caCORE Software Developer Kit (SDK)

Version 4.4 - Design Document for ISO Data Type Support within the SDK Generated Web Application UI

|  |  |  |
| --- | --- | --- |
|  | Center for Biomedical Informatics and Information Technology |  |
|  |
|  |

This is a U.S. Government work. July 1, 2010

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 | 07/05/2010 | Daniel Dumitru | Initial Draft |
| 0.2 | 07/06/2010 | Daniel Dumitru | Added Dynamic View Section |
| 1.0 | 07/07/2010 | Daniel Dumitru | Reformat to Design Document format |
|  |  |  |  |

Review

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Team/Role | Version | Date Reviewed | Reviewer Comments |
| Vijay Parmar | Team Lead | 1.0 | 07/06/2010 | Initial Review |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

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/RESTfulAPI\_Design.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](#_Toc269886899)

[1.1 Problem Scenario 4](#_Toc269886900)

[1.2 Scope 4](#_Toc269886901)

[2. Requirement Analysis 5](#_Toc269886902)

[2.1 Requirements 5](#_Toc269886903)

[2.1.1 Business Requirements 5](#_Toc269886904)

[2.1.2 Technical Requirements 6](#_Toc269886905)

[2.2 Use Cases 6](#_Toc269886906)

[2.3 Assumptions 7](#_Toc269886907)

[2.4 Dependencies 8](#_Toc269886908)

[2.5 Known Issues or Future Considerations 8](#_Toc269886909)

[2.6 Technical Environment 8](#_Toc269886910)

[3. Detail Design 9](#_Toc269886911)

[3.1 Overview 9](#_Toc269886912)

[3.1.1 Web Application UI Search Workflow 9](#_Toc269886913)

[3.2 Workflows 10](#_Toc269886914)

[3.2.1 Workflow Diagram Highlighting Components and Processes that Require Changes 10](#_Toc269886915)

[3.3 Code Generation Component Changes 12](#_Toc269886916)

[3.3.1 Build-reconfigure Script Changes 12](#_Toc269886917)

[3.3.2 caCORE Workbench Code Settings Tab Changes 12](#_Toc269886918)

[3.4 Criteria Action Component Changes 12](#_Toc269886919)

[3.4.1 Criteria JSP Component 15](#_Toc269886920)

[3.4.2 ClassCache Component 15](#_Toc269886921)

[3.4.3 HtmlUtils Component 15](#_Toc269886922)

[3.5 Criteria Input Validation Component Changes 16](#_Toc269886923)

[3.5.1 Criteria Input Validation Script Component 16](#_Toc269886924)

[3.6 Result Action Component Changes 17](#_Toc269886925)

[3.6.1 Result.java Component 17](#_Toc269886926)

[3.7 Database Changes 18](#_Toc269886927)

[4. Unit Testing 19](#_Toc269886928)

[4.1 JUnit Test Cases 19](#_Toc269886929)

[4.2 Test Case Scenarios 19](#_Toc269886930)

# 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 eXtenshible 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.

## Problem Scenario

Due to time constraints, support for ISO 21090 data types was not completed in SDK v4.3 for the following areas:

* Generated Web Application UI
* Web Service API
* RESTful HTML and JSON API’s
* Writable API
* CLM (Logging) API
* Castor Mapping
* Permissible Values Validator
* Hibernate Validator
* WSSD Generation
* Instance and Attribute Level Security

One of the primary goals for SDK v4.4 is to implement support for ISO data types within the SDK generated web application User Interface (UI).

## Scope

This document focuses on detailing the design and effort related to integrating ISO 21090 data type support within the SDK generated web application user-interface (UI).

# Requirement Analysis

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



Each requirement is detailed below.

## Requirements

### Business Requirements

The identified high-level business requirements are:

1. **Provide means to enable Web Application UI when ISO data type support is enabled**

As of SDK 4.3, the web application UI generation and display are unconditionally disabled whenever support for ISO data types is enabled. This restriction needs to be removed so that the end-user can conditionally specify at code generation time whether or not the web application UI should be accessible when the generated application is deployed to a tomcat or JBoss application server instance.

