

Version 4.4 - Design Document for RESTful API Support within the SDK

Version No: 0.2

Last Modified: 07/09/2010

Author : Santhosh Garmilla

Team : caCORE Software Development Kit (SDK)

Client : National Cancer Institute - Center for Bioinformatics,

National Institutes of Health,

US Department of Health and Human Services

**Document History**

Document Location

The most current version of this document is located in SVN under cacoresdk/projects/docs/SDK 4.4 docs/Development/Design.

Revision History

| **Version Number** | **Revision Date** | **Author** | **Summary of Changes** |
| --- | --- | --- | --- |
| 0.1 | 07/02/10 | Santhosh Garmilla | Initial Draft |
| 0.2 | 07/09/10 | Santhosh Garmilla | Updated TOC |
|  |  |  |  |

Review

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Team/Role | Version | Date Reviewed | Reviewer Comments |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Related Documents

More information can be found in the following related documents and websites:

|  |
| --- |
| Document Name |
| SVN: cacoresdk/projects/docs/SDK 4.4 docs/Development/Design/ SDK\_4.4\_ISO\_Data\_Type\_Support\_within\_Web\_UI\_Design\_Document.docx |
| Wiki: Supported ISO and non-ISO data types: https://wiki.nci.nih.gov/display/caCORE/7+Example+UML+Model+and+Mapping |
|  |
|  |
|  |

**Table of Contents**

[1. Introduction 4](#_Toc266696766)

[1.1 Scope 4](#_Toc266696767)

[2. Requirement Analysis 4](#_Toc266696768)

[2.1 Requirements 5](#_Toc266696769)

[2.1.1 Business Requirements 5](#_Toc266696770)

[3. Detail Design 6](#_Toc266696771)

[3.1 Overview 6](#_Toc266696772)

[3.2 Use case Scenarios 8](#_Toc266696773)

[3.2.1 Querying a Simple ISO Data Type 8](#_Toc266696774)

[3.2.2 Querying a Complex Data Type 8](#_Toc266696775)

[3.2.3 Querying a Complex ISO Data Type with Simple and Complex Attributes 9](#_Toc266696776)

[3.2.4 Querying a Collection of Complex Data Types 9](#_Toc266696777)

[3.3 Class Diagram 10](#_Toc266696778)

[3.4 Sequence Diagram 12](#_Toc266696779)

[4. Unit Testing 12](#_Toc266696780)

[4.1 JUnit Test Cases 12](#_Toc266696781)

[4.2 Test Case Scenarios 12](#_Toc266696782)

# Introduction

* ISO 21090 data types is an International Standard that:
* Provides a set of data type definitions for representing and exchanging basic concepts that are commonly encountered in healthcare environments in support of information exchange in the healthcare environment;
* Specifies a collection of healthcare related data types suitable for use in a number of health related information environments;
* Declares the semantics of these data types using the terminology, notations and data types defined in ISO 11404 rev 2005;
* Provides UML definitions of the same data types using the terminology, notation and types defined in Unified Modeling Language (UML) version 2.0;
* Defines an eXtensible Markup Language (XML) based representation of the data types suitable for use when exchanging information between information processing entities.
* The effort to support ISO 21090 data types within the caCORE SDK began with SDK v4.3, and continues with SDK v4.4.

## Scope

This document focuses on detailing the design and effort related to implementing RESTful services for ISO 21090 data type support within the SDK generated application.

# Requirement Analysis

The following diagram summarizes the identified formal and non-formal requirements related to the implementation of RESTful Queries for ISO 21090 data types within a SDK generated application



## Requirements

### Business Requirements

1. Support for Conversion of RESTful Query into ISO Java object

SDK 4.3 won’t support the conversion of RESTful Query ISO data types such as AD, EN, and DSET into ISO Java Objects. SDK system-web library must be updated to support this feature.

