# Sprint Integration Test of ISO Data Type Support within the SDK UI, RESTful API’s and Query Engine Components

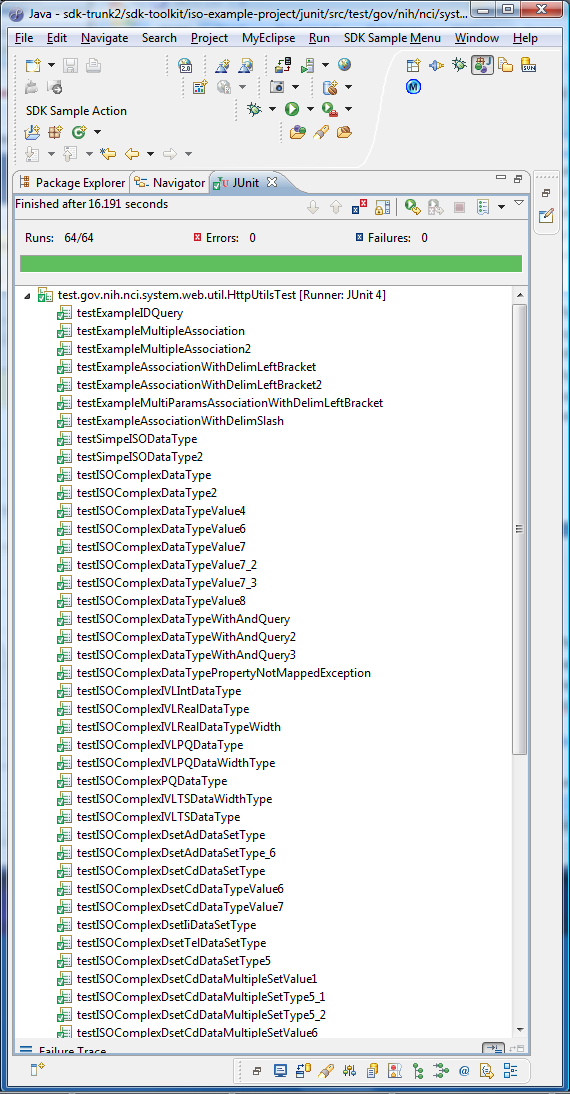
## List of ISO Data Types and Their Attributes Supported with SDK v4.4.

Support for the following ISO data types have been added to SDK v4.4 at the UI and query engine levels. Highlighted fields are supported within ISO data type library v2.1.1, but *not* within the SDK v4.4:

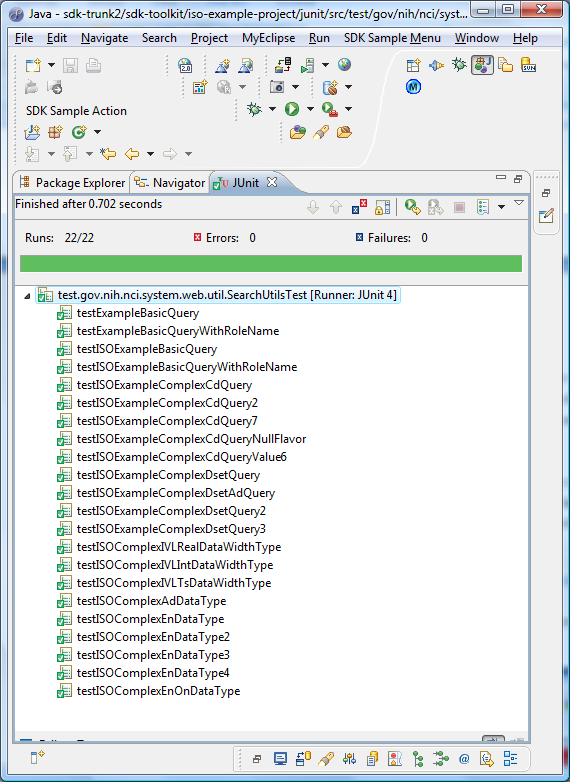
* AD (Address)
  + NullFlavor nullFlavor;
  + List<Adxp> part;
  + Set<PostalAddressUse> use;
  + QSet<Ts> useablePeriod;
* Adxp (Address Part)
  + NullFlavor nullFlavor;
  + String code;
  + String codeSystem;
  + String value;
  + AddressPartType type;
* BL
  + NullFlavor nullFlavor;
  + Boolean value;
* BLNONNULL
  + Boolean value;
* CD
  + NullFlavor nullFlavor;
  + String code;
  + String codeSystem;
  + String codeSystemName;
  + String codeSystemVersion;
  + St displayName;
  + EdText originalText;
  + String valueSet;
  + String valueSetVersion;
  + Set<Cd> translations;
* DSET<AD>
  + NullFlavor nullFlavor;
  + Set<AD> item
* DSET<CD>
  + NullFlavor nullFlavor;
  + Set< CD > item
* DSET<II>
  + NullFlavor nullFlavor;
  + Set< II > item
* DSET<TEL>
  + NullFlavor nullFlavor;
  + Set< TEL > item
* ED
  + NullFlavor nullFlavor;
  + Compression compression;
  + byte[] data;
  + String value;
  + String mediaType;
* EDTEXT
  + NullFlavor nullFlavor;
  + String value;
  + String mediaType;
* EN
  + NullFlavor nullFlavor;
  + List<Enxp> part;
  + Set<EntityNameUse> use;
* ENON
  + NullFlavor nullFlavor;
  + List<Enxp> part;
  + Set<EntityNameUse> use;
* ENPN
  + NullFlavor nullFlavor;
  + List<Enxp> part;
  + Set<EntityNameUse> use;
* Enxp
  + NullFlavor nullFlavor;
  + String code;
  + String codeSystem;
  + String codeSystemVersion;
  + Set<EntityNamePartQualifier> qualifier;
  + EntityNamePartType type;
  + String value;
* II
  + NullFlavor nullFlavor;
  + Boolean displayable;
  + String extension;
  + String identifierName;
  + IdentifierReliability reliability;
  + String root;
  + IdentifierScope scope;
* INT
  + NullFlavor nullFlavor;
  + Ed expression;
  + EdText originalText;
  + Qty uncertainty;
  + UncertaintyType uncertaintyType;
  + Integer value
* IVL<INT>
  + NullFlavor nullFlavor;
  + INT any;
  + INT high;
  + Boolean highClosed;
  + INT low;
  + Boolean lowClosed;
  + Qty width;
* IVL<PQ>
  + NullFlavor nullFlavor;
  + Ed expression;
  + EdText originalText;
  + Qty uncertainty;
  + UncertaintyType uncertaintyType;
  + PQ any;
  + PQ high;
  + Boolean highClosed;
  + PQ low;
  + Boolean lowClosed;
  + Qty width;
* IVL<REAL>
  + NullFlavor nullFlavor;
  + Ed expression;
  + EdText originalText;
  + Qty uncertainty;
  + UncertaintyType uncertaintyType;
  + REAL any;
  + REAL high;
  + Boolean highClosed;
  + REAL low;
  + Boolean lowClosed;
  + Qty width;
* IVL<TS>
  + NullFlavor nullFlavor;
  + Ed expression;
  + EdText originalText;
  + Qty uncertainty;
  + UncertaintyType uncertaintyType;
  + TS any;
  + TS high;
  + Boolean highClosed;
  + TS low;
  + Boolean lowClosed;
  + Qty width;
* PQ
  + NullFlavor nullFlavor;
  + Ed expression;
  + EdText originalText;
  + Qty uncertainty;
  + UncertaintyType uncertaintyType;
  + String unit;
  + Set<Pqr> translation;
  + Double value;
  + Integer precision;
* REAL
  + NullFlavor nullFlavor;
  + Ed expression;
  + EdText originalText;
  + Qty uncertainty;
  + UncertaintyType uncertaintyType;
  + Double value;
* SC
  + NullFlavor nullFlavor;
  + Cd code;
  + String value;
* ST
  + NullFlavor nullFlavor;
  + String value;
* STNT
  + NullFlavor nullFlavor;
  + String value;
* TEL
  + NullFlavor nullFlavor;
  + URI value;
* TELEMAIL
  + NullFlavor nullFlavor;
  + URI value;
* TELPERSON
  + NullFlavor nullFlavor;
  + URI value;
* TELPHONE
  + NullFlavor nullFlavor;
  + URI value;
* TELURL
  + NullFlavor nullFlavor;
  + URI value;
* TS
  + NullFlavor nullFlavor;
  + Ed expression;
  + EdText originalText;
  + Qty uncertainty;
  + UncertaintyType uncertaintyType;
  + Date value;