1. **Provide means to specify search criteria for ISO data type domain model attributes**

In order to support ISO 21090 data types, the SDK web application UI will need to be enhanced to be able to display ISO 21090 data type attribute values that may be complex and / or composite in nature, containing various values for a single attribute.

1. **Provide means to validate search criteria field entries prior to submission to server**

This is a system requirement that needs to ensure, for instance, that email input fields only contain valid email entries.

1. **Provide means to view ISO data type attributes on the web application search Results page**

As of SDK v4.3, the results page returned from a search query only supports the display of a single value for a given attribute of a row within the result set.   
  
In order to support ISO 21090 data types, the SDK web application UI will need to be enhanced to be able to display search results that contain ISO 21090 data type attribute values that are complex and / or composite in nature, containing various values for a single attribute.

### Technical Requirements

The following technical requirements need to be met in support of the identified business requirements:

1. **Modify the caCORE Workbench CodegenSettingsPanel.java file to remove API enabling restrictions**
2. **Modify build-reconfigure.xml script to remove API enabling restrictions when ISO data type support is enabled**
3. **Create helper class which is responsible for generating Search Criteria HTML for each ISO data type**
4. **Modify Criteria JSP to call new helper class whenever Search Criteria HTML needs to be generated for a given ISO data type**
5. **Modify ClassCache.java to indicate that ISO data type fields are searchable**
6. **Create client-side java script to validate search criteria field entries prior to submission to server**
7. **Integrate submitted ISO data type criteria with existing search functionality**

In order to avoid redundancy, details for each technical requirement are provided in the Detail Design section below.

## Use Cases

The following use case diagram identifies the actors that play a role in enabling support for and / or in using ISO 21090 data types within a SDK generated Web Application UI:



## Assumptions

1. The RESTful XML and JSON API’s are also being modified in SDK v4.4 to support ISO data types. As part of this separate effort, the XSLT Transformer that generates the Results page is being modified. The web application UI is dependent upon the same transformer. The assumption is that The XSLT Transformer responsible for generating the search Results page will be completed accurately and in a timely fashion.
2. The effort to include support for ISO data types at the web application level assumes that the SDK module that creates and executes the database query to retrieve ISO data type rows from a given database is already functioning correctly.

## Dependencies

1. The RESTful XML and JSON API’s are also being modified in SDK v4.4 to support ISO data types. As part of this separate effort, the XSLT Transformer that generates the Results page is being modified. The web application UI is dependent upon the same transformer.

## Known Issues or Future Considerations

1. Initial discussions were held regarding creating cross-project, reusable client-side UI widgets to represent each of the ISO data types. However, due to time constraints and the existing SDK search criteria UI generation architecture, the decision was made to incorporate server-side HTML-generation logic for each ISO data. This discussion may be revisited in a future release, and the decision made to move the ISO data type HTML-generation logic to client-side, reusable widgets.

## Technical Environment

* Client Interface
  + Internet Explorer 6.0 and above
  + Mozilla v1.5.0.3 and above
* Application Server
  + Apache Tomcat 5.5.20
  + JBoss 4.0.5
* Operating system
  + Windows XP
  + Linux/Unix
* Java 1.5.0\_11 or higher, but not Java 1.6

# Detail Design

This section details the modifications needed to the existing SDK web application UI to incorporate support for ISO data types.

## Overview

### Web Application UI Search Workflow

The following activity diagram models the overall behavior of the web application UI system, and the way in which these behaviors are related in an overall flow of the system based upon end-user interactions.

Figure : Web Application UI Search Workflow Diagram



In this hybrid activity diagram, the grey rectangular boxes represent primary system components that participate in the search workflow process.

The general steps involved in the process of searching within the SDK web application UI include:

1. User selects a class node in the domain class browser, which causes the client-side dojo.io.bind treeNodeSelected(nodeId) function to be invoked
2. The dojo.io.bind component creates and invokes an AJAX call to the server-side Criteria action component
3. The server-side Criteria action component coordinates the processes for generating the Search Criteria HTML input form, and returns the generated HTML
4. The client-side dojo.io.bind component receives the Search Criteria HTML data and updates the UI
5. The user enters the desired search criteria, and clicks the submit button
6. The client-side system validates the input data for any validation errors
   1. If validation errors exist, the user is prompted to correct them
   2. If validation errors do not exist, the Search Criteria form data is submitted to the Result action component on the server
7. The server-side Result action component receives the Search Criteria form data, and coordinates the following sub-processes:
   1. Formulate and invoke the database query
   2. Receive the database query results
   3. Invoke the system XSLT component with the database query results data in order to generate the Results HTML page
   4. Formulate the Reponse information and send it back to the client-system
8. The client-system receives the Response information and displays the Results HTML in a new window

## Workflows

### Workflow Diagram Highlighting Components and Processes that Require Changes

The following activity diagram highlights the components and processes that will need to be modified to support ISO data types at the web application UI level:

Figure : Workflow Diagram highlighting components and processes that will need to be changed in order to support ISO data types at the web application UI Level



There are four primary components that will need to be changed in order to support ISO data types at the web application UI level:

* Code Generation Component (not shown)
* Criteria Action Component
* Validation Component
* Result Action Component

The first component, Code Generation, will also need to be changed. However, it is not shown in the above workflow, as it is responsible for generating the system, and thus is no longer active once the system has been generated and deployed.

The following section details the changes required to each of the above-mentioned components.

## Code Generation Component Changes

Code Generation components are responsible for implementing the technical requirements related to the following high-level business requirement:

* **Provide means to enable Web Application UI when ISO data type support is enabled**

The Code Generation components that need to be changed include:

* build-reconfigure.xml script
* caCORE Workbench CodeSettingsPanel.java

Changes required to each component are detailed below.

### Build-reconfigure Script Changes

Restrictions were introduced in SDK v4.3 to the build-reconfigure.xml script to unconditionally disable various interfaces whenever ISO data type support is enabled. These restrictions need to be removed so that the decision to enable or disable a given interfaces is returned to the end-user.

### caCORE Workbench Code Settings Tab Changes

As of caCORE SDK 4.3 and caCORE Workbench 0.1.2, the web application interface is unconditionally disabled on the Codegen Settings tab whenever ISO data type support is enabled. Modify the logic so that the end-user can specify whether or not the Web interface should be enabled regardless of the value of the "Enable ISO Data Type Support" property.

## Criteria Action Component Changes

Criteria action components are responsible for implementing the technical requirements related to the following high-level business requirement:

* **Provide means to specify search criteria for ISO data type domain model attributes**

As of SDK 4.3, end-users are only able to specify search criteria for non-ISO data type attributes, which include primitive Java data types summarized below[[1]](#footnote-1):

|  |  |  |  |
| --- | --- | --- | --- |
| Supported non-ISO Attribute Data Types | | | |
|  | Primary Key | Simple Data type | Collection data type |
| String | Yes | Yes | Yes |
| Integer | Yes | Yes | Yes |
| Double | Yes | Yes | Yes |
| Boolean | No | Yes | Yes |
| Float | Yes | Yes | Yes |
| Short | Yes | Yes | Yes |
| Long | Yes | Yes | Yes |
| Byte | Yes | Yes | Yes |
| Character | Yes | Yes | Yes |
| Date | Not Supported by SDK | Yes | Not Supported by SDK |
| String (CLOB) | Not Supported by SDK | Yes | Not Supported by SDK |

Figure : Supported non-ISO attribute data types

At the web application UI level, this restriction translates into a dynamic search criteria screen that only supports display of simple (primitive) data types and collections of the same, as illustrated below:

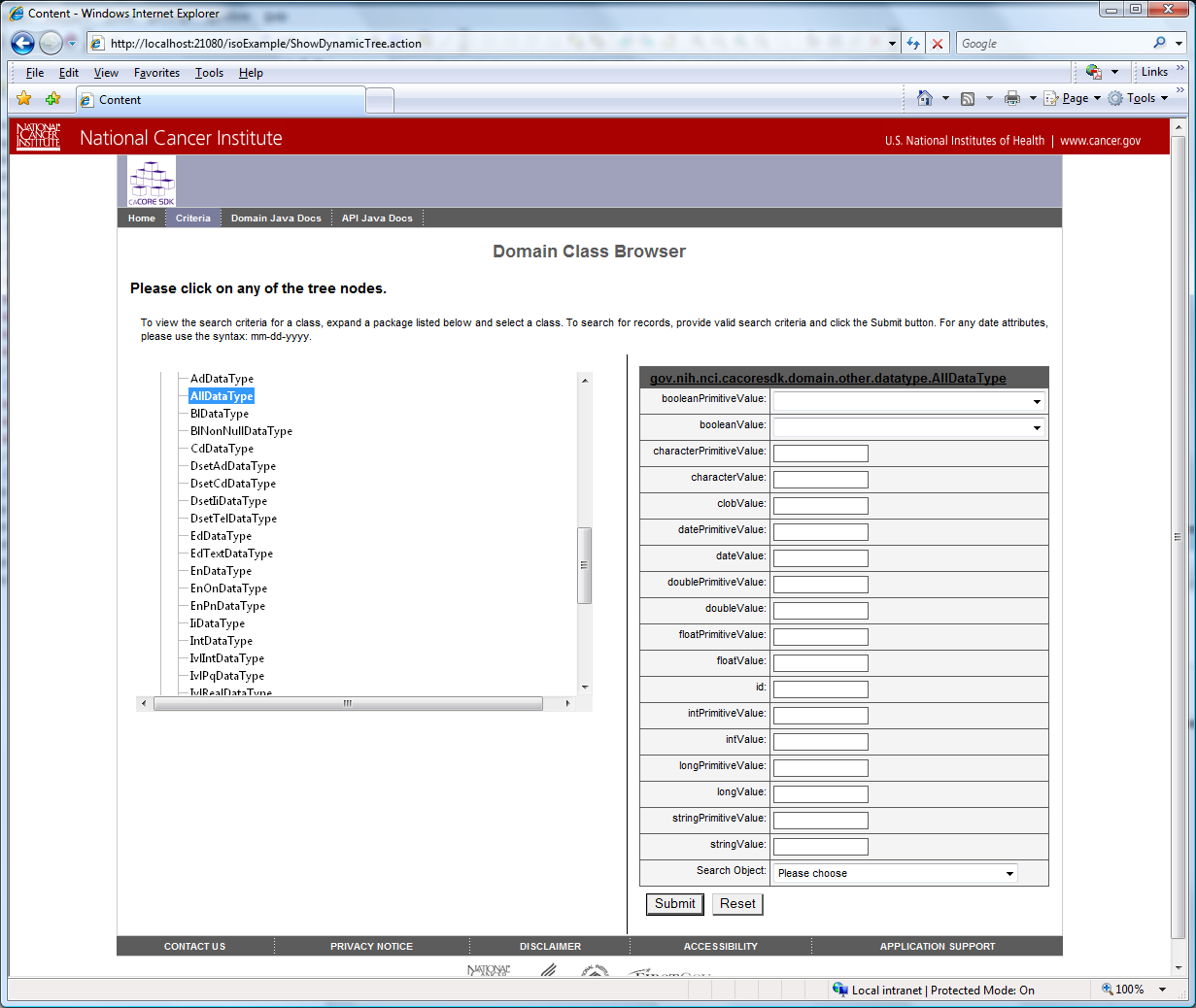


Figure : Existing Search Criteria input screen showing support for primitive data types

As shown above, the Search Criteria input form on the right-hand side only supports input of simple data types, where a given attribute value is specified via a single drop-down or text entry field.

