ISO Datatype Support

Scope Document

Version No: 1.2

Last Modified: 3/10/2010

Team : caCORE SDK, caGrid, caAdapter

Client : National Cancer Institute -   
 Center for Bioinformatics and Information Technology,

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 | 2/19/2010 | Scott Oster | Initial Draft |
| 0.2 | 2/19/2010 | Satish Patel | Added content for SDK and caAdapter, applied template |
| 0.3 | 2/19/2010 | Shannon Hastings | Cleanup |
| 0.4 | 2/19/2010 | John Eisenschmidt | Comments, copy edits |
| 0.5 | 2/22/2010 | Satish Patel | Reviewed and incorporated comments |
| 0.6 | 2/22/2010 | Scott Oster | Detailed query plan; removed metadata tasks |
| 0.7 | 2/24/2010 | Ye Wu | Clarified caAdapter section |
| 0.8 | 3/1/2010 | Satish Patel | Reconciled comments and changes |
| 0.9 | 3/1/2010 | Satish Patel | Added SDK constraints in Appendix A |
| 1.0 | 3/3/2010 | John Eisenschmidt | Added Security Appendix |
| 1.1 | 3/3/2010 | Satish Patel | Added SDK security in appendix. Added RESTful assumption |
| 1.2 | 3/10/2010 | Scott Oster | Added CQL section and updated all references from QBE approach to CQL approach |

Review

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

Related Documents

More information can be found in the following related documents:

|  |
| --- |
| **Document Name** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Table of Contents

1. Introduction 4

2. Problem Statement 4

3. Stakeholder and User Descriptions 5

3.1 Stakeholder Summary 5

4. Timeline Requirement 6

5. Assumptions 6

6. Deliverables 6

6.1 ISO21090 Java Library 6

6.2 Introduce Support 7

6.3 caAdapter Support: 7

6.4 SDK Support: 8

6.5 Service Development Process Documentation: 8

7. Release Non-Functional Requirements 11

8. APPENDIX 12

8.1 Appendix A – SDK Constraints on Database Table Configuration Options 12

8.2 Appendix B - Security Support in this Solution 14

8.2.1 Application Security (caCORE SDK) 14

8.2.2 Grid Security 14

Scope for ISO Datatype Support

# Introduction

The purpose of this document is to identify

1. Artifacts needed in order to support the ISO datatype in the core infrastructure tooling
2. High level features of the artifacts
3. Responsible teams to complete the delivery of the artifacts.

# Problem Statement

The Patient Outcomes Data Service (PODS) currently utilizes a custom implementation of a localization of the ISO 21090 data types. This localization provides the capability to serialize to, and deserialize from, the canonical XML representation (as defined by the ISO XML Schema) for the purposes of transport on the grid. Use of this implementation in clients and other implementations is non-trivial, and needs to be supported by caBIG® infrastructure and tooling.

Longer term, more complete localizations may be required to facilitate other use cases or domain usage and a similar, but more comprehensive, solution will be required. Additionally, the full impact of their use on the semantic infrastructure needs to be evaluated, and may introduce additional requirements.

# Stakeholder and User Descriptions

## Stakeholder Summary

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Responsibilities** |
| George Komatsoulis | Deputy Director, NCI |  |
| Charlie Mead | Chief Technology Officer, NCI CBIIT |  |
| Avinash Shanbhag | Director, Core Infrastructure Engineering | Oversees CBIIT caCORE software engineering, and caGrid development |
| Sichen Liu | Associate Director, Core Infrastructure Engineering | Oversees caAdapter, and SDK product management and development |
| Libby Prince | ISO 20190 Project Manager |  |
| PODS Project Team | The PODS implementation is the reference use case for this effort |  |
| COPPA Project Team | Contributor of the ISO datatype’s localized library | Provide support to the development teams to adopt the localized library, provide guidance on the library |
| caGrid Project Team | Development team | Provide support for the ISO datatypes in their managed tools |
| caAdapter Project Team | Development team | Provide support for the ISO datatypes in their managed tools |
| caCORE SDK Project Team | Development team | Provide support for the ISO datatypes in their managed tools |

# Timeline Requirement

* Patient Outcomes Service: Release Late March/Early April
* Infrastructure Tooling: “shortly thereafter”