## RESTful API Unit Test Results

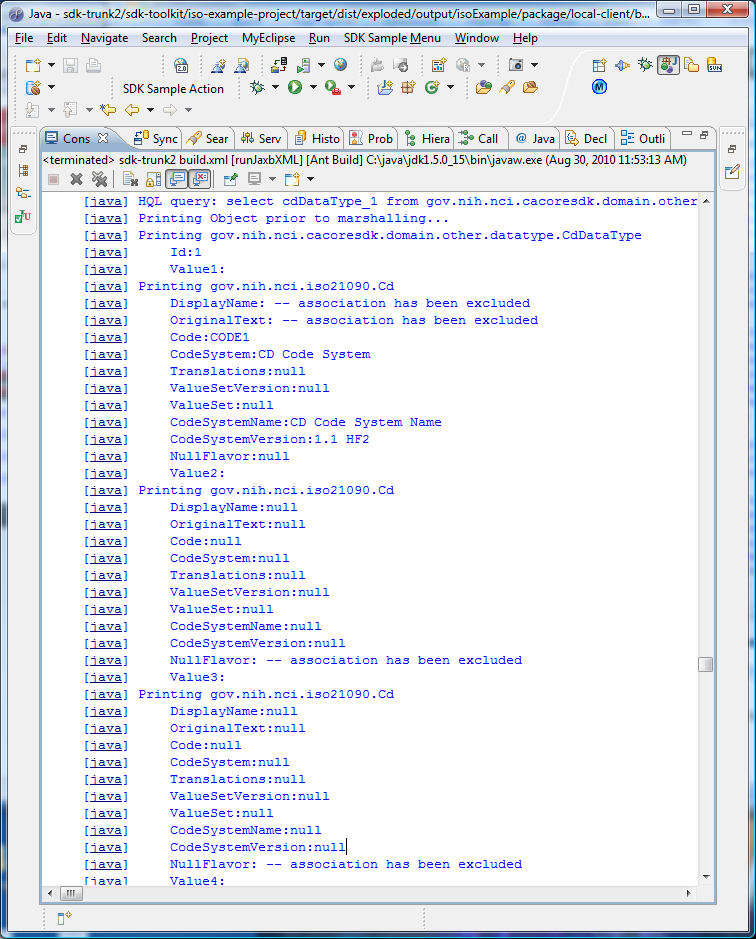
* After completing build of SDK sample system, execute junit HttpUtilsTest (sdk-toolkit/iso-example-project/junit/src/test/gov/nih/nci/system/web/util/HttpUtilsTest.java). All tests should complete successfully without reporting any failures or errors. A sample of the execution results is shown below:



* After completing build of SDK sample system, Run junit SearchUtilsTest (sdk-toolkit/iso-example-project/junit/src/test/gov/nih/nci/system/web/util/SearchUtilsTest.java). ). Should execute all tests without reporting any failures or errors. A sample of the execution results is shown below:



* After completing build of SDK sample system, run *runJaxbXML* target of the sdk-toolkit/iso-example-project/target/dist/exploded/output/isoExample/package/local-client/build.xml script. The script should complete successfully, and only report known errors related to implicit classes (AngelFish, Fish, Tank, etc.). A sample of the execution results is shown below:



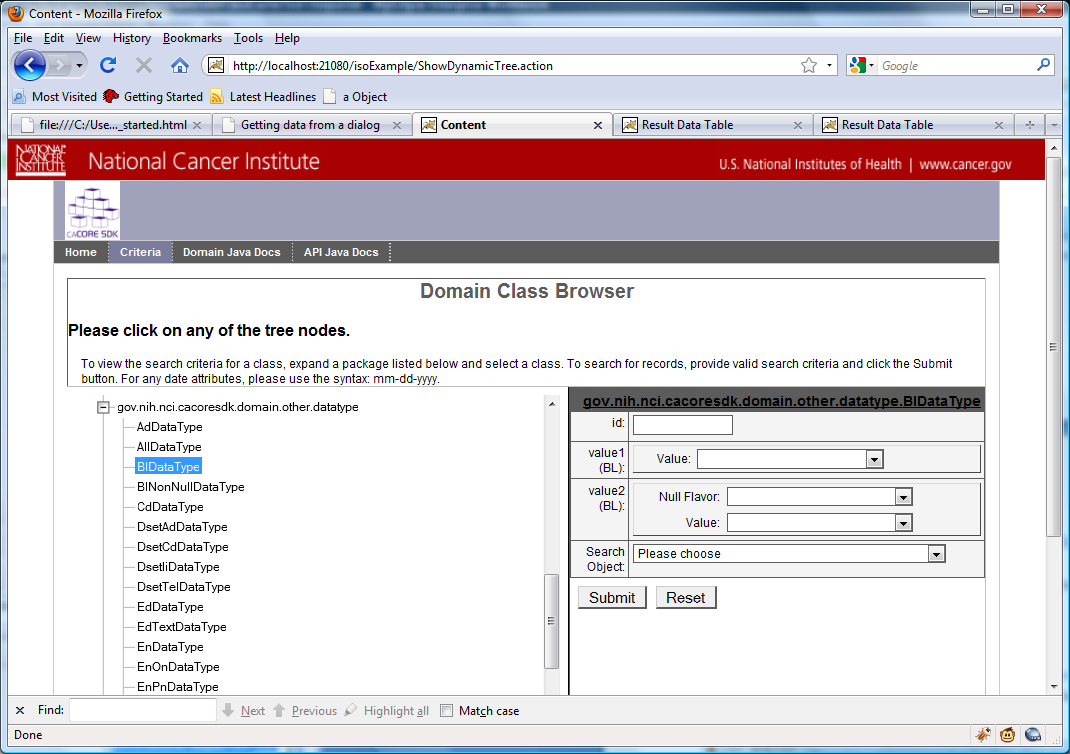
## UI Unit Tests and Results

* For any given ISO data type attribute, validate that Request Post parameters are properly submitted and received at the server
* For any given ISO data type attribute, validate that the Request Post parameters are correctly translated into a corresponding GetHTML RESTful API query
* For any given ISO data type attribute, validate that the GetHTML RESTful API query gets properly translated into a corresponding hibernate HQL query
* For any given ISO data type attribute, validate that the expected HTML results from the HQL query are returned and properly displayed in a new Window

The following sections provide details as to how the UI Unit tests can be performed.

## A Sample of Simple ISO Data Type BL UI Search Criteria Input Screen:

The following diagram illustrates a portion of the generated Search Criteria UI screen for a simple ISO Data Type, BL (Boolean):



As shown, attribute ‘value1’ has a single field, ‘value’. Attribute ‘value2’ has two fields, ‘nullFlavor’ and ‘value’. This illustrates the use of the metadata to only show fields that are mapped to the database.

The corresponding Hibernate metadata used to dynamically generate this UI is shown below:

Class Name gov.nih.nci.cacoresdk.domain.other.datatype.BlDataType

Attribute Name >> value1

Fields >>>> value

Attribute Name >> value2

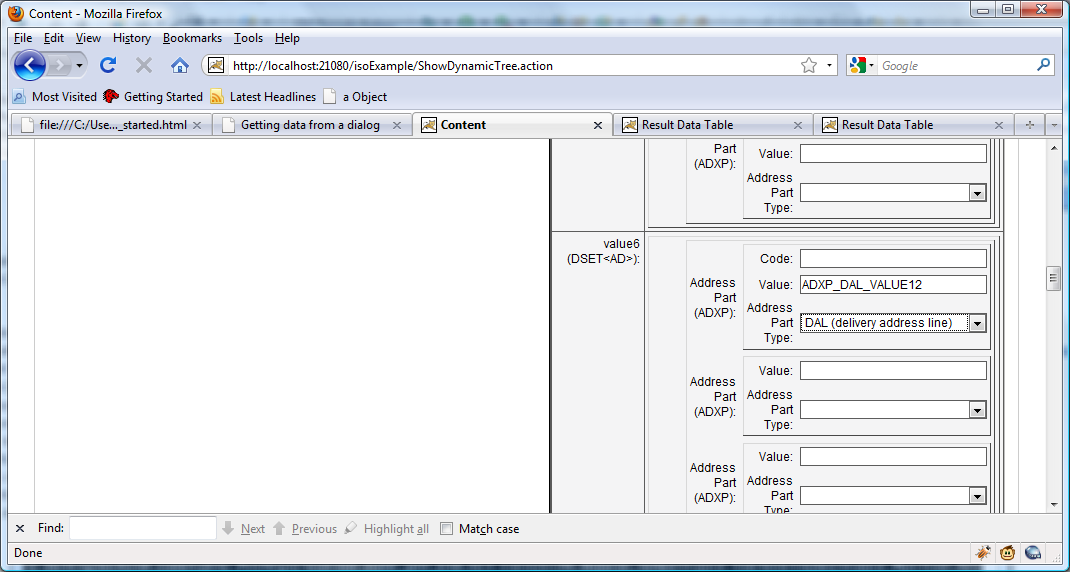
Fields >>>> nullFlavor , value

Attribute Name >> id

Fields >>>> id

## A Portion of Complex DSET AD UI Search Criteria Input Screen:

The following diagram illustrates a portion of the generated Search Criteria UI screen for complex ISO Data type DSET<AD>, which is basically a collection of AD (Address) parts (ADXP):



As shown, Address Part one has three fields for the ADXP part, Code, Value, and Address Part Type. Address Part Two only has two fields, Value and Address Part Type. This illustrates the use of the metadata to only show fields that are mapped to the database.