In order to support ISO 21090 data types, the SDK web application UI will need to be enhanced to be able to display ISO 21090 data type attribute values that may be complex and / or composite in nature, containing various values for a single attribute, as shown below:

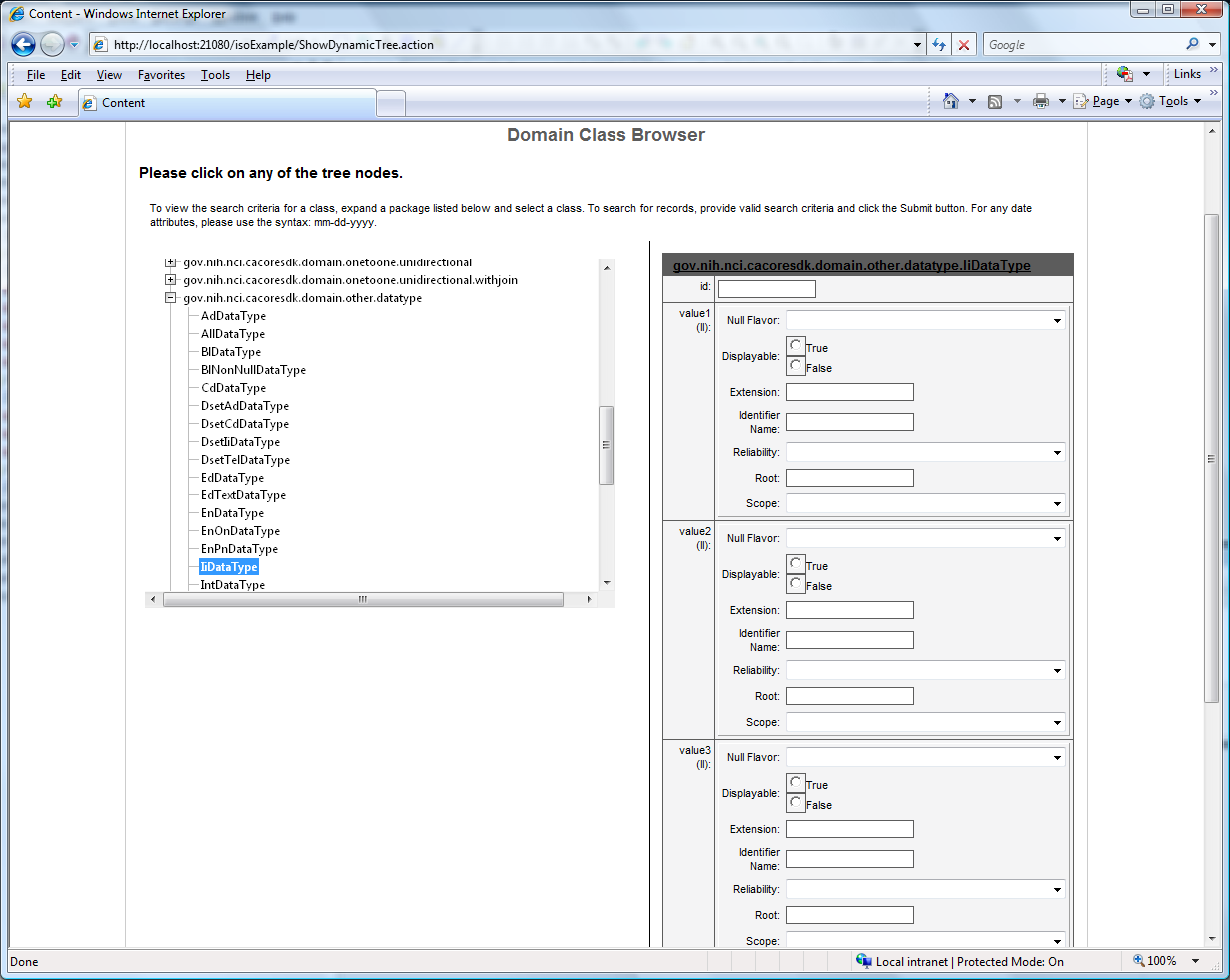


Figure : Sample Search Criteria input form changes showing various values for a single ISO data type attribute