# Assumptions

* COPPA/PODS localization (30 ISO datatypes, datatype attributes, and inward/outward XML serialization mapping ) is sufficient for the initial iteration of support
* caDSR registration, and other semantic infrastructure representations, are out of scope for the initial iteration.
* OCL constraint enforcement is not required of the ISO datatype Java library
* Data Service (CQL) support will be limited and experimental for the initial iteration. The notion of CQL is tightly coupled to the conceptual object representation of the data being queried, and is expected to come from a registry in the semantic infrastructure (currently caDSR). As integration with the semantic infrastructure is out of scope in the short term, and the nature of such registration is unknown, the short term CQL support will require assumptions be made about how such information be presented in the DomainModel, and in turn, interpreted during query processing.
* SDK will support one specific manner in which each datatype will be mapped to the database. In longer term, additional database structures for each datatype will be supported. (See Appendix A for more details)
* SDK will support only Java API interface to access the data. RESTful and web service interface will not be supported for this release.
* Security will be supported at the Grid service layer only. (See Appendix B for more details).

# Deliverables

## ISO21090 Java Library

An independent software project will be developed by a cross-project team, in support of the following:

1. Java Beans for Localization
2. Serialization support for to/from those Beans to standard ISO XSD
3. Hibernate support for ORM mapping those Beans to database

The existing COPPA implementation will be evaluated for fitness as a starting point, and enhanced as necessary. Minimally, the removal of the intermediary JAXB-generated beans (via more complex mapping descriptions) will be explored.

As mentioned in the assumption, the Hibernate mapping layer for the datatypes will allow one database structure for each datatype. Additional database structures for each datatype will be supported in incremental releases.

**Key Personnel:**

* Scott Oster
* Shannon Hastings
* Steve Langella
* Satish Patel
* Todd Parnell
* Steve Lustbader
* Abe Evans-el

**Dependencies:**

* Existing Patient Outcomes Service implementation

**Deployment Model:**

A software distribution consisting of documentation, jar files, XSDs, and configuration files will be provided.

## Introduce Support

An Introduce datatype extension will be developed to provide *point and click* support for using this library, and ISO data types in service interface (and other schemas which reference them).

**Key Personnel:**

* Scott Oster
* Shannon Hastings
* Steve Langella

**Dependencies:**

* The ISO21090 Java Library developed above

**Deployment Model:**

The new extension will be provided for immediate consumption via the Introduce Software Update site, on the most recent release of caGrid (currently caGrid 1.3).

## caAdapter Support:

caAdapter’s Model Mapping Service (MMS) module will be enhanced to load object models and data model with ISO 21090 datatyps. caAdapter will also perform adequate customization and validation. In addition, a drag and drop interface will be enhanced to perform mapping between object models and data models with ISO datatypes.

**Key Personnel:**

* Eugene Wang
* Ye wu

**Dependencies:**

* The ISO21090 Java Library developed above
* UML tag value specification from SDK team

**Deployment Model:**

caAdapter software distribution and user documentation will be provided.

## SDK Support:

The caCORE SDK 4.2.1 will be enhanced to support ISO 21090 data types. The code generator framework will be enhanced to leverage the ISO 21090 library enabled with the Hibernate support. The SDK’s middleware system (a.k.a. SDK generated system) will be enhanced to provide constrained CQL and HQL query support and serialization of the beans with ISO datatypes.

**Key Personnel:**

* Satish Patel
* Dan Dumitru

**Dependencies:**

* The ISO21090 Java Library developed above

**Deployment Model:**

The SDK distribution consisting of software, UML tag value specification for model mapping, and user documentation will be provided.

## Introduce Data Service Support (CQL):

The caGrid Introduce tool currently provides the ability to create caGrid Data Services that are pre-implemented with support for exposing CQL query access to information stored in an SDK generated system. This support is enabled via the Introduce Extension framework, which in turn, provides the notion of data service “styles.” Each version of the SDK is supported as a “style.” Both Introduce extensions and Data Service styles can be installed into a pre-existing Introduce installation, thus providing the ability to decouple them from the caGrid core release process. In order to support the ISO21090 datatypes in the Data Service Infrastructure, a number of components will be implemented:

1. DomainModel generation

Data Services require standard metadata -- DomainModels -- to describe the object model being used to expose query access to the data exposed by the service. The Data Service infrastructure provides the ability to generate this metadata by extracting it from a Project in caDSR, transforming an XMI model, or leveraging a user-supplied file. We will only be able to use the last two approaches for this effort (as the semantic infrastructure itegration is out of scope), and will therefore need to extend the “from XMI” approach to understand how to represent the ISO21090 datatypes (see implications below in the Approaches)

1. CQL to HQL implementation

The crux of supporting the ISO datatypes in the Data Service infrastructure is in actually implementing the query language against the underlying database, and dealing with the idiosyncrasies of the Object-Relational Mapping (ORM); in this case via Hibernate. We will need to extend the current CQL to HQL (Hibernate’s query language) translation to accommodate how the SDK structures the ORM.