The corresponding Hibernate metadata used to dynamically generate the UI is shown below. The metadata for attribute ‘value6’ of sample Class DsetAdDatatype is highlighted:

Class Name gov.nih.nci.cacoresdk.domain.other.datatype.DsetAdDataType

Attribute Name >> value1

Fields >>>> {item<gov.nih.nci.iso21090.Ad>=[{part\_0=[value]}]} ,

Attribute Name >> value4

Fields >>>> {item<gov.nih.nci.iso21090.Ad>=[{part\_0=[value, code, codeSystem]}, {part\_1=[value, code, codeSystem]}]} ,

Attribute Name >> value7

Fields >>>> {} ,

Attribute Name >> value2

Fields >>>> {item<gov.nih.nci.iso21090.Ad>=[{part\_0=[value, code]}]} ,

Attribute Name >> value8

Fields >>>> {} ,

Attribute Name >> value5

Fields >>>> {item<gov.nih.nci.iso21090.Ad>=[{part\_0=[value, code, codeSystem]}, {part\_1=[code, codeSystem, value]}, {part\_2=[value, code, codeSystem]}]} ,

Attribute Name >> value6

Fields >>>> {item<gov.nih.nci.iso21090.Ad>=[{part\_0=[value, code]}, {part\_1=[value]}, {part\_2=[value]}, {part\_3=[value]}, {part\_4=[value, code, codeSystem]}, {part\_5=[value]}, {part\_6=[value]}, {part\_7=[value]}, {part\_8=[value]}, {part\_9=[value]}, {part\_10=[value, code, codeSystem]}, {part\_11=[value]}, {part\_12=[value]}, {part\_13=[value]}, {part\_14=[value]}, {part\_15=[value]}, {part\_16=[value]}, {part\_18=[value]}, {part\_19=[value]}, {part\_20=[value]}, {part\_21=[value]}, {part\_22=[value, code, codeSystem]}, {part\_23=[value]}, {part\_24=[value]}, {part\_25=[value]}, {part\_26=[value]}, {part\_27=[value, codeSystem, code]}]} ,

Attribute Name >> id

Fields >>>> id ,

Attribute Name >> value3

Fields >>>> {item<gov.nih.nci.iso21090.Ad>=[{part\_0=[value, code, codeSystem]}]} ,

## Debug Output showing progression of submitted query from request parameters to RESTful GetHTML Query, to Ultimately HQL Query:

Note: The debug output listed below is shown on the Tomcat / JBoss app server console.

Request Parameters received at Server:

parameterValue: ADXP\_DAL\_VALUE12

isoParamPrefix: value6.item.part10

isoParamKey: value

parameterValue: ADXP\_DAL\_VALUE12

parameterValue: DAL

isoParamPrefix: value6.item.part10

isoParamKey: type

parameterValue: DAL

key: item

key: part10

key: value

key: type

Translated RESTful GetHTML Query Sent as a Redirect to Server:

GetHTML?query=gov.nih.nci.cacoresdk.domain.other.datatype.DsetAdDataType&gov.nih.nci.cacoresdk.domain.other.datatype.DsetAdDataType[@value6=[@item=[@part=[@value=ADXP\_DAL\_VALUE12][@type=DAL]]]]

### Translated HQL Query

11:00:29,942 INFO [NestedCriteria2HQL] HQL query: select dsetAdDataType\_1 from gov.nih.nci.cacoresdk.domain.other.datatype.DsetAdDataType dsetAdDataType\_1 inner join dsetAdDataType\_1.value6.item as ad\_0 where ad\_0.part\_10.value=? ))

HQL query: select dsetAdDataType\_1 from gov.nih.nci.cacoresdk.domain.other.datatype.DsetAdDataType dsetAdDataType\_1 inner join dsetAdDataType\_1.value6.item as ad\_0 where ad\_0.part\_10.value=? ))

11:00:29,949 INFO [DAO] HQL Query :select dsetAdDataType\_1 from gov.nih.nci.cacoresdk.domain.other.datatype.DsetAdDataType as dsetAdDataType\_1 where dsetAdDataType\_1.id in (select dsetAdDataType\_1.id from gov.nih.nci.cacoresdk.domain.other.datatype.DsetAdDataType dsetAdDataType\_1 inner join dsetAdDataType\_1.value6.item as ad\_0 where ad\_0.part\_10.value=? )))

## Resulting HTML from Query:

Note the use of nested tables to depict ISO data type results for a particular attribute:

