OHF XDS Metadata Model

Architecture & API Documentation

Version 0.0.1

seknoop[AT]us[DOT]ibm[DOT]com | Sarah Knoop

Contents

1. Introduction

The Eclipse Foundation is a not-for-profit corporation formed to advance the creation, evolution, promotion, and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services. Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software.

www.eclipse.org

The Eclipse Open Healthcare Framework (EOHF) is a project within Eclipse formed for the purpose of expediting healthcare informatics technology. The project is composed of extensible frameworks and tools which emphasize the use of existing and emerging standards in order to encourage interoperable open source infrastructure, thereby lowering integration barriers.

www.eclipse.org/ohf

The Integrating the Healthcare Enterprise (IHE) is an initiative by healthcare professionals and industry to improve the way computer systems in healthcare share information. IHE promotes the coordinated use of established standards such as DICOM and HL7 to address specific clinical needs in support of optimal patient care. Systems developed in accordance with IHE communicate with one another better, are easier to implement, and enable care providers to use information more effectively.

www.ihe.net

The IHE Technical Frameworks are a resource for users, developers and implementers of healthcare imaging and information systems. They define specific implementations of established standards to achieve effective systems integration, facilitate appropriate sharing of medical information and support optimal patient care. They are expanded annually, after a period of public review, and maintained regularly by the IHE Technical Committees through the identification and correction of errata.

http://www.ihe.net/Technical_Framework/index.cfm

This document describes the current release of the Eclipse OHF model representation of IHE XDS Metadata. The OHF XDS Metadata Model is intended to provide a conceptual representation of XDS Metadata as well as functions that format the model in variety of representations, including ebXML. The XDS Metadata Model itself is generated by EMF. The process of formulating an XDS Metadata Model instance from a data source is referred to as “extraction” or “metadata extraction”. The process of producing an alternative representation from a XDS Metadata Model instance is referred to as “transformation” or “metadata transformation”.

This model and its companion extraction and transformation plugins are used to support the OHF XDS Document Source and the OHF XDS Document Consumer in the following IHE Transactions: ITI-15: Provide and Register Document Set, ITI-16: Query Registry, ITI-17: Retrieve Document and ITI-18: Registry Stored Query. This model can also be used by other XDS actors, such as the XDS Document Registry or XDS Document Repository, as well as applications that utilize the OHF XDS Document Source and OHF XDS Document Consumer.

2. Getting Started

2.1 Platform Requirements

Verify that the following platform requirements are installed on your workstation, and if not follow the links provided to download and install.

Eclipse SDK 3.2, or later	http://www.eclipse.org/downloads/
Java JDK 1.4.2, or later	http://java.sun.com/javase/downloads/index.jsp
Eclipse Modeling Framework 2.2.0	http://www.eclipse.org/emf/

2.2 Source Files

Information on how to access the Eclipse CVS technology repository is found on the eclipse wiki:

http://wiki.eclipse.org/index.php/CVS_Howto

Download from dev.eclipse.org/technology/org.eclipse.ohf/plugins

- org.eclipse.ohf.ihe.xds.metadata
- org.eclipse.ohf.ihe.xds.metadata.extract
- org.eclipse.ohf.ihe.xds.metadata.transform

For details regarding plugin contents, see the README.txt located in the resources/doc folder of each plugin.

2.3 Dependencies

Document dependencies the source has (feel free to extend this section)

2.3.1 Other OHF Plugins

Plugin dependencies include the following from dev.eclipse.org/technology/org.eclipse.ohf/plugins

• org.apache.log4j	Debug, warning and error logging
• org.eclipse.ohf.ihe.common.ebXML._2._1	Model of ebXML v2.1: rim, query and rs

2.3.2 External Sources

The XDS Metadata Model and its companion extraction and transformation plugins have no external dependencies at this time.

2.4 Resources

2.4.1 Extraction and Transformation Documentation

[COMMENT] SEK – may reference extraction/transformation table that documents how the model is re-formatted in ebXML, time permitting.

2.4.2 IHE ITI Technical Framework

Nine IHE IT Infrastructure Integration Profiles are specified as Final Text in the Version 2.0 ITI Technical Framework: Cross-Enterprise Document Sharing (XDS), Patient Identifier Cross-Referencing (PIX), Patient Demographics Query (PDQ), Audit trail and Node Authentication (ATNA), Consistent Time (CT), Enterprise User Authentication (EUA), Retrieve Information for Display (RID), Patient Synchronized Applications (PSA), and Personnel White Pages (PWP).

The IHE ITI Technical Framework can be found on the following website: http://www.ihe.net/Technical_Framework/index.cfm#IT.

Key sections relevant to the OHF XDS Document Consumer include (but are not limited to):

- Volume 1, Section 10 and Appendices A,B, E, J, K, L and M
- Volume 2, Section 1, Section 2, Sections 3.14, 3.15, 3.16 and Appendices B, E, J and L

2.4.3 Newsgroup

Any unanswered technical questions may be posted to Eclipse OHF newsgroup. The newsgroup is located at news://news.eclipse.org/eclipse.technology.ohf.

You can request a password at: http://www.eclipse.org/newsgroups/main.html.

3. API Documentation

The XDS Metadata model is a conceptual representation of XDS Metadata. The companion plugin org.eclipse.ohf.ihe.xds.metadata.extract provides methods to create whole (ProvideAndRegisterDocumentSetType) or partial (DocumentEntryType, for example) instances of the XDS Metadata Model from ebXML instances, files formatted in ebXML or files compliant with the metadata.xsd (see Appendix A – Metadata.xsd). The companion plugin org.eclipse.ohf.xds.metadata.transform provides methods to render whole or partial instances of the XDS Metadata Model to other formats. The use cases below highlight how to create and manipulate XDS Metadata Model instances as well as how to use some of the extraction and transformation functions. Compliance with the edits to the IHE Technical Framework for 2006 is pending. Integration with OHF HL7v2 tooling is pending.

3.1 Use Case 1 – Creating Whole and Partial Model Instances

Below we describe how to create whole and partial instances of the metadata model.

3.1.1 Flow of Execution

XDS Metadata Model instances are created via an EMF generate factory called MetadataFactory and enabled through a single method call.

3.1.2 API Highlights

Only one method is highlighted in the following section for the creation of whole or partial instances of the XDS Metadata Model. It does not particularly need further explanation here; however, the complete javadoc can be downloaded from the Eclipse CVS technology repository. See 2.2 for details.

3.1.3 Sample Code

3.1.3.1 Description

Below we include sample code for the creation of ProvideAndRegisterDocumentSetType, SubmissionSetType, FolderType, and DocumentEntryType instances. We note that instances created by the MetadataFactory are not populated with any default values and would need to be populated with metadata values to be of use.

3.1.3.2 Code

```
////////////////////////////////////////////////////////////////////////////
//Creation of a ProvideAndRegisterDocumentSetType instance (the "whole" model)
////////////////////////////////////////////////////////////////////////////
ProvideAndRegisterDocumentSetType docSet =
MetadataFactory.eINSTANCE.createProvideAndRegisterDocumentSetType();

////////////////////////////////////////////////////////////////////////////
//Creation of a SubmissionSetType, representative of just XDS Submssion Set
//Metadata
////////////////////////////////////////////////////////////////////////////
SubmissionSetType subSet = MetadataFactory.eINSTANCE.createSubmissionSetType();
```

```
////////////////////////////////////////////////////////////////////////////
//Creation of a FolderType, representative of just XDS Folder Metadata
////////////////////////////////////////////////////////////////////////////
FolderType folder = MetadataFactory.eINSTANCE.createFolderType();

////////////////////////////////////////////////////////////////////////////
//Creation of a DocumentEntryType, representative of just XDS Document Entry
//Metadata
////////////////////////////////////////////////////////////////////////////
DocumentEntryType docEntry =
MetadataFactory.eINSTANCE.createDocumentEntryType();
```

3.2 Use Case 2 – Manipulating Whole and Partial Model Instances

Below we describe how to manipulate metadata values in the model, focusing on the DocumentEntryType.

3.2.1 Flow of Execution

Several methods are highlighted in the section below. There is no ordering requirements on these method calls; thereby no flow of execution.

3.2.2 API Highlights

Documentation regarding the metadata attributes highlighted below is best obtained from the ITI Technical Framework, Volume 2, Section 3.14. See 2.4.2 for details. The complete javadoc for the XDS Metadata Model can be downloaded from the Eclipse CVS technology repository; however, since EMF was used to generate the entire model representation, it may not be as descriptive as one would desire. See 2.2 for details.

3.2.3 Sample Code

3.2.3.1 Description

Below we demonstrate how to set and get metadata values for the DocumentEntryType (representative of XDS Document Entry metadata). We also highlight a small “gottcha” regarding “copying” of values between EMF-based model instances.