1. Support for Conversion of ISO Java Object into HQL query

SDK system-core library must be updated to support this feature.

1. Support for GETHTML and GETJSON

As of SDK 4.3, the web application UI generation and display are unconditionally disabled whenever support for ISO data types is enabled. SDK system-web library must be updated for XSLT conversion of RESTful XML response into respective HTML or JSON outputs.

1. JUnit Test cases to validate the output response from RESTful Queries and Query By Example (QBE).

# Detail Design

## Overview

The Representational State Transfer (REST) interface provided by the SDK is a simple URL interface that transmits domain-specific data over HTTP without an additional messaging layer, such as SOAP, or session tracking via HTTP cookies.

The URL used by the REST interface adheres to the following pattern:

|  |  |
| --- | --- |
| REST Interface URL Pattern | http://<server\_name><server\_port>/<project\_name>/ <REST\_type>?query=<target>&<criteria>[&rolename=<rolename>] |

The following table describes each of the parameters of the REST URL:

| Parameter | Description |
| --- | --- |
| server\_name | A string identifying the server or host name.  **For example:** localhost or 127.0.0.1. |
| server\_port | A string identifying the port number to which the SDK server is listening. **For example:** 80 or 8080 |
| Project\_name | A string identifying the project name used for building and deploying the SDK application. **For example:** myproject or example.  **NOTE:** This value coincides with the PROJECT\_NAME property found within the codegen.properties file. |
| REST\_type | Either GetXML for XML output or GetJSON for JavaScript Object Notation output |
| Target | A string identifying the qualified or non-qualified query target/result class name.  **For example:** gov.nih.nci.cacoresdk.domain.inheritance.childwithassociation.Bank |
| Criteria | A string identifying the qualified or non-qualified criteria class name to be used as a filter/constraint on the result set.  **For example:** The SDK sample model Credit class has an association to the Bank class via its issuingBank attribute. If desired, the value of the id attribute of the criteria class instance can also be supplied in order to further constrain the result set. The pattern for such a criteria string is:  *<criteria\_class\_name>[@<attribute\_name>=<attribute\_value>]*  So that the criteria entry of *Credit[@id=3]* indicates that only target/result class instances that are associated to the Credit record with an id value of 3 are to be returned. |
| Rolename | The name of the attribute within the criteria class that identifies the association to be traversed when retrieving the target/result class(es). One example is the issuingBank attribute of the Credit class found within the sample SDK model.  The rolename property must be specified whenever the Criteria class has two or more attributes representing associations to the same target/result class type.  **For example:** The Child class within the sample SDK model contains two attributes: *mother* and *father*. Both represent instances of the Parent class. In this scenario, specifying a value of rolename=mother or rolename=father within the REST URL would ensure that the correct Parent instance is returned. |

Table 1‑1 Descriptions of the parameters of the REST URL

The following table provides some example URLs taken from the sample SDK model.

|  |  |
| --- | --- |
| Sample XML REST URL | [http://localhost:8080/example/GetXML?query=Bank&Credit[@id=3]&roleName=issuingBank](http://localhost:8080/example/GetXML?query=Bank&Credit%5b@id=3%5d&roleName=issuingBank) |
| Sample JSON REST URL | [http://localhost:8080/example/GetJSON?query=Bank&Credit[@id=3]&roleName=issuingBank](http://localhost:8080/example/GetJSON?query=Bank&Credit%5b@id=3%5d&roleName=issuingBank) |

Table 1‑2 Example URLs from the sample SDK model

While such a URL can be invoked directly through a browser, it is more frequently invoked programmatically via a remote client program. An example of such a program, *TestGetXMLClient.java*, is provided in the following folder, created by the SDK Code Generator:

*\target\dist\exploded\output\example\conf\system-template\package\remote-client\src*

[*Figure 1-1*](#Figure_1_1) below shows the XML output produced from invoking the *Sample XML REST URL* shown above for the sample ISO21090 SDK model.