1. Data Service Style support

Lastly, to deliver this support to Data Service developers, we need to provide a Data Service Style that can support the SDK 4.2.1’s support for ISO21090 datatypes. This will provide a wizard to walk the developer through the creation and modification process, customize the service as necessary for the support, and provide the XML Schemas, serialization framework, and libraries needed .

The complex structure of the ISO21090 datatypes presents a unique challenge for a CQL implementation, as CQL as a language is designed to work with a logical model of a class hierarchy, and is built upon the premise of UML Classes having UML Associations to other UML Classes, and consisting of UML Attributes of primitive datatypes (for example, String, integer, Boolean, etc). The ISO21090 “datatypes” are effectively highly nested complex structures; some of which are as complex as other simple UML Class models. In order to implement CQL over a model that uses these datatypes, either the CQL language needs to be extended (i.e. the assumption of restriction solely via simple predicates removed), or the interpretation/presentation of the model needs to change for query purposes (i.e. the DomainModel cannot simply list an ISO21090 datatype as a UMLAttribute datatype, but rather needs to re-express it as a UML Association). The particular approach taken will need to be determined during the analysis and design phase of the project.

Approach 1: Extend CQL

The CQL language could be extended to provide alternative restriction capabilities that are expressive enough to deal with “datatypes” that are actually complex structures. This is briefly described here: [https://cabig-kc.nci.nih.gov/CTMS/KC/index.php/CCTS\_BRIDG\_Architecture#Implementation\_of\_ISO\_Datatypes](https://cabig-kc.nci.nih.gov/CTMS/KC/index.php/CCTS_BRIDG_Architecture" \l "Implementation_of_ISO_Datatypes) as the notion of “ComplexAttribute.” This extension could be achieved via XML Schema extension (subsetting the language), or more natively via the CQL2 extension capabilities (CQL2 is designed with extension in mind). In the former approach, a “standard” CQL query could be sent to the service, and processed normally as long as the query did not “touch” ISO21090 datatypes, but extended CQL queries could additionally be sent by custom clients, which the service would process appropriately (XML Schema natively supports this concept of extension). This has the advantage of being somewhat backwards compatible, but requires “special knowledge” on the client to actually be able to make use of the extensions. In the latter approach (CQL2), the use of extension is more explicitly supported by the language and service infrastructure, and therefore is programmatically discoverable by clients. However, it still requires “special knowledge” on the client-side to know how to leverage these extension points (that in this case are explicitly, instead of implicitly, defined).

Approach 2: Modify the DomainModel presentation

The second approach is to convert the use of an ISO21090 datatype as an UML Attribute datatype into a more “standard” UML Association to a UML Class when it is presented in the DomainModel metadata. A CQL query is interpreted and validated against the DomainModel metadata, which is effectively a representation of the UML model for the data exposed by the Data Service. In this approach, the UML model described would be transformed such that all ISO datatypes were rewritten as Associations to other Classes. The attribute name would be used as the rolename, and the datatype name would be used as the associated Classname. In some cases, where datatypes were localized differently within the same model, intermediary classes would need to be created. For example, if A and B both leverage II, but localize it differently, an AII and BII “Class’ may need to be created in the DomainModel with A associated to AII and B associating to BII. The processing of the CQL query would be largely unaffected by this, it would just need to understand the transformation algorithm used, and do the converse mapping when constructing the HQL query. The main downside to this approach is that the conceptual model used to query the service is slightly different than the “true” model, which is more of an issue when integrating with an official registry in the semantic infrastructure. If this approach is taken long term, the registry will need to use a similar transformation (or at least accommodate it). In this approach, the CQL language would not change, nor would any “special knowledge” be required on the client side to query a Data Service leveraging ISO21090 datatypes; the query could be constructed purely via DomainModel introspection as is the standard approach.

**Key Personnel:**

* David Ervin
* Scott Oster
* Shannon Hastings
* Satish Patel

**Dependencies:**

* The ISO21090 Java Library developed above
* caCORE SDK Support
* Introduce Support

**Deployment Model:**

The new extension will be provided for immediate consumption via the Introduce Software Update site, on the most recent release of caGrid (currently caGrid 1.3).

## Service Development Process Documentation:

The process of developing a grid data service that leverages the ISO21090 in its operations, and queries the SDK, will be documented. The documentation will describe how to use the SDK to create the generated system, create a new grid service in Introduce, add the ISO21090 data types to the service via the Introduce support, and create and implement grid service operations which pass these data types. A step-by-step tutorial will be provided, as well as a reference implementation “example service.” The tutorial and example service will provide reusable code snippets that demonstrate service layer operations in the grid service interface implemented via the SDK-generated APIs. It is expected this process will be refined over time, and will drive the requirements for later iterations of the infrastructure support.

**Key Personnel:**

* Scott Oster
* Shannon Hastings
* Steve Langella
* David Ervin
* Satish Patel

**Dependencies:**

* The ISO21090 Java Library developed above
* caCORE SDK Support
* Introduce Support
* Introduce Data Service Support

**Deployment Model:**

A tutorial will be available on the caGrid website, with a downloadable “solution service.”

# Release Non-Functional Requirements

1. **Open Source**

The product will continue to use open source tools and technologies. If an appropriate open source software or tool is not available then prior approval from the product manager will be required before using the tool.

1. **Technology Stack compatibility**

The products will continue to adhere to the CBIIT technology stack when choosing the versions of the tools it’s using. Any deviation from the technology stack will be pre-approved by the product manager.

1. **Automated builds**

The products will continue to support their existing automated build processes.

# APPENDIX

## Appendix A – SDK Constraints on Database Table Configuration Options

The ISO datatype is a complex structure with multiple attributes within each datatype. The nested attributes within each datatype can be of simple string type or can be other complex ISO datatypes. As these datatypes are used in the persistence tier, the user has to map these datatypes in the database. In order to perform mapping of the datatypes from the object layer to the database, one has to prepare the database structure to store the information in each of the datatype. Due to the complex nature of the ISO 21090 datatype structure, one can map the datatype in more than one way in the database table.

**Example:**

AD is one of the commonly used ISO datatype. The UML diagram from AD is shown below:



The AD datatype can be mapped in more than one way to the relational database. Following are two of the possible alternatives to map the datatypes to the relational database.

1. Map attributes of the AD datatype in the columns of a separate table with foreign key reference to the table corresponding to the Person object. This scenario is illustrated by PERSON1 and ADDRESS1 tables.
2. Map attributes of the AD datatype in the columns of the table corresponding to the Person object. This scenario is illustrated by PERSON2 table.

These two database mapping options for AD datatype are illustrated in the diagram below:



**Advantage:**

The proposed approach of providing support for one way to perform database table mapping will limit the delivery scope for the development teams in following manner

1. **SDK:**
2. Reduced number of O/R configurations in the Hibernate user types library
3. Reduced number of configurations in the code generator
4. Reduced validation requirements
5. Simplified Query By Example translator
6. Simplified Query specifications for the end user
7. **caAdapter:**
8. Reduced number of O/R configurations
9. Reduced validation requirements
10. **caGrid:**
11. Simplified Query By Example translator
12. Simplified Query specifications for the end user

Given the short timeframe of the project, the reduced scope will help the development team to focus on the core and essential functionalities to make the solution work.

**Disadvantages:**

In the first release of ISO21090 implementation, the SDK will allow using one of the possible options to create the database table structure. Due to the proposed restriction, the user of the SDK will be required to create data model based on the specifications provided by the SDK.

**Resolution:**

The SDK’s proposed architecture to support the ISO21090 datatype can be extended in future to provide support for alternate database configurations not supported in the initial version. SDK team can incrementally add support for these additional database configurations in the future.

## Appendix B - Security Support in this Solution

### Application Security (caCORE SDK)

Security features in the caCORE SDK will not be supported in this release. User can secure the application using the grid security when creating the grid service.

### Grid Security

#### Authentication

Existing x.509/PKI-based authentication will work with ISO-type services as it does with standard data and analytical services.

#### Authorization

The use of CSM and/or Grid Grouper to perform authorization decisions on the invocation of service operations will work with ISO-type services as it does with standard data and analytical services.