3.2.3.2 Code

```
////////////////////////////////////////////////////////////////////////////
//Assume that a DocumentEntryType has been created as in 3.1.3.2
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//Set the typeCode on the Document Entry
////////////////////////////////////////////////////////////////////////////

CodedMetadataType typeCode =
MetadataFactory.eINSTANCE.createCodedMetadataType();

typeCode.setCode(“11488-4”);

typeCode.setDisplayName(“Consultation Note”);
```

```
typeCode.setSchemeName("LOINC");

docEntry.setTypeCode(typeCode);


////////////////////////////////////////////////////////////////////////////
//Set the serviceStartTime and serviceStopTime on the Document Entry
////////////////////////////////////////////////////////////////////////////

docEntry.setServiceStartTime("20030815");

docEntry.setServiceStopTime("20050726");


////////////////////////////////////////////////////////////////////////////
//Set title on the Document Entry
////////////////////////////////////////////////////////////////////////////

docEntry.setTitle("This is an example");


////////////////////////////////////////////////////////////////////////////
//Set the patientId on the Document Entry
////////////////////////////////////////////////////////////////////////////

CX patientId = MetadataFactory.eINSTANCE.createCX();

patientId.setIdNumber("1234567890");

patientId.setAssigningAuthorityUniversalId("1.3.6.1.4.1.21367.2005.1.1");

patientId.setAssigningAuthorityUniversalIdType("ISO");

docEntry.setPatientId(patientId);


////////////////////////////////////////////////////////////////////////////
//Assume now that a SubmissionSetType has been created as in 3.1.3.2. Now we
//want the SubmissionSet.patientId to be the same as the DocumentEntry.patientId
// … we just "copy", right?
////////////////////////////////////////////////////////////////////////////

~~subset.setPatientId(docEntry.getPatientId());~~

////////////////////////////////////////////////////////////////////////////
// NO!!!!! EMF models try to be true to XML, in that elements are unique and can
//only exist in one place. The net effect of the above code is that it MOVES the
//CX we created for DocumentEntry.patientId to SubmissionSet.patientId leaving
//DocumentEntry.patientId empty. The following will work to replicate the
//patientId in the Submission Set metadata.
////////////////////////////////////////////////////////////////////////////

CX ssPatientId = MetadataFactory.eINSTANCE.createCX();

ssPatientId.setIdNumber(docEntry.getPatientId().getIdNumer());

ssPatientId.setAssigningAuthorityUniversalId(
docEntry.getPatientId().getAssigningAuthorityUniversalId());

ssPatientId.setAssigningAuthorityUniversalIdType(
docEntry.getPatientId().getAssigningAuthorityUniversalIdType());
```

```
subSet.setPatientId(patientId);
```

3.3 Use Case 3 – Extracting Metadata from ebXML Instance

In this use case we show how to extract Document Entry metadata from an ebXML ExtrinsicObject instance from the published ebXML v2.1 model org.eclipse.ohf.ihe.common.ebxml._2._1. The extraction of Submission Set or Folder metadata from an ebXML RegistryPackage or an entire ProvideAndRegisterDocumentSet from an ebXML SubmitObjectRequest is analogous to this use case.

3.3.1 Flow of Execution

Steps for extracting metadata from an ebXML instance are as follows:

1. obtain an ebXML instance populated with metadata values
2. create the corresponding extractor for that metadata
3. invoke the extract() method

3.3.2 API Highlights

Below we highlight the extract() method on the EbXML_2_1DocumentEntryExtractor used in the following section. A similar extract function is available to extract Submission Set, Folder and complete XDS Metadata from their corresponding ebXML instances. The complete javadoc can be downloaded from the Eclipse CVS technology repository. See 2.2 for details.

EbXML_2_1DocumentEntryExtractor()

EbXML_2_1DocumentEntryExtractor(ExtrinsicObjectType docData, AssociationType1 parentData)
Loads the ebXML structures containing Document Entry metadata. UUIDs for classificationSchemes, identificationSchemes, objectTypes and associationTypes are expected to conform to those fixed by XDS documentataion.

Parameters:

`docData` - the ExtrinsicObject for the document entry metadata.

`parentData` - the Association with the parent of this document. Can be null if no parent document is referenced.

EbXML_2_1DocumentEntryExtractor.extract()

DocumentEntryType	**extract**() Builds an DocumentEntryType containing XDSDocumentEntry metadata, including parent document data, from the ExtrinsicObject and Association parameter provided. Will extract the ExtrinsicObject/@id for the entryUUID metadata. Construction assumes that standard HL7 V2.5 message delimiters are used. See MessageDelimiters.

	Length restrictions are not implemented, presently.
	Specified by:
	`extract` in interface `DocumentEntryExtractor`
	Returns:
	`DocumentEntryType` object populated with metadata from the extractor.
	Throws:
	`MetadataExtractionException`

3.3.3 Sample Code

3.3.3.1 Description

Below we demonstrate Document Entry metadata extraction from an ExtrinsicObject instance from the model org.eclipse.ohf.ihe.common.ebXML._2._1.

3.3.3.2 Code

```
////////////////////////////////////////////////////////////////////////////
//Assume we have an ExtrinsicObjectType instance called docData, populated with
//Document Entry metadata values. Additionally, assume that for this particular
//Document Entry that no parent document is to be associated with it. Thus, we
//can call the extraction process with "null" as the second argument.
////////////////////////////////////////////////////////////////////////////

// create the extractor

EbXML_2_1DocumentEntryExtractor deExtractor = new
EbXML_2_1DocumentEntryExtractor(docData, null);


// invoke the extraction process

DocumentEntryType documentEntry = deExtractor.extract();
```

3.4 Use Case 4 – Extracting Metadata from ebXML File

In this use case we show how to extract a complete set of XDS Metadata from a properly formatted ebXML SubmitObjectRequest file. The extraction of Submission Set, Folder or Document Entry metadata from a properly formatted ebXML file is analogous to this use case.

3.4.1 Flow of Execution

Steps for extracting metadata from an ebXML file are as follows:

1. obtain an properly formatted ebXML file populated with metadata values
2. create the corresponding extractor for that metadata

3. invoke the extract() method

3.4.2 API Highlights

Below we highlight the extract() method on the EbXML_2_1FileProvideAndRegisterDocumentSetExtractor used in the following section. A similar extract function is available to extract Submission Set, Folder and Document Entry metadata from their corresponding ebXML files. The complete javadoc can be downloaded from the Eclipse CVS technology repository. See 2.2 for details.