This figure shows a sample of what the search criteria page needs to look like for, as an example, the sample class *IiDataType* of the *iso-project* sample model. This class contains various attributes, *value1* through *value6* of ISO data type *II*. The *II* ISO data type is a complex composite object with six (6) attributes: *displayable*, *extension*, *identifier name*, *reliability*, *root*, and *scope*, as shown in the figure below enlarged for improved readability:

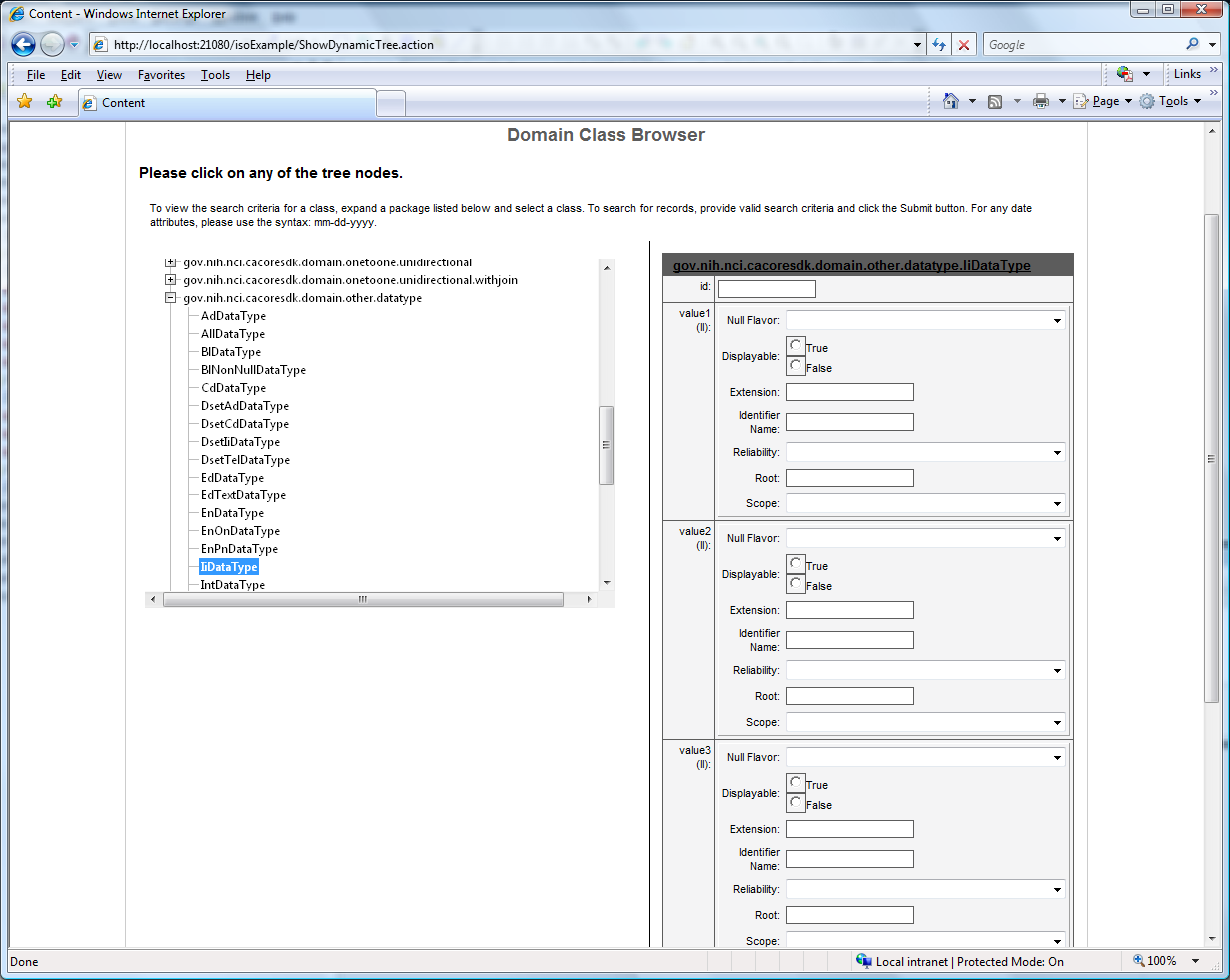


Figure : Sample II ISO data type search criteria input form enlarged to show details

The Criteria action components responsible for generating the Search Criteria HTML input form include:

* Criteria.jsp
* ClassCache.java
* HtmlUtils.java

Changes required to each component are detailed below.

### Criteria JSP Component

**Technical Requirement: Modify Criteria JSP to call new helper class whenever Search Criteria HTML needs to be generated for a given ISO data type**

The Criteria JSP component is responsible for coordinating the Search Criteria HTML generation process. It will need to be modified to be able to recognize and generate HTML input fields for each ISO data type. It will accomplish this task by calling a new helper class, HtmlUtils, which is responsible for actually generating the HTML corresponding to a given ISO data type.

### ClassCache Component

**Technical Requirement: Modify ClassCache.java to indicate that ISO data type fields are searchable**

Only "searchable" fields are displayed in the Search Criteria input form. *ClassCache* has the information and responsibility for determining at run time which class attributes meet the criteria for being considered searchable. By default, ISO data type attributes are not considered searchable, and so do not show up in the Search Criteria input form. *ClassCache* needs to be modified accordingly to ensure that ISO data type class attributes are considered searchable for SDK v4.4.

### HtmlUtils Component

**Technical Requirement: Create helper class, HtmlUtils, which is responsible for generating Search Criteria HTML for each ISO data type**

The server-side logic for generating search criteria form HTML for each ISO data type should be re-factored into a single helper class, instead of being incorporated in the search criteria JSP page. This will allow for easier, modular movement of the HTML-generation logic to client-side JavaScript if ISO data type UI widgets are implemented at a later time.

Also, as ISO data types can be complex and / or complex, HtmlUtils must create period ‘.’ delimited *name* and *id* HTML input attributes to indicate the nested level of the attribute. As an example, see highlighted *name* and *id* input attributes in the HTML snippet below for sample class CdDataType *value4* attribute:

<table border="1" cellpadding="3" cellspacing="0" width="100%">

<tbody>

<tr>

<td class="isoFormLabel">Null Flavor:</td>

<td class="isoFormField">

<select name="value4.nullFlavor" id="value4.nullFlavor" class="formFieldSized">

<option value=""></option>

<option value="NI">NI (No Information)</option>

<option value="INV">INV (Invalid)</option>

...

</select>

</td>

</tr>

<tr>

<td class="isoFormLabel">Original Text (ED):</td>

<td class="isoFormField">

<table border="1" cellpadding="3" cellspacing="0" width="100%">

<tbody>

<tr>

<td class="isoFormLabel">Null Flavor:</td>

<td class="isoFormField">

<select name="value4.originalText.nullFlavor" id="value4.originalText.nullFlavor" class="formFieldSized">

<option value=""></option>

<option value="NI">NI (No Information)</option>

<option value="INV">INV (Invalid)</option>

...

</select>

</td>

</tr>

<tr>

<td class="isoFormLabel">Value:</td>

<td class="isoFormField">

<input name="value4.originalText.value" id="value4.originalText.value" class="formFieldSized" type="text">

</td>

</tr>

</tbody>

</table>

</td>

</tr>

...

