EMC® XDS Repository Connector for Documentum®

Version 1.7

Installation Guide

EMC Corporation
Corporate Headquarters
Hopkinton, MA 01748-9103
1-508-435-1000
www.EMC.com

Documentation Feedback

Your opinion matters. We want to hear from you regarding our product documentation. If you have feedback about how we can make our documentation better or easier to use, please send us your feedback directly at IIGDocumentationFeedback@emc.com

Table of Contents

Revision History

Revision Date	Description
March 2015	Initial publication.

Chapter 1

About XDS Repository Connector for Documentum

This chapter contains the following topics:

Overview

The Cross-enterprise Document Sharing (XDS) Repository Connector for Documentum enables you to share patient medical records stored in a Documentum Repository through XDS. Healthcare Integration Portfolio (HIP) implements industry standard Integrating the Healthcare Enterprise (IHE) XDS.b transactions which allow the hospitals and organizations that comprise your healthcare community to submit and consume patient medical records and healthcare information.

The XDS Repository Server supports:

- Documentum as a cross-enterprise repository for storing and maintaining healthcare content
- Operating as a local repository maintained by an individual provider, or as a central repository where multiple providers can store and share healthcare records
- Automatic registration of all healthcare documents submitted to the Documentum Repository through the XDS Repository Server
- Documentum specific processors to register/unregister Documentum documents with the XDS Registry
- Processing of Admission, Discharge, Transfer (ADT) and Merge Patient Identities requests
- Sending outbound Medical Document Management (MDM) messages to inform external Health Level Seven (HL7) systems about the newly added content and cancelled content
- Custom processing for Provide and Register (PnR)
- Trusted Node Authentication through TLS certificates

- User authentication through Cross-Enterprise User Assertion (XUA)
- Message Queue for Inbound and Outbound HL7 messages
- Integration with PPIC Server to enforce Patient Privacy Policies whenever requests are sent to the XDS Repository Server to retrieve patient records.

Components

The XDS Repository Connector for Documentum consists of the following components:

- EMC Documentum Content Server (with the Healthcare Information Model for Documentum)
- XDS Repository Server

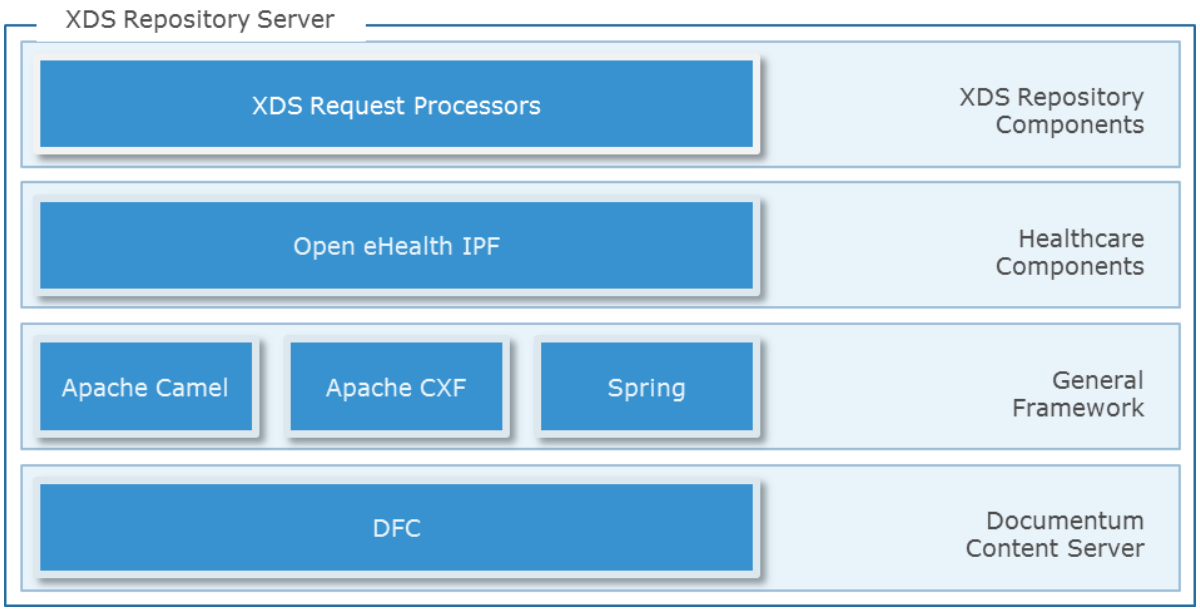
You can deploy both components on the same machine or on separate machines. Deploying the components in separate machines is recommended to optimize performance and scalability.

Load Balancing and Scalability, page 30 provides details on scalability and load balancing.

Architecture

The XDS Repository Server is a J2EE web application built on the Apache Camel open-source routing and mediation framework. All XDS document content resides in the Documentum repository.

The following figure shows the XDS Repository Server architecture:



The top layer contains XDS Request Processors that handle and process request messages. These processors handle messages for generating Documentum Content Server requests using Documentum Foundation Classes (DFC) and for retrieving documents from the Documentum Content Server.

The second layer contains healthcare components from the Open eHealth Integration Platform (IPF). These are Apache Camel specific components for the XDS Repository that include request validators, SOAP components, and message convertors.

The third layer contains the general application framework which includes Apache Camel, Apache CXF and Spring. Apache CXF is an Apache Camel component that handles message requests formats from a wide variety of formats. This layer also uses the Spring ApplicationContext to provide configuration properties to the application.

The fourth layer consists of the Documentum Content Server.

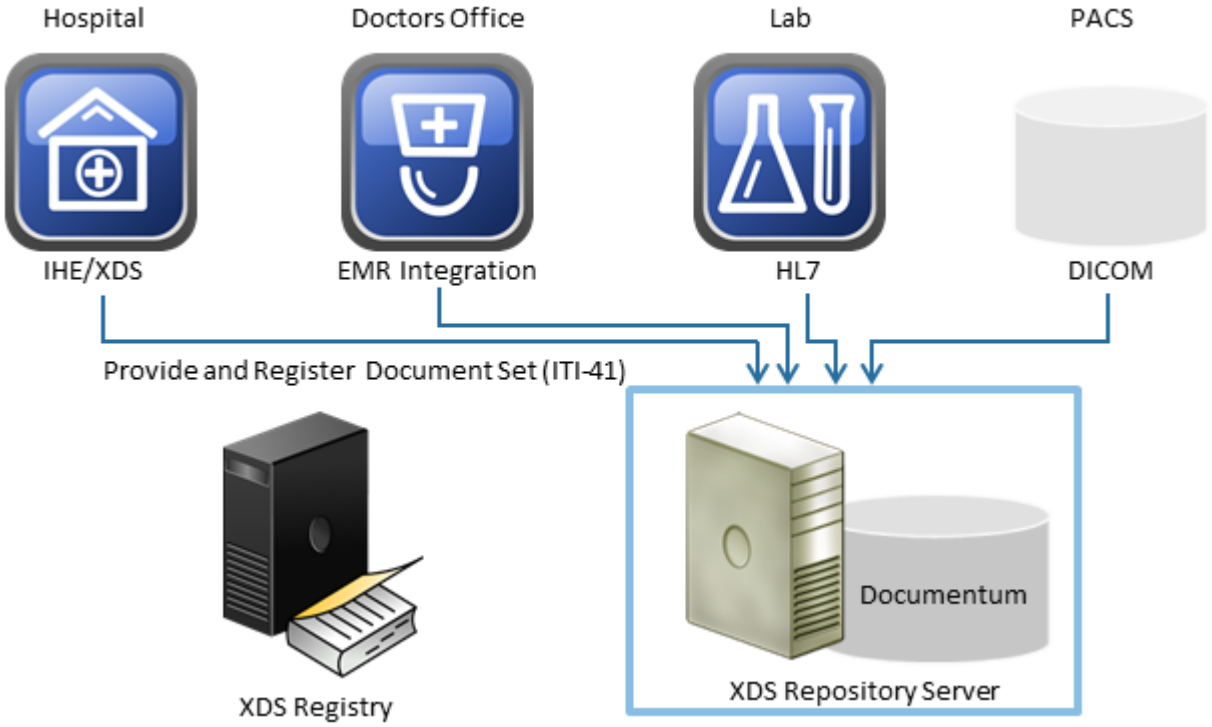
Workflow

A healthcare community consists of a variety of healthcare consumers and providers that need to access and share patient healthcare records. Healthcare records include administrative records (patient information) and patient medical records (X-rays, doctor reports, lab results).

There are many potential consumers and providers in a healthcare community. Some common examples are hospitals, physician's offices, labs, pharmacies, insurance companies, and Picture Archiving and Communications Systems (PACS).

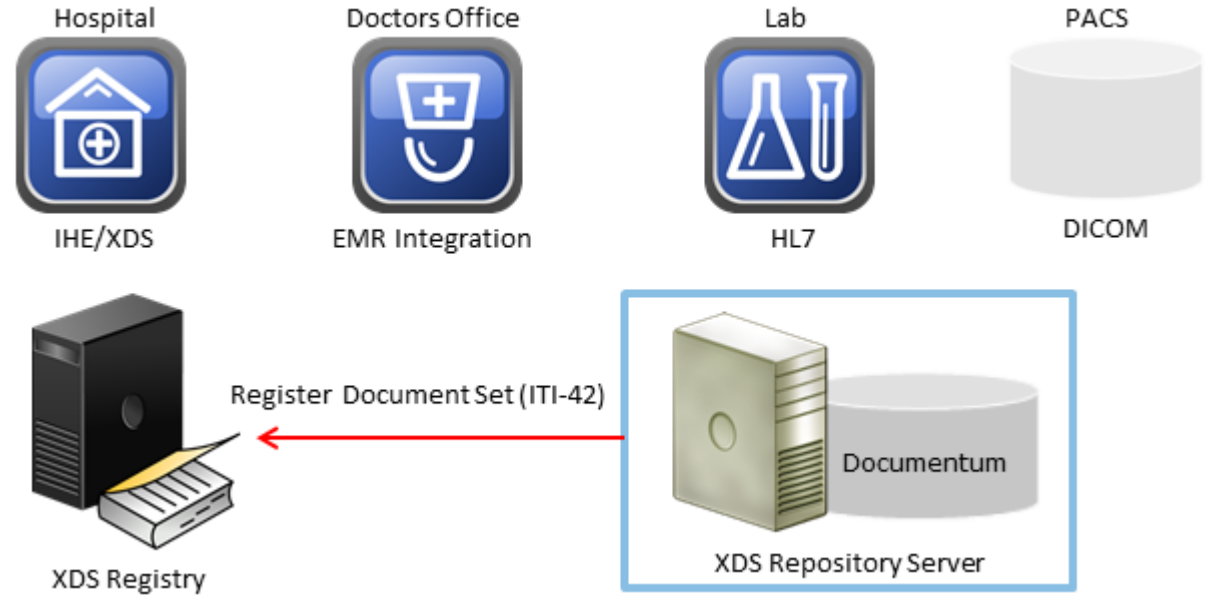
Healthcare providers use the `Provide and Register Document Set` transaction (ITI-41) to submit new healthcare records to the XDS Repository. The XDS Repository stores the submitted healthcare records in Documentum.

The following figure shows the `Provide and Register Document Set` transaction:



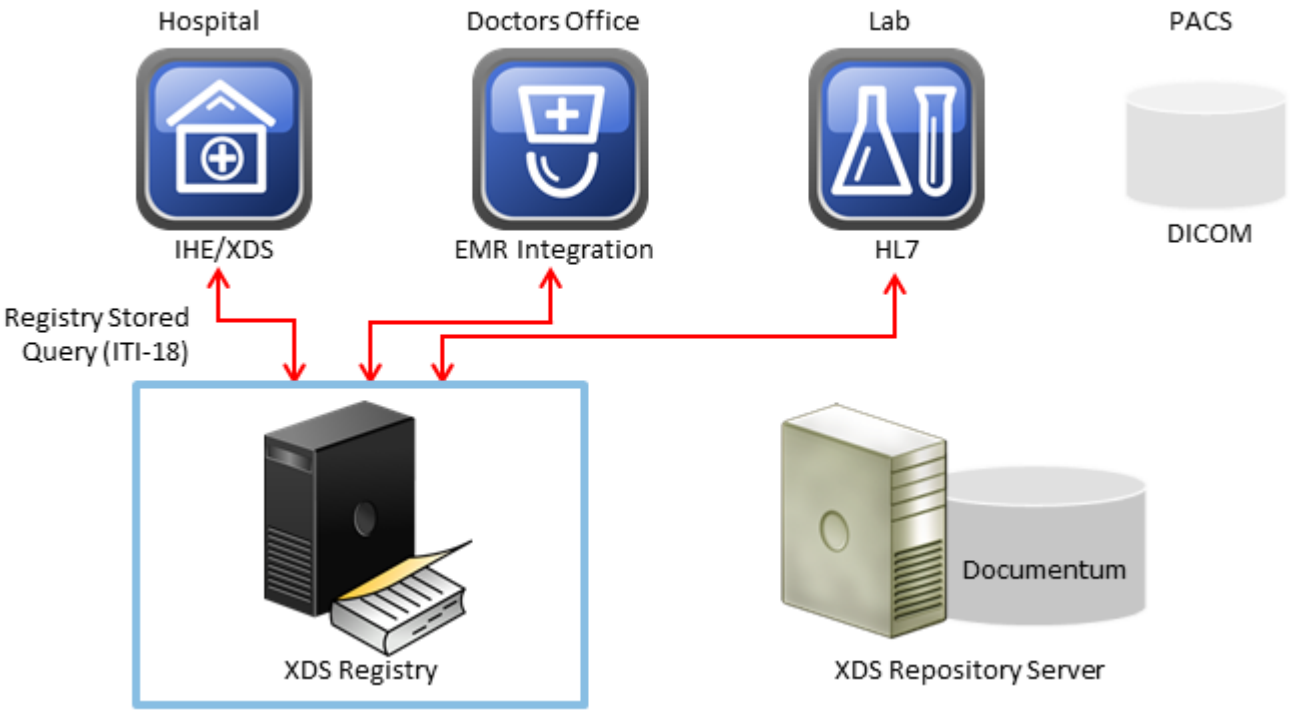
The XDS Repository Server then automatically registers the relevant document metadata with the XDS Registry. This is done using the `Register Document Set` transaction (ITI-42). Registering a healthcare record with the XDS Registry enables other healthcare providers to find the record.

The following figure shows the `Register Document Set` transaction:



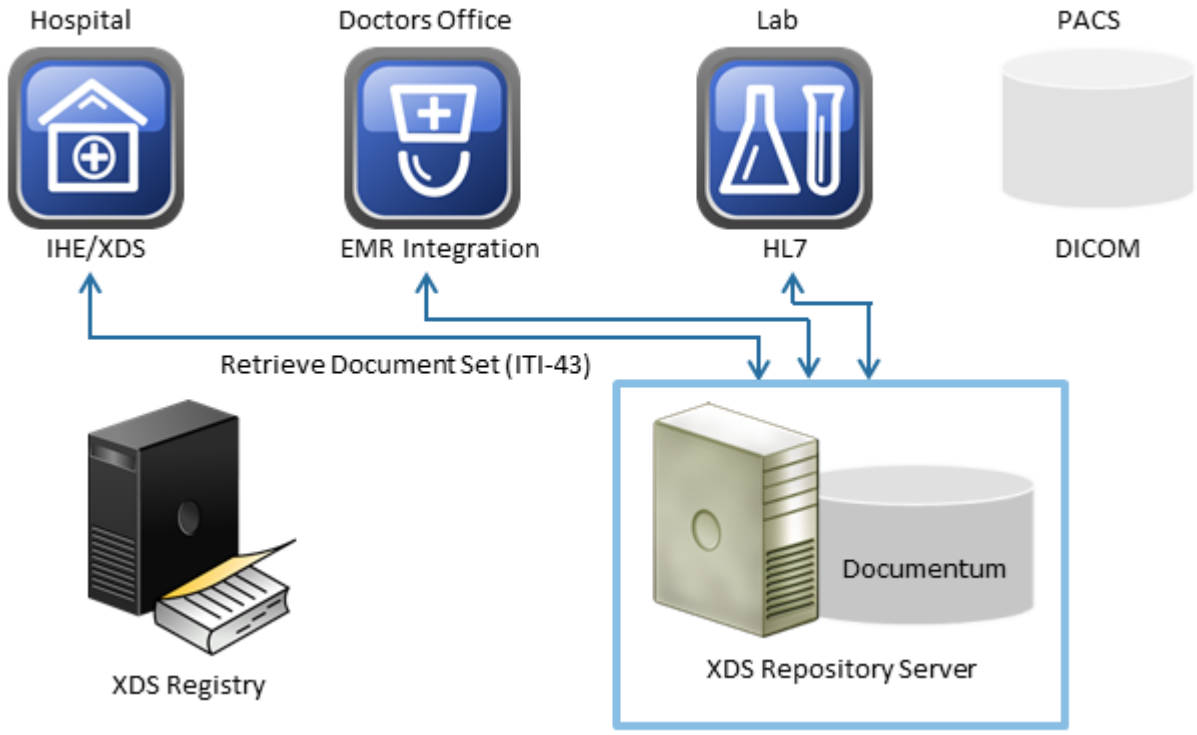
When healthcare providers need to obtain a patient's healthcare records, they query the Registry to get a list of healthcare records of that patient and their repository locations. The Registry is queried with the `Registry Stored Query` request (ITI-18). The Registry provides locations of the healthcare records.

The following figure shows querying the Registry with the `Registry Stored Query` request:



Document Consumers can then retrieve the documents from the Repository using the `Retrieve Document Set` transaction (ITI-43).

The following figure shows the `Retrieve Document Set` transaction:



Endpoints

The following table lists the endpoints for the XDS Repository Server:

IHE Transaction	Description	Endpoint
ITI-41	Provide and Register Document Set	https://<*host:port*>/cs-repository/services/xds-iti41
ITI-41as	Provide and Register Document Set Asynchronous	https://<*host:port*>/cs-repository/services/xds-iti41as

IHE Transaction	Description	Endpoint
ITI-43	Retrieve Document Set	https://<*host:port*>/cs-repository/services/xds-iti43
ITI-43as	Retrieve Document Set Asynchronous	https://<*host:port*>/cs-repository/services/xds-iti43as

Chapter 2

Features

This chapter contains the following topics:

Patient Identity Notifications

Healthcare providers maintain their own repository of medical records. HIP XDS Repository provides an XDS Repository that can also serve as a repository for the entire community. Patient records are categorized by patient identity. XDS Repository stores these patient records in separate folders for each patient using their patient identity and provides enterprise content management through Documentum.

ITI-8 Patient Identity Feed is the transaction in which a **Patient Identity Source** sends a message to the **Patient Identity Cross Reference Manager** and **Document Registry** whenever a patient is admitted, pre-admitted, registered, or when the patient demographic data is modified.

The XDS Repository Server supports only the Merge Patient Identity notification (ADT^A40). The Patient Identity Source generates the **Merge Patient–Internal ID** notification denoted by (ADT^A40) whenever two patient records are merged. Two records are merged when they reference the same patient in the Patient Identifier Domain.

The XDS Repository Server listens to the ITI-8 Patient Identity feeds through two Minimal Lower Layer Protocol (MLLP) ports, one secure and the other, non-secure. The HTTPS properties must be set if you want to enable secure HTTPS MLLP port. You must edit the `cs-repository.properties` file to configure these ports.

XDS Repository Transactions

The XDS Repository supports the following transactions:

- **ITI-41 Provide and Register Document Set-b**: Transaction [ITI-41] is used by a Document Source to provide a set of documents to the Document Repository, and to request that the Document Repository store these documents and then register them with the Document Registry.
- **ITI-42 Register Document Set**: Transaction [ITI-42] is used by a Document Repository Actor to register a set of documents with the Document Registry in XDS.b.

  The Document Repository submits document metadata to a Document Registry. The Document Registry receives and stores document metadata.
- **ITI-43 Retrieve Document Set**: Transaction [ITI-43] is used by a Document Consumer to retrieve a set of documents from the Document Repository, On-Demand Document Source, or Initiating Gateway.

HIP Customizations

Customization for Message Routes

By default, HIP provides a groovy file where the default HIP SOAP routes are defined.

The XDS Repository Server permits you to create your own groovy file and customize the external SOAP routes to filter or validate incoming requests and outgoing responses. After customizing the SOAP routes, you must configure the `soap.routeBuilderScriptSource` property in the `cs-repository.properties` to point to the customized groovy file that you created so that you can override the default groovy file.

If you do not configure the `soap.routeBuilderScriptSource` property with the location of your custom groovy file, the default groovy file is taken.

The following shows an example of customizing a soap route:

Default soap route:

```
// Retrieve Document Set (ITI-43)
        from('xds-iti43:xds-iti43' + options)
            .process(iti43RequestValidator())
            .to('direct:retrieveDocumentSet')
```

Customized soap route:

```
// Custom Retrieve Document Set (ITI-43)
        from('xds-iti43:xds-iti43' + options)
            .to('direct:customiRetrieveDocumentSetProcessor')
            .choice().when() { isConsentAuthorized(it) }
               .to('direct:retrieveDocumentSet’')
               .process(iti43ResponseValidator())
            .end()
```

Configuring the Custom SOAP Routes, page 79 provides information about configuring the soap route builder.

Custom Extension for Document Creation

The XDS Repository Server provides hooks for the Provide and Register request processing. These hooks allow you to customize the document creation process, so that you can define custom Documentum object types and attributes for newly created XDS Documents, folder location, ACL permissions and so on.

You can also configure the default Documentum object type and attributes used for PnR.

The `Provide and Register Document Set` transaction (ITI-41) is used by a Document Source to provide a set of documents to the Document Repository, and to request the Document Repository to store these documents and then register them with the Document Registry.

A PnR transaction carries:

- Metadata describing one or more documents
- Within metadata, one `XDSDocumentEntry` object per document
- XDS Submission Set definition along with optional folder and any related associations
- Zero or more XDS Folder definitions along with linkage to new or existing documents
- Zero or more documents

Whenever an `XDSDocumentEntry` object is added to the XDS Repository through PnR transaction, a document is created in the XDS Repository. The Document Creation Extensions feature allows you to alter the way the documents are created in the Documentum by means of custom extensions. This is done by dynamic assignment of object types and attributes and other processes irrespective of the Document Source that creates the XDS Documents.

The following shows examples of a few methods that can be overridden:

```
public class CustomCsContentObjectHandler extends DefaultCSContentObjectHandler{
    public CustomCsContentObjectHandler(CSRepositoryConfig config) {
        super(config);
    }

//For MIME TYPE
    @Override
    protected String resolveMimeType(IDfSession session, Context context)
    {
        String mimeType = "text/plain";
        return mimeType;
    }
//For ACL
   @Override
    protected ACLDefinition resolveACL(IDfSession session, Context context) throws DfException
    {
        return new ACLDefinition("test_acl", "test");
    }

//FOR CUSTOM PROPERTIES
    @Override
    protected void setObjectProperties(IDfSession session, Context context,
                                       IDfSysObject object, String mimeType) throws DfException
```

```
    {
        object.setString("mime_type", mimeType);
        object.setString("availability_status", "Approved");
    }

//FOR  Object Type
    @Override
    protected String resolveObjectType(IDfSession session, Context context, String objectType)
    {
        String contentType = "dm_note";
        return contentType;
    }
}
```

Configuration to Support PnR Custom Extensions, page 84 provides steps to provide custom extensions.

Custom HL7 Message Processing

By customization, you can enable HIP Repository to process HL7 messages that are not supported out-of-the-box (OOTB) by HIP Repository.

If you want to process an HL7 message that is not supported out of the box by the Repository, you must create and inject new routes that can process the additional HL7 messages. You also must instruct the HIP Repository to forward the additional HL7 messages to the newly created routes for processing.