EbXML_2_1FileProvideAndRegisterDocumentSetExtractor()

EbXML_2_1FileProvideAndRegisterDocumentSetExtractor(java.io.File ebXMLmetadataFile)
Creates an extractor for the file given. File must contain single instance of a RegistryPackageType. Must be conformant to the modified rs.xsd. UUIDs in the file are assumed to match those in UUIDs.

Parameters:

ebXMLmetadataFile - file conformant to modified rs.xsd populated with metadata values

EbXML_2_1FileProvideAndRegisterDocumentSetExtractor.extract()

ProvideAndRegisterDocumentSetType	**extract**() Renders a ProvideAndRegisterDocumentSetType object from a file containing a single instance of a SubmitObjectRequest. Must be conformant to the modified rs.xsd. UUIDs in the file are assumed to match those in UUIDs. **Specified by:** extract in interface ProvideAndRegisterDocumentSetExtractor **Throws:** MetadataExtractionException **See Also:** ProvideAndRegisterDocumentSetExtractor.extract()

3.4.3 Sample Code

3.4.3.1 Description

Below we demonstrate complete XDS Metadata extraction from a file containing a single SubmitObjectRequest as specified by ITI Technical Framework XDS documentation and conformant to the

modified rs.xsd available in org.eclipse.ohf.ihe.common.ebXML._2._1/resources/schema from the Eclipse CVS technology repository.

3.4.3.2 Code

```
////////////////////////////////////////////////////////////////////////
//Assume we have a XDS documentation conformant SubmitObjectRequest XML file
//populated with metadata values. Assume we have the file in
//"C:/temp/sample1.xml".
////////////////////////////////////////////////////////////////////////

File file = new File ("C:/temp/sample1.xml");

// create the extractor

EbXML_2_1FileProvideAndRegisterDocumentSetExtractor dsExtractor = new
EbXML_2_1FileProvideAndRegisterDocumentSetExtractor(file);


// invoke the extraction process

ProvideAndRegisterDocumentSetType docSet = dsExtractor.extract();
```

3.5 Use Case 5 – Extracting Metadata from Metadata.xsd Compliant File

In this use case we show how to extract Submission Set Metadata from a XML file that conforms to the published metadata.xsd (see Appendix A – Metadata.xsd). The extraction of Folder or Document Entry metadata as well as complete XDS Metadata for a Provide and Register Document Set Transaction from a schema compliant XML file is analogous to this use case.

3.5.1 Flow of Execution

Steps for extracting metadata from an metadata.xsd compliant XML file are as follows:

1. obtain an properly formatted XML file populated with metadata values
2. create the corresponding extractor for that metadata
3. invoke the extract() method

3.5.2 API Highlights

Below we highlight the extract() method on the FileSubmissionSetExtractor used in the following section. A similar extract function is available to extract ProvideAndRegisterDocumentSet, Folder and Document Entry metadata from their corresponding schema compliant XML files. The complete javadoc can be downloaded from the Eclipse CVS technology repository. See 2.2 for details.

FileSubmissionSetExtractor()

`FileSubmissionSetExtractor(java.io.File subSetFile)`
Creates an extractor for the file given. File must contain a single instance of a SubmissionSetType conformant to the metadata.xsd.

Parameters:

subSetFile - file containing Submission Set metadata values, conformant to metadata.xsd

FileSubmissionSetExtractor.extract()

SubmissionSetType	**extract**() Renders a SubmissionSetType object from a file containing a single instance of a SubmissionSetType conformant to the metadata.xsd. **Specified by:** extract in interface SubmissionSetExtractor **Throws:** MetadataExtractionException **See Also:** SubmissionSetExtractor.extract()

3.5.3 Sample Code

Below we demonstrate complete XDS Metadata extraction from an XML file containing a single SubmissionSet instance conformant to metadata.xsd available in org.eclipse.ohf.ihe.xds.metadata/resources/schema from the Eclipse CVS technology repository.

3.5.3.1 Code

```
////////////////////////////////////////////////////////////////////////////
//Assume we have a metadata.xsd conformant SubmissionSet XML file populated with
//metadata values. Assume we have the file in "C:/temp/sample2.xml".
////////////////////////////////////////////////////////////////////////////

File file = new File ("C:/temp/sample2.xml");

// create the extractor

FileSubmissionSetExtractor setExtractor = new FileSubmissionSetExtractor(file);


// invoke the extraction process

ProvideAndRegisterDocumentSetType docSet = dsExtractor.extract();
```

3.6 Use Case 6 –Transforming Metadata to an ebXML Instance

In this use case we show how to transform complete XDS Metadata for a Provide and Register Document Set Transaction from an XDS Metadata Model instance (ProvideAndRegisterDocumentSetType) to an ebXML SubmitObjectRequest instance from the published ebXML v2.1 model org.eclipse.ohf.ihe.common.ebxml._2._1. The transformation of Document Entry metadata to an ExtrinsicObject or transformation of Submission Set or Folder metadata to an ebXML RegistryPackage is analogous to this use case.