</tbody>

</table>

## Criteria Input Validation Component Changes

The Search Criteria Input Validation component is responsible for implementing the technical requirements related to the following high-level business requirement:

**Business Requirement: Provide means to validate search criteria field entries prior to submission to server**

### Criteria Input Validation Script Component

**Technical Requirement: Create helper class, HtmlUtils, which is responsible for generating Search Criteria HTML for each ISO data type**

As of SDK v4.3, the client-side system does not validate any of the search criteria entries prior to submission to the server. With the introduction of support for ISO data types, the requirement to validate entries prior to submission becomes even more important. ISO data type attributes can be complex in nature, with multiple fields composing a single attribute. If a given field, such as Null Flavor, is populated for an ISO data type attribute, other input fields will need to be disabled, as they are mutually exclusive.

The Search Criteria Input Validation script does not exist, and will need to be created.

## Result Action Component Changes

The Result Action component is responsible for implementing the technical requirements related to the following high-level business requirement:

**Business Requirement: Provide means to view ISO data type attributes on the web application Results page**

The Result action components responsible for supporting the requirement to view ISO data type attributes on the Results page include:

* Result.java

Changes required to each component are detailed below.

### Result.java Component

The Result.java component is responsible for converting the submitted Search Criteria parameters into a GetHTML RESTful query, and submitting the query back to the application server via a re-direct call.

Changes needed to the Result.java component include determining if a given Search Criteria parameter represents an ISO data type. If it does, the *key* portion of the parameter must be tokenized at the ‘.’ delimiter to identify the distinct levels.

## Database Changes

The database scripts for the SDK sample models (both standard and ISO versions) were modified in SDK v4.3. Minor changes are necessary for the standard version of the sample model for SDK v4.4 in order to properly map TEACHER nullFlavor column to the database. No additional changes need to be made for the current effort to enable ISO data type support at the web application UI level.

# Unit Testing

## JUnit Test Cases

This solution requires integration with a hosted web application that need to be deployed and be up and running. Consequently automated HTTPUnits/JUnit will not be available.

Integration tests will be performed manually and provided for various flows and exceptions paths.

## 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
  + For any given ISO data type attribute, validate that generated UI screen reflects the hibernate metadata, only displaying input fields defined in the metadata
* Test the Search Criteria validation JavaScripts. Validate that non-String attributes enforce type related validation. String types do not have any validation constraints:
  + Integer and Long data types should contain only an optional minus sign, followed by digits
  + Float and double data types should contain only an optional minus sign, followed by digits, followed by an optional decimal point, followed by digits.
  + Character data type attributes should only contain a single alphanumeric character including the underscore.
  + PQV precision attribute should only contain a positive integer
  + TEL value attribute should start with the mailto:, tel:, x-text-tel:, x-text-fax: URI scheme, or a valid URL.
  + TEL PERSON value attribute should start with the mailto:, tel:, x-text-tel:, or x-text-fax: URI scheme, followed by a valid email address or phone number.
  + TEL PHONE value attribute should start with the tel:, x-text-tel:, or x-text-fax: URI scheme, followed by a valid phone number.
  + TEL EMAIL value attribute should start with "mailto:" URI scheme, followed by a valid email address.
  + TS timestamp value attribute should follow the pattern: yyyy-mm-dd hh:mm:ss or mm-dd-yyyy hh:mm:ss
* Test the Search Criteria query
  + Validate that Request Post parameters are properly submitted and received at the server
  + Validate that the Request Post parameters are correctly translated into a corresponding GetHTML RESTful API query
  + Validate that the GetHTML RESTful API query gets properly translated into a corresponding hibernate HQL query
* Test the Search Results
  + Validate that the expected HTML results from the HQL query are returned and properly displayed in a new window

1. For more information, please refer to https://wiki.nci.nih.gov/display/caCORE/7+Example+UML+Model+and+Mapping. [↑](#footnote-ref-1)