By default, HIP Repository rejects message types that are not supported out-of-the-box or that are not specified by the customers. The current version of HIP enables you to process unknown messages. You can enable HIP Repository to process unknown messages by defining a custom spring bean and adding a new messagetoroute map with entry key configured as **`defaultRoute`** and value configured to any custom route that can process unknown messages.

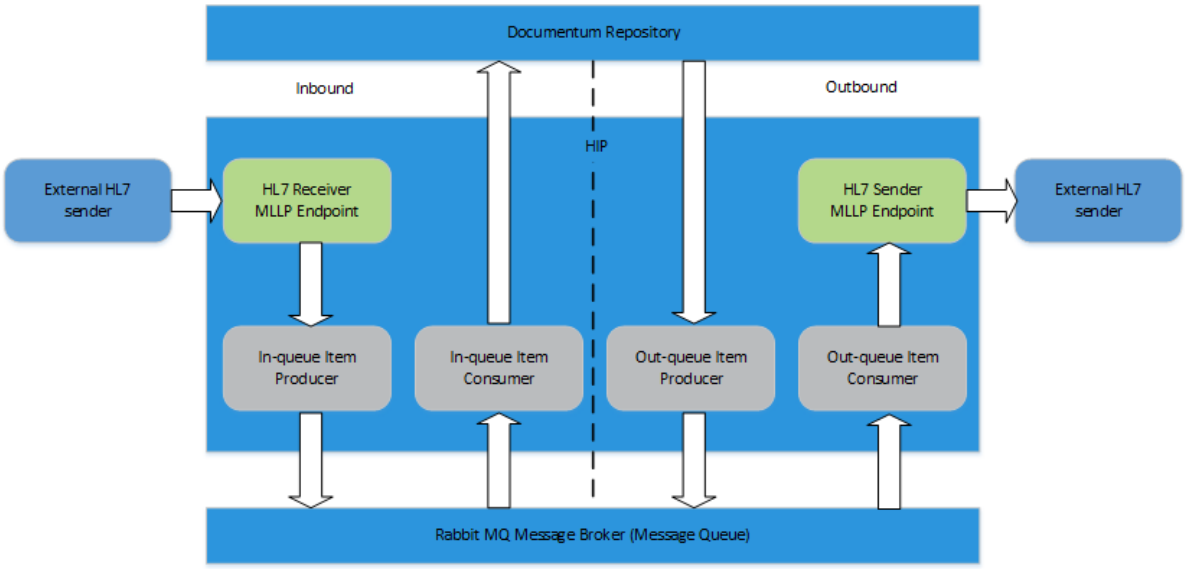
Configuring the Repository Server Context Extensions, page 82 provides details on enabling HIP Repository to process additional HL7 messages and unknown messages.

Message Queue

The XDS Repository Server utilizes a Message Queue, which enables the configuration of inbound and outbound message types to be processed. The Message Queue also supports asynchronous processing of each message.

The Message Queue is configured to process specific inbound and outbound messages for the HIP XDS Repository. However, additional custom message types can also be configured for the Message Queue, which can then be processed by custom message processors.

The following figure shows the working of a Message Queue:



The Message Queue supports the following features:

- Publishing and subscription of messages from message queue
- Re-delivery and support for dead letter queue in HIP

You can enable the Message Queue feature by configuring the `Hl7.queue.enabled` property in `hl7.properties` file.

Custom HL7 Message Processing, page 20 provides information on support for custom HL7 messages.

Configuring the HL7 Message Queue Properties, page 61 provides details on the Message Queue properties that you must configure.

HL7 Messages

HL7 messaging standards define how information is packaged and communicated from one system to another. Such standards specify the language, structure, and data types required for seamless integration of information from one system to another.

Inbound Messages

The XDS Repository supports the MergePatientIdentity notification and simple MDM messages for new document notifications.

HL7 Versions supported for inbound messages

`ADT^A34(V23),ADT^A40(V231,V24,V25,V251)`

`MDM T02 (V23,V231,V24,V25,V251)`

Some clinical systems send medical records in the OBX segment of HL7 MDM T02 message. HIP Repository receives MDM T02 message, extracts medical record from the OBX Segment and stores it as instance of `dmh_document` type. MDM T02 OBX-2 segment supports text (TX) data, formatted (FT) data, and encapsulated data (ED). In OBX-5 segment, multi-part MDM T02 messages support ASCII and BASE-64 encoding.

Customization to Support HL7 MDM T02 Inbound Messages Processing, page 83 provides details on the configuration to enable HL7 MDM T02 inbound message processing.

HL7 Inbound Message Queue

The XDS Repository Server provides an Inbound Message Queue to enable HIP to store inbound messages and send positive acknowledgement to the external systems before all messages are actually processed.

If the Message Queue is enabled, the ADT Merge messages are routed through the Message Queue. However, the MDM T02 messages are not routed through the Message Queue. They are directly processed by the queue items.

HL7 In Queue Item Object Type

The healthcare `dmh_hl7_in_queue_item` object type stores inbound HL7 Merge Patient Identities requests. The XDS Repository Server creates the records in this table when it accepts HL7 Merge Patient Identities requests. The record is processed by the Documentum Job and Method Framework.

For each request, the server updates the Retiring Patient record by moving it to the Surviving Patient records folder and merging it with the Surviving Patient identity. The server processes each request in the order received and rejects duplicate merge requests. The HIP Healthcare Information Model installs the job and method framework required for HL7 Merge Patient Identities requests. Use Documentum Administrator to configure the HIP Patient Merge Job properties.

- **Name**: dmh_hl7_in_queue_item
- **Supertype**: dm_sysobject
- **Object type tag**: 0b
- **Indexes**: none

The HL7 In Queue Item table provides the following properties. All properties occur once in each folder.

Name	Type	Description
completion_status	Integer	The completion status can be: • 0— To be processed • 1— In progress • 2— Success • 3— Failed
creation_time_double	Double	Item creation time.
execution_start_time	Time	The time when item execution starts.

Name	Type	Description
execution_end_time	Time	The time when item execution finishes.
item_arguments	String (64)	Not currently used.
message_type	String (32)	The message type. For example, HL7_ADT_MERGE.
message_control_id	String (20)	The message unique control ID.
process_id	String (64)	Not currently used.
retiring_patient_id	String (64)	The retiring patient ID.
surviving_patient_id	String (64)	The surviving patient ID.
status_description	String (1024)	The status description detailed message.
sending_application_id	String (185)	The ID of the sending application that sends HL7 messages.
sending_facility_id	String (185)	The ID of the sending facility that sends HL7 messages.

Outbound Messages

The XDS Repository Server supports sending outbound MDM messages to inform external HL7 systems about the newly added content and cancelled content.

HL7 Versions supported for outbound messages

`V23,V231,V24,V25,V251`

Supported Message Type/Event:

`MDM_T01,MDM_T02, MDM_T11`

`DEFAULT =2.3.1`

HL7 Outbound Message Queue

The XDS Repository Server provides Outbound Message Queue to enable storing outbound MDM messages and failed messages. If the external systems fail to receive outbound HL7 v2 messages due to network issues, the failed messages are moved to a 'Failed' Message Queue. HIP components can try resending the messages at a later time.

If the outbound message delivery is successful, HIP Repository receives an acknowledgement from the external systems and the status of message delivery is updated in the `dmh_hl7_out_queue_item`. This queue is used to maintain the status of all messages that are sent. Depending on whether the message sending is success or failure, HIP updates the status.

HL7 Outbound Message for Connector for Epic EOB

EMC's Connector for Epic enables users to store scanned Explanation of Benefits (EOB) documents issued by the payer in Documentum in a file cabinet called `<Payer cabinet>`.

Whenever an Epic user creates an EOB, it gets stored in the Documentum repository.

Configuring the HL7 EOB Message for C4E, page 62 provides details about the configuration of HL7 EOB messages.

HL7 Out Queue Item Object Type

The XDS Repository Server can send HL7 messages to external systems notifying them about new documents in the repository. The XDS Repository Server sends MDM T01, MDM T02 and MDMT11 messages with the content object ID as the unique document identifier.

The healthcare `dmh_hl7_out_queue_item` object type stores all outbound MDM T01, MDM T02 and MDMT11 message requests. Applications that send healthcare documents to the repository must create a queue item in this object type. The record must contain a healthcare document object ID. The XDS Repository Server periodically looks for items in this table, processes them, and sends MDM T01, MDM T02 and MDMT11 messages to external HL7 systems.

- **Name**: dmh_hl7_out_queue_item
- **Supertype**: dm_sysobject
- **Object type tag**: 0b
- **Indexes**: none

The HL7 Out Queue Item table provides the following properties. All properties occur once in each folder.

Name	Type	Description
content_object_id	String (32)	The healthcare document object ID.
completion_status	Integer	The completion status can be: • 0—To be processed • 1—In progress • 2—Success • 3—Failed
creation_time_double	Double	The item creation time.
item_arguments	String (64)	The HL7 message field values.
message_type	String (32)	The message type, for example HL7–MDM_T01.
process_id	String (64)	The unique ID for the processing of HL7 outbound requests.
retry_count	String (185)	Not currently used.
status_description	String (1024)	The status description detailed message.

Documentum Integrations

Documentum Object Types, Attributes, and Folders

HIP XDS Repository Server uses Healthcare Information Model (HIM) for Documentum by default. HIM is the central component of the Documentum healthcare repository that provides a single unifying data model for storing, retrieving, and managing all healthcare-related documents and information in Documentum.

HIP Repository supports mapping the Documentum object types and attributes to the following IHE object types and attributes:

- Healthcare document object type and attributes
- Repository folder location (defaults to `/Patients` cabinet)
- Patient folder object type and attributes

HIP Repository provides a file called `hip-dctm-mapping.properties` file that you can use to specify user-specific object types and attributes. Repository then maps the object types according to the configuration given by the user.

Configuring the HIP Documentum Mapping Properties, page 85 provides details about configuring the `hip-dctm-mapping.properties` file to use user-specific object types.

Documentum Mime Types

HIP XDS Repository Server provides a file called `mimetype.properties` to configure mime type to Documentum format name mapping.

Repository follows the order mentioned below to determine the Documentum format to be used for an incoming document:

1. The `mimetype.properties` file is consulted to check if a Documentum format name exists for the mime type of the incoming document. If yes, the format name is selected from the `mimetype.properties` file.
2. The format object in Documentum has an attribute called `mime_type`. Repository checks if any one of the values of this attribute matches with the mime type of the incoming document. If yes, the matching format name is selected from the format object.
3. If the XDS Repository Server is not able to find the mime type in both `mimetype.properties` file and Documentum format object, the format is set to `binary`.

The topic mimetype.properties, page 117 provides a sample `mimetype.properties` file.

Documentum XDS Queue

The Documentum XDS Queue is a Documentum-based queue for initiating XDS processing on documents in the Documentum repository.

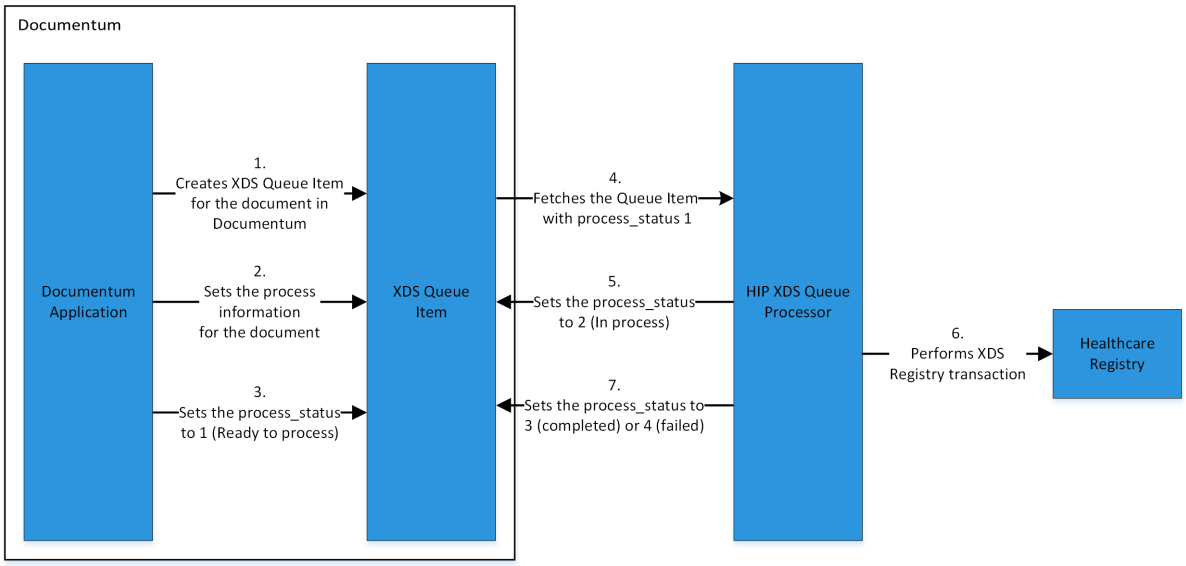
The Documentum XDS Queue currently supports the following XDS processes:

- Registering documents with the XDS Registry
- Deleting XDS Registry entries
- Deprecating XDS Registry entries

The `dmh_xds_queue_item` object type stores the registration information, delete registration, and deprecate registration information of patient health records.

The HIP XDS Queue processors periodically query the Documentum XDS queue (`dmh_xds_queue_item`) and process the individual queue items accordingly, to perform the XDS processes.

The following diagram shows the Documentum XDS Queue Process:



XDS Queue Item for Registration, Delete Registration and Deprecate Registration

The healthcare `dmh_xds_queue_item` object type stores the registration and delete information of patient health records.

For the registration process, the documents referenced by `dmh_xds_queue_item` are submitted to the XDS Registry for registration using the XML Registration data contained in the `dmh_xds_queue_item` content.

For the delete registration process, all document entries in the XDS Registry which has the document unique ID (`dmh_xds_queue_item.document_unique_id`) and any associations are removed.

For the deprecate registration process, the document entry in the XDS Registry which has the document unique ID (`dmh_xds_queue_item.document_unique_id`) are marked for deletion.

- **Name**: dmh_xds_queue_item
- **Supertype**: dm_document

- **Object type tag**: 09
- **Indexes**: none

The `dmh_xds_queue_item` has the following properties. All properties occur once in each folder.

Name	Type	Description
document_object_id	String (32)	The object ID of the document to be registered. This property is not set for delete registration process.
document_unique_id	String (64)	The unique ID of the document to be deleted. This property is not set for the registration process.
process_id	String (128)	The unique ID for each batch process. The `process_id` will be set to a unique ID for each batch by the poller.

process_type	String (64)	The type of the process.
		
		For registration process:
		`register`
		For delete registration process:
		`delete`
		For deprecate registration process:
		`deprecate`

process_status	Integer	The status of the process: • 0—Not ready to process • 1—Ready to process • 2—In process • 3—Process completed • 4—Process failure
process_error	String (1024)	The processing error information.
patient_id	String (256)	The ID of the patient. The Patient ID field is not used for validation purposes.
register_patient	Boolean	If set to **`True`**, the patient identity provided in `dmh_xds_queue_item.patient-id` is registered prior to registering the patient document. The `patient_id` property must be configured if this property is set to **`True`**.

XDS Queue Item for Registration

The 'register' XDS Queue Item is used to register a Documentum (non-XDS) document with the XDS Registry. Any Documentum application or process can create a 'register' XDS Queue Item for the HIP XDS Queue Item Processor to perform the registration.

During the registration process, the XDS submission set and registration information of documents referenced by `dmh_xds_queue_item` are submitted to the XDS Registry using the XML Registration data contained in the `dmh_xds_queue_item` content. The registration is then performed by the XDS Queue Item Processor by submitting the `ITI-42 RegisterDocumentSet` SOAP request to the configured XDS Registry Server endpoint in `registry.registerDocumentSetUrl`.

To ensure successful registration, the Registration XML that is created when a document is imported, must adhere to the XDS registration schema and must have all the mandatory metadata attributes configured appropriately. Also, you must ensure that the `patient_id` property is configured and the `register_patient` property is set to true in the `dmh_xds_queue_item` table.

Each metadata attribute has a specific meaning defined by the IHE standards. Therefore, the metadata configuration must be according to the definition provided by the IHE standards. Incorrect configuration of metadata attributes leads to registration failure.

Configuring the Registry Properties, page 69 provides details on configuring the `Register Document Set` properties.

Configuring the XDS Queue Item Processor Properties, page 72 provides details on configuring the registration polling properties.

XDS Registration XML, page 118 provides a sample content of XDS Registration XML file.

XDS Registration Schema, page 121 provides an example of registration schema.

XDS Queue Item for Delete Registration

The 'delete' XDS Queue item is used to delete a document registration from the XDS Registry for a Documentum document.

Any Documentum application or process can create a "delete" XDS Queue Item for the HIP XDS Queue Item Processor to perform the delete-registration.

During the delete registration process, the document entry in the XDS Registry, all submission sets and all associations that reference the document unique id (`dmh_xds_queue_item.document_unique_id`) are removed. The Delete Registration is then performed by the XDS Queue Item Processor by submitting the `ITI-62 DeleteDocumentSet` SOAP request to the configured XDS Registry Server endpoint in `registry.deleteDocumentSetUrl`.

The `Delete Document Set` properties are required only when the Queue based registration process and `Delete Document Set` transaction are enabled.

Configuring the Registry Properties, page 69 provides details on configuring the `Delete Document Set` properties.

XDS Queue Item for Deprecate Registration

The 'deprecate' XDS Queue Item is used to deprecate a Documentum document registration entry in the XDS Registry.

Any Documentum application or process can create an "deprecate" XDS Queue Item for the HIP XDS Queue Item Processor to perform the deprecate-registration.

During the deprecate registration process, the document entry in the XDS Registry with the document unique ID (`dmh_xds_queue_item.document_unique_id`) is deprecated. Unlike the delete registration process, deprecate registration process deprecates only the document entry and not the submission sets and associations related to the document entry. Deprecating a document entry only marks the document entry for deletion rather than actually deleting it. The deprecate registration is then performed by the XDS Queue Item Processor by submitting the `ITI-57 UpdateDocumentSet` SOAP request to the configured XDS Registry Server endpoint.