3.6.1 Flow of Execution

Steps for transforming metadata to an ebXML instance are as follows:

1. obtain an XDS Metadata Model instance populated with metadata values
2. create the corresponding transformer for that metadata
3. invoke the transform() method
4. call the appropriate get() method to obtain the ebXML instance

3.6.2 API Highlights

Below we highlight the transform() method on the EbXML_2_1ProvideAndRegisterDocumentSetTransformer used in the following section. A similar transform function is available to extract Submission Set, Folder and Document Entry model instances to their corresponding ebXML instances. The complete javadoc can be downloaded from the Eclipse CVS technology repository. See 2.2 for details.

EbXML_2_1ProvideAndRegisterDocumentSetTransformer()
EbXML_2_1ProvideAndRegisterDocumentSetTransformer() Creates the transformer.

EbXML_2_1ProvideAndRegisterDocumentSetTransformer.transform()	
void	**transform**(ProvideAndRegisterDocumentSetType docSet) Renders a SubmitObjectsRequestType containing the metadata values of the parameter, formatted in ebXML **Specified by:** transform in interface ProvideAndRegisterDocumentSetTransformer **Throws:** MetadataTransformationException

3.6.3 Sample Code

3.6.3.1 Description

Below we demonstrate complete XDS Metadata transformation to SubmitObjectRequest instance from the model org.eclipse.ohf.ihe.common.ebXML._2._1.

3.6.3.2 Code

```
////////////////////////////////////////////////////////////////////////////
//Assume we have a ProvideAndRegisterDocumentSetType called docSet populated
//with metadata values.
////////////////////////////////////////////////////////////////////////////

// create the transformer

EbXML_2_1ProvideAndRegisterDocumentSetTransformer dsTransformer = new
EbXML_2_1ProvideAndRegisterDocumentSetTransformer();


// invoke the transformation process

dsTransformer.transform(docSet);


// get the transformation results

SubmitObjectRequestType subObj = dsTransformer.getSubmitReq();
```

3.7 Use Case 7 –Transforming Metadata to a Metadata.xsd Compliant File

In this use case we show how to transform Folder Metadata to a XML file that conforms to the published metadata.xsd (see Appendix A – Metadata.xsd). The transformation of Submission Set or Document Entry metadata as well as complete XDS Metadata for a Provide and Register Document Set Transaction from a schema compliant XML file is analogous to this use case.

3.7.1 Flow of Execution

Steps for transforming metadata to an ebXML instance are as follows:

1. obtain an XDS Metadata Model instance populated with metadata values
2. create the corresponding transformer for that metadata
3. invoke the transform() method
4. call the appropriate get() method to obtain the ebXML instance

3.7.2 API Highlights

Below we highlight the transform() method on the FileFolderTransformer used in the following section. A similar transform function is available to extract Submission Set, Folder and ProvideAndRegisterDocumentSet model instances to their corresponding XML files. The complete javadoc can be downloaded from the Eclipse CVS technology repository. See 2.2 for details.

FileFolderTransformer()

FileFolderTransformer(java.lang.String path)
Creates a transformer.

Parameters:

path - path to output file

FileFolderTransformer.transform()

void	**transform**(FolderType folder) Renders a File containing a single instance of a FolderType conformant to the metadata.xsd. **Specified by:** transform in interface FolderTransformer **Throws:** MetadataTransformationException

3.7.3 Sample Code

3.7.3.1 Description

Below we demonstrate Folder Metadata transformation to XML file containing a single Folder instance conformant to metadata.xsd available in org.eclipse.ohf.ihe.xds.metadata/resources/schema from the Eclipse CVS technology repository.

3.7.3.2 Code

```
////////////////////////////////////////////////////////////////////////////
//Assume we have a FolderType called folder populated with metadata values. Also
//assume that the file we want produced will have the following path
//"C:/temp/sample3.xml".
////////////////////////////////////////////////////////////////////////////

// create the transformer

FileFolderTransformer folTransformer = new
FileFolderTransformer("C:/temp/sample3.xml");


// invoke the transformation process

folTransformer.transform(folder);
```

```
// get the transformation results
File file = folTransformer.getFolderFile();
```

4. Security

4.1 Node Authentication

Node Authentication is not done by the XDS Metadata Model or its companion extraction and transformation plugins because there is no remote host involved. This could potentially change in the future if extraction and transformation functions are offered as a consolidated online service.

4.2 Auditing

Auditing to an IHE Audit Repository is not done by the XDS Metadata model nor its companion extraction and transformation plugins. The act of auditing a transaction is to take place on the implementation layer that uses this model. In other words, IHE actors (such as the XDS Document Source and XDS Document Consumer) using this plugin are expected to take appropriate auditing measures when utilizing this model and its companion extraction and transformation plugins.

5. Configuration

Logging is the only configurable aspect of these plugins at this time. For more information about logging, consult Section 6 of this document.

