

caCORE SDK

RESTful API Design Document

Version No: 0.1

Last Modified: 07/02/10

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

Revision History

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

Review

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

Table of Contents

[1. Introduction 4](#_Toc266177683)

[2. Technical Requirements 4](#_Toc266177684)

[3. Overview 5](#_Toc266177686)

[4. Accessing Data from a Thin Client via the RESTful Interface 8](#_Toc266177687)

[5. Implementation 11](#_Toc266177688)

Design for RESTful Service

# Introduction

The purpose of this document is to understand Requirements, Implementation and Design details about RESTful Services for caCore SDK 4.4 release.

# Technical Requirements

# Restful API support for ISO Data types

* + AD
  + EN
  + DSET<CD>
* **Support for GETHTML and GETJSON queries**
  + XSLT for HTML
  + JSON for XSLT
* **Test Cases for Restful Api’s and QueryByExample**

# 

# 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 1-3*](#Figure_1_3) below shows the XML output produced from invoking the *Sample XML REST URL* shown above for the sample SDK model.

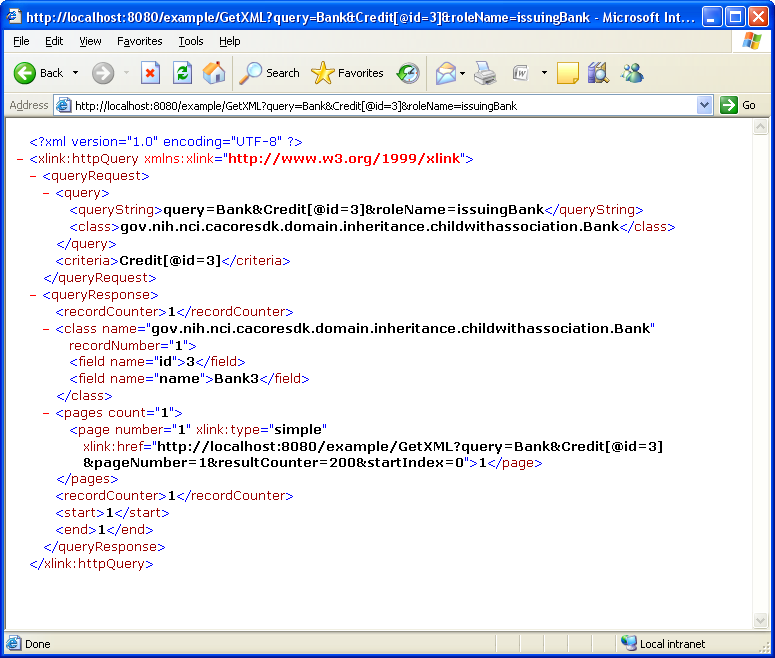


Figure 1-3 Sample XML output from REST call

[*Figure 1-4*](#Figure_1_4)below shows the JSON output produced from invoking the *Sample JSON REST URL* shown above for the sample SDK model.

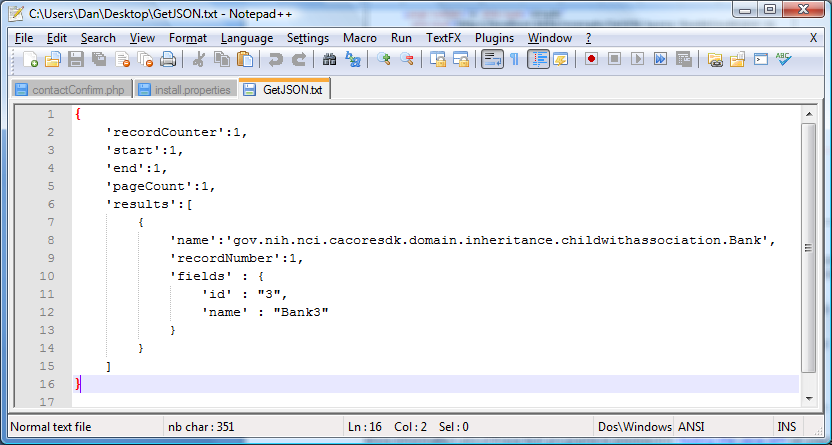


Figure 1-4 JSON output produced by Sample JSON REST URL from the sample SDK model

# Accessing Data from a Thin Client via the RESTful Interface

#### **Using the Target Parameter**

The *Target* parameter can contain the path to the Target class in addition to the Target class itself. Similar to the Nested Criteria interface, the path is a comma-separated list of classes, starting with the Target class and ending with the Criteria class.

For example:

http://localhost:8080/example/GetXML?query=Grandparent,Parent&Child[@id=3]

This query will return all objects of type Grandparent with an association from an object of type Parent that in turn has an association from an object of type Child containing an attribute id equal to 3.

#### **Using the Criteria Parameter**

The *Criteria* parameter can be a created in such a way to include restrictions on associated classes, as well as on the attributes of a Criteria class. The pattern for creating such criteria is one or more CriteriaObjects delimited by a / (forward slash).

Each CriteriaObject consists of the Class Name and zero or more attribute-value pairs, each enclosed in a set of [ ] (square brackets), with attribute name prefixed by an @ (at) sign.

For example:

http://localhost:8080/example/GetXML?query=Grandparent&Parent[@name=ParentName]/Child[@id=3]

Because the Parent has a direct association to the Grandparent, the path does not need to include Parent and Child, but only the target, Grandparent.

The CriteriaObject created by the above query restricts the list to only Parent objects that have the attribute name = parentName and that are associated with a Child class containing the attribute id = 3.

*Criteria* with association between domain objects can also be created using [] (square brackets). Left bracket is used as a delimiter to identity association. This pattern more useful in the case of complex data types or querying on multiple attributes of multiple types.

For example:

http://localhost:8080/example/GetXML?query=Card&Suit[@id=1][Deck[@id=1]]

This query will return all Card objects with an association with Suit with id equal to 1, that intern has association with Deck with id equal to 1.

#### **Accessing ISO 21090 data types via the RESTful Interface**

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).

# Implementation

## 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 call to database 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.