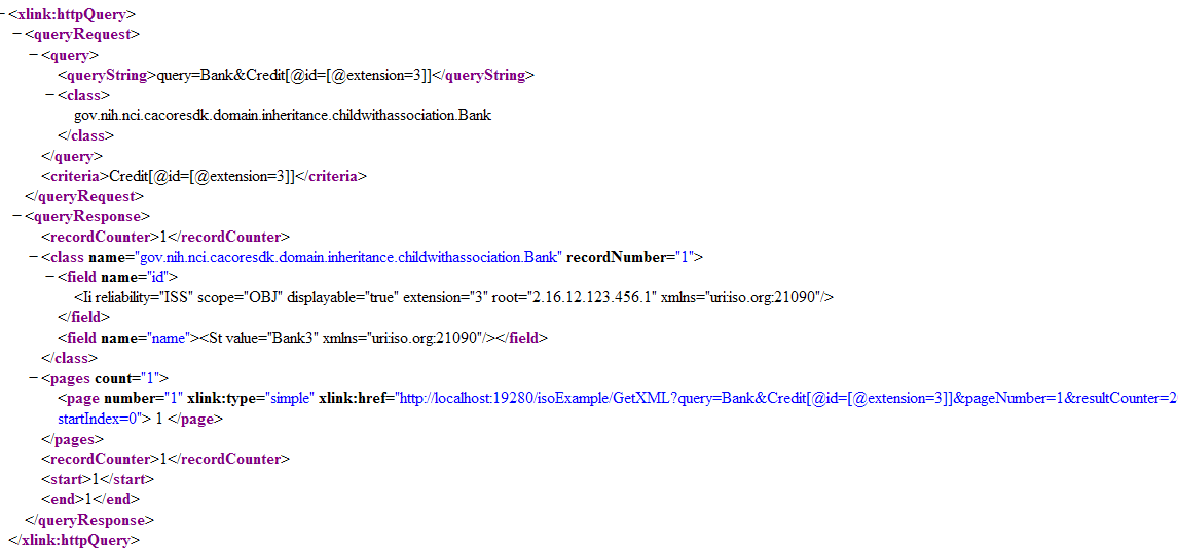


Figure 1-1 Sample XML output from REST call

## Use case Scenarios

The RESTful interface now supports querying domain data of ISO 21090 data types. The basic syntax of the RESTful query URL remains the same as in the previous release except that ISO 21090 data types adds support for simple data types, complex data types, complex data types with simple and complex attributes, complex data types with collection attributes, collection of complex data types, and, finally, complex collection of complex data types with collection attributes. Knowledge of ISO 21090 data types will help in constructing a query for the RESTful interface. The following sections provide different examples in querying ISO 21090 data types via the RESTful thin-client interface.

### Querying a Simple ISO Data Type

As of SDK v4.3, the SDK supports the querying of simple ISO data types via the RESTful interface.

As an example, BL is a simple ISO 21090 data type with a simple attribute named *value* of type *Boolean*. The following example demonstrates how to format a RESTful query of a logical model object that has a *BL* data type attribute:

*http://localhost:8080/example/GetXML?query=BLDataType&BLDataType[@value1=[@value=true]]*

This query will return all objects of class *BLDataType* (a sample class from the iso-example-project model) that contain a *value1* attribute of ISO 21090 data type *BL* that has an attribute *value* equal to *true*.

### Querying a Complex Data Type

As of SDK v4.3, the SDK also supports the querying of complex ISO data types via the RESTful interface.

As an example, *II* is a complex ISO 21090 data type containing a complex attribute named *extension*. The following example demonstrates how to format a RESTful query of a logical model object that has an *II* data type attribute:

*http://localhost:8080/example/GetXML?query=Deck,Suit&Card[@id=[@extension=3]]*

This query will return all objects of type Deck with an association from an object of type Suit that in turn has an association from an object of type Card containing an *id* attribute of an ISO 21090 data type *II* that in turn has an attribute *extension* equal to *3*.