6. Debugging Recommendations

The XDS Metadata Model companion plugins for extraction and transformation use Apache Log4j. If you are experiencing bugs related to these functions, you may enable debug level logging by adding the following category to your log4j XML configuration file.

```
 <category name="org.eclipse.ohf.ihe.xds.metadata.extract">
    <priority value="debug" />
 </category >

<category name="org.eclipse.ohf.ihe.xds.metadata.transform">
    <priority value="debug" />
 </category >
```

For more information about Log4j please see: http://logging.apache.org/log4j/docs/

For an example log4j XML configuration file and to see how it is loaded at run time see `org.eclipse.ohf.ihe.xds.source/resources/conf/submitTest_log4j.xml` and `org.eclipse.ohf.ihe.xds.source/src_tests/SubmitTest.java`, respectively. These can be obtained by downloading the plugin `org.eclipse.ohf.ihe.xds.source` from the Eclipse CVS technology repository. See 2.2 for details.

7. Pending Integration and API changes

Below is a laundry list of anticipated changes to the XDS Metadata Model

1. IHE 2006 Change Proposal Integration – Currently in the XDS Metadata Model the attributes concerning the “author” of the document are unrelated. XDS CP 122 out of IHE will bind these pieces of information together, ensuring that for a single authorPerson, their member institutions, specialities and roles can be identified. This will introduce an AuthorType into the metadata model.
2. Support for Linking to Existing Metadata – Document submission in XDS allows a new submission to link to existing documents and folders on the XDS Registry. The XDS Metadata Model will have to add information to support this feature.
3. Support for DSG – Enabling submission of digitally signed documents also requires changes to the metadata model. We are awaiting resolution to internal IHE discussions on how to test DSG at Connectathon 2007 before considering these modifications.
4. Integration with HL7v2 tooling – Since several metadata attributes use HL7v2 data types, we hope to provide verification and serialization of these via OHF HL7v2 tooling. This integration is pending.

8. Additional Sections – repeat as necessary

Any additional sections needed are added at this point.

9. Glossary

Define any non-common knowledge terms or acronyms here. Provide web-site reference if applicable.

Appendix A – Metadata.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:tns="urn:org:eclipse:ohf:ihe:xds:metadata"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:org:eclipse:ohf:ihe:xds:metadata">

      <!--  import the xml namespace for xml:lang tags -->
      <import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <complexType name="ProvideAndRegisterDocumentSetType">
      <sequence>
            <element name="submissionSet" type="tns:SubmissionSetType"/>
            <element maxOccurs="unbounded" minOccurs="0" name="folder" type="tns:FolderType"/>
            <element maxOccurs="unbounded" name="documentEntry" type="tns:DocumentEntryType"/>
      </sequence>
  </complexType>
  <element name="ProvideAndRegisterDocumentSetType" type="tns:ProvideAndRegisterDocumentSetType"/>

  <complexType name="DocumentEntryType">
            <sequence>
                  <element minOccurs="0" maxOccurs="unbounded" name="authorInstitution"
type="tns:XON"/>
                  <element minOccurs="0" maxOccurs="unbounded" name="authorPerson" type="tns:XCN"/>
                  <element minOccurs="0" maxOccurs="unbounded" name="authorRole" type="xsd:string"/>
                  <element minOccurs="0" maxOccurs="unbounded" name="authorSpeciality"
type="xsd:string"/>
                  <element minOccurs="0" name="availabilityStatus" type="tns:AvailabilityStatusType"/>
                  <element minOccurs="0" name="classCode" type="tns:CodedMetadataType"/>
                  <element minOccurs="0" name="comments" type="tns:InternationalStringType"/>
                  <element minOccurs="0" name="confidentialityCode" type="tns:CodedMetadataType"/>
                  <element minOccurs="0" name="creationTime" type="tns:DTM"/>
                  <element minOccurs="0" name="entryUUID" type="xsd:string"/>
                  <element maxOccurs="unbounded" minOccurs="0" name="eventCode"
type="tns:CodedMetadataType"/>
                  <element minOccurs="0" name="formatCode" type="tns:CodedMetadataType"/>
                  <element minOccurs="0" name="hash" type="xsd:string"/>
                  <element minOccurs="0" name="healthCareFacilityTypeCode"
type="tns:CodedMetadataType"/>
                  <element minOccurs="0" name="languageCode" type="xsd:string"/>
```

```
                  <element maxOccurs="unbounded" minOccurs="0" name="legalAuthenticator"
type="tns:XCN"/>
                  <element minOccurs="0" name="LONG-URI" type="xsd:string"/>
                  <element minOccurs="0" name="mimeType" type="xsd:string"/>
                  <element minOccurs="0" name="parentDocumentRelationship">
                        <simpleType>
                              <restriction base="xsd:NMTOKEN">
                                    <enumeration value="XFRM"/>
                                    <enumeration value="RPLC"/>
                                    <enumeration value="APND"/>
                                    <enumeration value="XFRM_RPLC"/>
                              </restriction>
                        </simpleType>
                  </element>
                  <element minOccurs="0" name="parentDocumentId" type="tns:uuid"/>
                  <element minOccurs="0" name="patientId" type="tns:CX"/>
                  <element minOccurs="0" name="practiceSettingCode" type="tns:CodedMetadataType"/>
                  <element minOccurs="0" name="serviceStartTime" type="tns:DTM"/>
                  <element minOccurs="0" name="serviceStopTime" type="tns:DTM"/>
                  <element minOccurs="0" name="sourcePatientId" type="tns:CX"/>
                  <element minOccurs="0" name="size" type="xsd:string"/>
                  <element minOccurs="0" name="sourcePatientInfo" type="tns:SourcePatientInfoType"/>
                  <element minOccurs="0" name="title" type="tns:InternationalStringType"/>
                  <element minOccurs="0" name="typeCode" type="tns:CodedMetadataType"/>
                  <element minOccurs="0" name="uniqueId" type="tns:oid_extension"/>
                  <element minOccurs="0" name="uri" type="xsd:string"/>
            </sequence>
      </complexType>
      <element name="DocumentEntry" type="tns:DocumentEntryType"/>

