your program can identify Calyx RIM Data Exchange Data Exchange transactions by using the transaction IDs (URLs) that the Calyx RIM Data Exchange Data Exchange Web service returns after your program posts messages.

The Calyx RIM Data Exchange Data Exchange Web service periodically purges this log. The default period is every 24 hours, but the Calyx RIM Config wizard can configure this period. (The InSight Config wizard is used in the installation of Calyx RIM.)

Calyx RIM Data Exchange Data Exchange Log Examples

The following is an example of a log entry for an import that succeeded:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dataExchangeStatusMessage>
    <received>2014-12-15T17:33:19.851-05:00</received>
    <processed>2014-12-15T17:33:20.867-05:00</processed>
    <parsemessages>
        <successes>
            <successText>Successfully performed parse for ADD
of object with id: my family</successText>
        </successes>
    </parsemessages>
    <storemessages>
```

```
        <successes>
            <successText>Successfully performed store operation
for ADD of object with id: my family</successText>
        </successes>
    </storemessages>
</dataExchangeStatusMessage>
```

The following is an example of a log entry for an import that failed:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dataExchangeStatusMessage>
    <received>2014-12-15T17:34:41.634-05:00</received>
    <processed>2014-12-15T17:34:41.743-05:00</processed>
   <parsemessages>
        <successes>
            <successText>Successfully performed parse for ADD
of object with id: my family</successText>
        </successes>
    </parsemessages>
    <storemessages>
        <errors>
            <errorText>Family Code, Family Name must be unique
in the system.</errorText>
        </errors>
    </storemessages>
</dataExchangeStatusMessage>
```

InSight Audit Trail

Calyx RIM audit trail queries result in detailed information about the database actions that occurred during an Calyx RIM Data Exchange import or export.

Post a Message to the Data Exchange Web Service

The Data Exchange Web Service consumes text/xml posted to the Calyx RIM 7.0 server on the following path of the Calyx RIM context: `rest/dataexchange/process`

In this example, Calyx RIM 7.0 is running on the server InSight6 on port 8080.

To send the message, you can use an application to post the message to the following path as text/xml:

```
http://InSight6:8080/InSight/rest/dataexchange/process
```

You receive a `200 OK` response with a URL similar to:

```
http://InSight6:8080/InSight/rest/dataexchange/status/275003
```

UPSERT Message Components

Data Exchange Header

```
<dataexchange version=”3”>
```

The recommended version is the current version. For Calyx RIM Data Exchange 6.0, the current version is version 3.

Method

```
<method type=”UPSERT”>
```

Must be UPSERT for all UPSERT operations.

Object

```
<object id="ID1"
type="com::liquent::InSight::manager::registration::domain::Application">
```

id must be unique for all objects in the message. A message can include multiple objects.

type must reference a metaclass unique type.

Field

```
<field id="statusId">
```

id must reference the field name in the appropriate metaclass defined by type in the object component.

Query

```
<query id="query for updateID1"
type="com::liquent::InSight::manager::registration::domain::Application">
```

idmust be unique for all queries in the Data Exchange message. A message can include multiple queries.

type must reference a metaclass unique type that you are querying against.

Values

```
<values>
```

A <values> tag has multiple values for a field. The values are usually for a multi-select field in the user interface.

Value

```
<value>Planned</value>
```

If the referenced field is a drop-down list in the user interface UI, the value must be a valid value in the list. For free-text fields, normal validation rules apply.

Data Exchange API

Data Exchange API Document

Title
Data Exchange 7.0 API Document

Calyx RIM 7.0 Data Exchange Updates

This table describes the Data Exchange updates for Calyx RIM 7.0.

Data Exchange API updates:

- Application has moved to the top entity in its own hierarchy.
- The Family Id is removed. Application Id is used now independently.
- Application Code value is added to the Application Id field. (Format: "app code" + space + "app name"). Example: ApplicationId in queryFor is now (appCode1234 appName1234)
- The uniqueness criteria for the Application has changed. (Uniqueness for API is used in "/create-or-update" controller operation).

Entity Name	DE 6.2	DE 7.0 Updates	Comment
Application	- Family Id is required - selectedProducts field is not required	- Family Id is removed - selectedProducts field is required	To search for a specific Application associated with the current entity, some of the entities under the Application consisted of (familyId + applicationId) fields and used the following values for the applicationId: - family Id - PF name value - application Id - App name value After the removal of family Id, Application Id is now used independently.

Entity Name	DE 6.2	DE 7.0 Updates	Comment
Event	– **Family Id** is required – Searching/querying applicationId by "appName"	– **Family Id** is removed – Searching/ querying applicationId by using concatenated fields separated by space. Example: "appCode appName"	N/A
EventCountry	– **Family Id** is required – Searching/querying applicationId by "appName"	– **Family Id** is removed – Searching/ querying applicationId by using concatenated fields separated by space. Example: "appCode appName"	
EventCountrySchedule	– **Family Id** is required – Searching/querying applicationId by "appName"	– **Family Id** is removed – Searching/ querying applicationId by using concatenated fields separated by space. Example: "appCode appName"	
AppEventProduct	– **Family Id** is required – Searching/querying applicationId by "appName"	– **Family Id** is removed – Searching/ querying applicationId by using concatenated fields separated by space. Example: "appCode	

Data Exchange 7.0 Metadoc Document

Title
Data Exchange 7.0 Metadoc Document

Index

A

API 25

D

Data 22
Data Admin 7
Data Exchange 1, 2, 4–6, 9, 11, 14–18, 20, 23, 25, 27