The following query is an example of using multiple attributes in the criteria. This query will return all objects of type *CdDataType* where the attribute *value4* is of CD ISO data type, and the CD attributes *code* and *codeSystem* equal *CODE8* and *SYS1, respectively*:

*http://localhost:8080/example/GetXML?query=CdDataType&CdDataType[@value4=[@code=CODE8][@codeSystem=SYS1]]*

### Querying a Complex ISO Data Type with Simple and Complex Attributes

As of SDK v4.3, the SDK also supports the querying of complex ISO data types with simple and complex attributes via the RESTful interface.

As an example, CD is a complex ISO 21090 data type with both simple and complex attributes. The following example demonstrates how to format a RESTful query of a logical model object that has a CD data type attribute:

*http://localhost:8080/example/GetXML?query=CdDataType&CdDataType[@value1=[@code=CODE1]]*

This query will return all objects of type *CdDataType* (a sample class from the iso-example-project model) containing a *value1* attribute of an ISO 21090 data type CD that in turn has a simple attribute *code* equal to *CODE1*.

As a second example, the following query will return all objects of type *CdDataType* containing an attribute of an ISO 21090 data type CD that in turn has a complex attribute *originalText* of ISO 21090 data type *ED.TEXT* with an attribute *value* equal to *CDText*.

*http://localhost:8080/example/GetXML?query=CdDataType&CdDataType[@value4=[@originalText=[@value=CDText]]]*

### Querying a Collection of Complex Data Types

Finally, as of SDK v4.3, the SDK also supports the querying of a collection of complex ISO data types via the RESTful interface.

As an example, DSET<TEL> is a collection of complex ISO data type TEL. The following example demonstrates how to format a RESTful query of a logical model object that has a *DSET<TEL>* data type attribute:

*http://localhost:8080/example/GetXML?query=DsetTelDataType&DsetTelDataType[@value1=[@item=[@value=tel://123-456-7891]]]*

This query will return all objects of type *DsetTelDataType* (a sample class from the iso-example-project model) containing a *value1* attribute of an ISO 21090 data type *DSET<TEL>* that is a collection of *TEL* data type with attribute *value* equal to [*tel://123-456-7891*](tel://123-456-7891).

## Class Diagram



Figure: 1-5 RESTful API Components Class Diagram

## Sequence Diagram



Figure:1-6 RESTful Api Sequence Diagram

[Figure 1-6](#Figure_1_6) shows the Sequence Diagram for RESTful Api. When User/Application issues RESTful Query using SDK Remote Client, doGet method of HttpQuery get’s invoked. It intern delegates the request to HttpUtils. HttpUtils uses SDK SearchUtils Api to create Query by Example (QBE) Object. ORMDAOImpl converts QBE object into HQL query using utility class NestedCriteria2HQL.ORMDAOImpl makes a database call using resultant HQL query and returns the result set to the HttpQuery. HttpQuery does further processing to convert the result set into respective XML, JSON or HTML outputs.

# Unit Testing

## JUnit Test Cases

JUnit Test Case *HttpUtilsTest* and *SearchUtilsTest* validate all the use case scenarios for RESTful queries. They can be executed from Eclipse or from command line using build script.

## Test Case Scenarios

The test case scenarios will be developed in conjunction with the QA Team. Based on the initial design, the overall test scenarios are as mentioned below. Note that based on data each of these scenarios can have multiple test cases.

* Test the Search Criteria input screen for each of the supported ISO data types
  + Validate that each attribute for a given ISO data type contains the correct collection of sub-attributes for the data type
* Test the Search Criteria validation scripts
  + Validate that if a Null Flavor is selected for a given ISO data type attribute, the remaining fields that compose the attribute are disabled
* Test the Search Criteria query
* Test the Search Results