      <complexType name="SubmissionSetType">
            <sequence>
                  <element minOccurs="0" maxOccurs="unbounded" name="authorInstitution"
type="tns:XON"/>
                  <element minOccurs="0" maxOccurs="unbounded" name="authorPerson" type="tns:XCN"/>
                  <element minOccurs="0" maxOccurs="unbounded" name="authorRole" type="xsd:string"/>
                  <element minOccurs="0" maxOccurs="unbounded" name="authorSpeciality"
type="xsd:string"/>
                  <element minOccurs="0" name="availabilityStatus" type="tns:AvailabilityStatusType"/>
                  <element minOccurs="0" name="comments" type="tns:InternationalStringType"/>
                  <element minOccurs="0" name="contentTypeCode" type="tns:CodedMetadataType"/>
```

```
                  <element minOccurs="0" name="entryUUID" type="xsd:string"/>
                  <element minOccurs="0" name="patientId" type="tns:CX"/>
                  <element minOccurs="0" name="sourceId" type="tns:oid"/>
                  <element minOccurs="0" name="submissionTime" type="tns:DTM"/>
                  <element minOccurs="0" name="title" type="tns:InternationalStringType"/>
                  <element minOccurs="0" name="uniqueId" type="tns:oid"/>
            </sequence>
      </complexType>
      <element name="SubmissionSet" type="tns:SubmissionSetType"/>

      <complexType name="FolderType">
            <sequence>
                  <element maxOccurs="unbounded" minOccurs="0" name="associatedDocuments"
type="xsd:string"/>
                  <element minOccurs="0" name="availabilityStatus" type="tns:AvailabilityStatusType"/>
                  <element maxOccurs="unbounded" minOccurs="0" name="code"
type="tns:CodedMetadataType"/>
                  <element minOccurs="0" name="comments" type="tns:InternationalStringType"/>
                  <element minOccurs="0" name="entryUUID" type="xsd:string"/>
                  <element minOccurs="0" name="lastUpdateTime" type="tns:DTM"/>
                  <element minOccurs="0" name="patientId" type="tns:CX"/>
                  <element minOccurs="0" name="title" type="tns:InternationalStringType"/>
                  <element minOccurs="0" name="uniqueId" type="tns:oid"/>
        </sequence>
      </complexType>
      <element name="Folder" type="tns:FolderType"/>

      <complexType name="CodedMetadataType">
            <sequence>
                  <element minOccurs="0" name="code" type="xsd:string"/>
                  <element minOccurs="0" name="displayName" type="tns:InternationalStringType"/>
                  <element minOccurs="0" name="schemeName" type="xsd:string"/>
                  <element minOccurs="0" name="schemeUUID" type="xsd:string"/>
            </sequence>
      </complexType>

      <!-- Following are adaptations for HL7v2 messaging data types-->
      <complexType name="XON">
            <sequence>
                  <element name="organizationName" type="xsd:string"/> <!-- XON.1 -->
                  <element name="idNumber" type="xsd:string"/> <!-- XON.3 -->
```

```
                  <element name="assigningAuthorityName" type="xsd:string"/> <!-- XON.6.1 -->
                  <element name="assigningAuthorityUniversalId" type="tns:oid"/> <!-- XON.6.2 -->
                  <element name="assigningAuthorityUniversalIdType" type="xsd:string"/> <!-- XON.6.3,
bind to vocab -->
            </sequence>
      </complexType>

      <complexType name="XTN">
            <sequence>
                  <element name="unformattedTelephoneNumber" type="xsd:string"/> <!-- XTN.12 -->
            </sequence>
      </complexType>

      <complexType name="CX">
            <sequence>
                  <element name="idNumber" type="xsd:string"/> <!-- CX.1 -->
                  <element name="assigningAuthorityUniversalId" type="tns:oid"/> <!-- CX.4.2 -->
                  <element name="assigningAuthorityUniversalIdType" type="xsd:string"/> <!-- CX.4.3,
bind to vocab -->
            </sequence>
      </complexType>

      <complexType name="XCN">
            <sequence>
                  <element name="idNumber" type="xsd:string"/> <!-- XCN.1 -->
                  <element name="familyName" type="xsd:string"/> <!-- XCN.2.1 -->
                  <element name="givenName" type="xsd:string"/> <!-- XCN.3 -->
                  <element name="otherName" type="xsd:string"/> <!-- XCN.4 -->
                  <element name="suffix" type="xsd:string"/> <!-- XCN.5 -->
                  <element name="prefix" type="xsd:string"/> <!-- XCN.6 -->
                  <element name="assigningAuthorityName" type="xsd:string"/> <!-- XCN.9.1 -->
                  <element name="assigningAuthorityUniversalId" type="tns:oid"/> <!-- XCN.9.2 -->
                  <element name="assigningAuthorityUniversalIdType" type="xsd:string"/> <!-- XCN.9.3 --
>
            </sequence>
      </complexType>
      <complexType name="XPN">
            <sequence>
                  <element name="familyName" type="xsd:string"/> <!-- XPN.1.1 -->
                  <element name="givenName" type="xsd:string"/> <!-- XPN.2 -->
                  <element name="otherName" type="xsd:string"/> <!-- XPN.3 -->
```

```
                  <element name="suffix" type="xsd:string"/> <!-- XPN.4 -->
                  <element name="prefix" type="xsd:string"/> <!-- XPN.5 -->
            </sequence>
      </complexType>

      <complexType name="XAD">
            <sequence>
                  <element name="streetAddress" type="xsd:string"/> <!-- XAD.1.1 -->
                  <element name="otherDesignation" type="xsd:string"/> <!-- XAD.2 -->
                  <element name="city" type="xsd:string"/> <!-- XAD.3 -->
                  <element name="stateOrProvince" type="xsd:string"/> <!-- XAD.4 -->
                  <element name="zipOrPostalCode" type="xsd:string"/> <!-- XAD.5 -->
                  <element name="country" type="xsd:string"/> <!-- XAD.6 -->
                  <element name="countyParishCode" type="xsd:string"/> <!-- XAD.9 -->
            </sequence>
      </complexType>

      <complexType name="SourcePatientInfoType"><!-- PID Segment -->
            <sequence>
                  <element name="patientIdentifier" type="tns:CX" maxOccurs="unbounded"/> <!-- PID-3 --
>
                  <element name="patientName" type="tns:XPN" maxOccurs="unbounded"/> <!-- PID-5 -->
                  <element name="patientDateOfBirth" type="tns:DTM"/> <!-- PID-7 -->
                  <element name="patientSex" type="xsd:string"/> <!-- PID-8, bind to restricted vocab
in XDS: M,F,U,O -->
                  <element name="patientAddress" type="tns:XAD"/> <!-- PID-11 -->
                  <element name="patientPhoneHome" type="tns:XTN"/> <!-- PID-13 -->
                  <element name="patientPhoneBusiness" type="tns:XTN"/> <!-- PID-14 -->
            </sequence>
      </complexType>

      <!-- taken from HL7 CDA R2 schema (ts type) -->
      <simpleType name="DTM">
            <restriction base="xsd:string">
                  <pattern value="[0-9]{1,8}|([0-9]{9,14}|[0-9]{14,14}\.[0-9]+)([+\-][0-9]{1,4})?"/>
            </restriction>
      </simpleType>

      <!-- adapted from HL7 CDA R2 schema (uuid type with pre-pend of "urn:uuid:") -->
      <simpleType name="uuid">
            <restriction base="xsd:string">
```

```
                  <pattern value="urn:uuid:[0-9a-zA-Z]{8}-[0-9a-zA-Z]{4}-[0-9a-zA-Z]{4}-[0-9a-zA-Z]{4}-
[0-9a-zA-Z]{12}"/>
            </restriction>
      </simpleType>

      <!-- taken from HL7 CDA R2 schema -->
      <simpleType name="oid">
            <restriction base="xsd:string">
                  <pattern value="([1-9][0-9]*)(\.[1-9][0-9]*)*"/>
            </restriction>
      </simpleType>

      <!-- adapted from HL7 CDA R2 schema (oid type with optional suffix of "(^[0-9]+)") -->
      <simpleType name="oid_extension">
            <restriction base="xsd:string">
                  <pattern value="([1-9][0-9]*)(\.[1-9][0-9]*)*(^[0-9]+)?"/>
            </restriction>
      </simpleType>

      <!-- Taken from ebXML2.1 schema -->
      <complexType name="InternationalStringType">
            <sequence maxOccurs="unbounded" minOccurs="0">
                  <element name="LocalizedString" type="tns:LocalizedStringType"/>
            </sequence>
      </complexType>

      <!-- Taken from ebXML2.1 schema, attribute value changed from an ebXML type to xds:string -->
      <complexType name="LocalizedStringType">
            <attribute default="en-us" ref="xml:lang"/>
            <attribute default="UTF-8" name="charset"/>
            <attribute name="value" type="xsd:string" use="required"/>
      </complexType>

      <!-- Taken from ebXML2.1 rim schema -->
      <simpleType name="AvailabilityStatusType">
            <restriction base="xsd:NMTOKEN">
                  <enumeration value="Submitted"/>
                  <enumeration value="Approved"/>
                  <enumeration value="Deprecated"/>
                  <enumeration value="Withdrawn"/>
            </restriction>
```

```
      </simpleType>
</schema>
```