Configuring the Registry Properties, page 69 provides details on configuring the `Deprecate Document Set` properties.

EMR Integrations

Epic Integration

HIP supports integration of the Repository with Epic using Connector for Epic (C4E). HIP also supports registration and deregistration of documents created in Documentum through C4E.

For successful registration, you must ensure that the `patient_id` property is configured and the `register_patient` property is set to **`True`** in the `dmh_xds_queue_item` table.

The topic *XDS Queue Item for Registration, Delete Registration and Deprecate Registration* in the *EMC Documentum Healthcare Integration Portfolio Healthcare Information Model for Documentum* provides details about the properties of `dmh_xds_queue_item`.

Chapter 6, Epic Integration provides details about the configuration that you must perform to enable integration of repository with C4E.

Security

HIP provides the following security features:

- TLS or Trusted Node Authentication
- Patient Privacy Policy enforcement
- User Authentication through XUA
- Auditing

Chapter 3, Security Configuration provides more information about the security features.

Business Continuance

The information provided in this topic applies only to the XDS Repository Server and not to Documentum Content Server.

Refer the Documentum-product documentation for information about business continuance for Documentum.

Load Balancing and Scalability

HIP adopts the following methods for load balancing and scaling the environment:

- Using multiple XDS Repository instances. In this method, instantiate multiple instances of XDS Repository Server and use a Load Balancer to route incoming requests to the XDS Repository cluster. In the event of a failure, the Load Balancer routes requests to available XDS Repository Servers to ensure the system remains available.
- Configuring server connections to Documentum repositories. Each XDS Repository Server connects to a single Documentum repository, but a single Documentum repository can accept connections from multiple XDS Repository Servers. You can instantiate multiple XDS Repository Servers to serve the same or different Documentum repositories.

Data Backup and Recovery

Data backup and recovery strategies should focus on the Documentum repository. XDS Repository Server does not store XDS documents or transaction data; it only maintains the initial server configuration information. This information resides in the `cs-repository.properties` file and is typically backed up with the entire server installation. There is no Recovery Point Objective for XDS Repository Server because the server does not store transaction information or repository content. Follow recommended backup and recovery procedures for the Documentum Content Server.

High Availability and Disaster Recovery

High Availability Disaster Recovery (HADR) is achieved by implementing a backup strategy for each XDS Repository Server in the environment, including the Documentum Content Server.

For example:

- **Spare Instance**: Create spare instances of XDS Repository Server that you can manually start when an active XDS Repository Server fails. You can create a spare instance through VM images, ESXI servers, or other similar methods.
- **Active-Passive**: Configure a Universal Fail-Over Server (UFO) that is active and able to take over the functionality of the failing server.
- **Active-Active**: Configure multiple XDS Repository Servers to serve a single Documentum instance. If one server fails, the other servers remain available. The XDS Repository Servers may reside on different machines and different locations.

If the XDS Repository Server or Documentum Server fails and another server replaces it, the components may have difficulty connecting to the new server. Reconfigure the components to point to the new server or use a DNS alias for the server and simply update the alias.

The whitepaper *High Availability Configuration For a Multiple Region EMC Healthcare Integration Portfolio (HIP) Registry and Repository* provides more details on HADR.

Usage Reporting

Usage Reporting is implemented to support the subscription pricing model for customers. That is, the customers can be billed based on the usage of the product against the licenses purchased. Usage Reporting provides a monthly report on the usage of the product by the customer. The usage of Repository is calculated based on the number of documents stored and number of unique patients in the Repository.

Standard Usage Reports are scheduled to be automatically generated on the last day of each month. However, you can also generate adhoc reports.

A Usage Reporting web application provides the ability to view the monthly reports and to generate adhoc reports. The monthly reports are stored in the Documentum Repository as `dm_document` instances. However, adhoc reports that users generate are not stored in any location; you can only download them to your local system.

For the Usage Reporting web application to generate scheduled reports, you must create a folder `/System/HIP/Repository/Usage Reports` in Documentum. The scheduled Usage Reports are stored as `dm_document` with `hip_usage_report_acl` applied to them.

You can launch the Usage Reporting user interface by using the following URL:

```
localhost:port/cs-repository/reports
```

The default username and password to log in to the web application are configured in the `cs-repository.properties` file. You can modify the login configuration, if required.

When you log in to Usage Reporting, you can see a list of monthly reports that are generated. You can click a report to view the details.

Each report shows the following details:

- **Image Study Count**: Number of documents stored in the Repository.
- **Unique Patient Count**: Number of unique patients in the Repository.

The number of unique patients in the Repository is the count of `dmh_patient_folder` in the cabinet configured in the **patient.folder.location** property of `cs-repository.properties` file.

- **Generated on**: Date and time when the report is generated.
- **Generated By**: Name of the user who generates the reports.
- **Host**: IP of the host machine that generates the reports.
- **Type**: Name of the product for which the reports are generated.
- **Version**: Version of the product for which the reports are generated.

The **Generate Report Now** button enables you to create adhoc reports that shows the usage details from the day of product purchase.

The name of the reports are same as their IDs. The report ID consists of the timestamp when the reports are generated. That is, the ID of a report is of the format yyyyMMddHHmmss.

The Usage Reporting user interface also provides options to download the reports in XML, HTML, or PDF formats.

Configuring the Usage Report Properties, page 81 provides information on configuring the Usage Reporting properties.

Chapter 3

Security Configuration

This chapter contains the following topics:

Authentication Configuration

Trusted Node Authentication

The XDS Repository Server supports Trusted Node Authentication using TLS certificates.

HIP provides system security through SSL/TLS standards. System security provides access to HIP systems through data encryption and data confidentiality. TLS performs mutual authentication of client and server systems and encryption of data through certificates issued by the assigning authority.

The TLS keystore and truststore credentials are configured in the `cs-repository.properties` file. These credentials do not have any default value.

XDS Repository requires you to configure the connection credentials to the Documentum database where the healthcare information is stored. The connection credentials do not have default values.

The XDS Repository Server SOAP requests can be configured to be accessed through HTTP (non-secure), which allows these requests to be sent to the HIP XDS Repository Server without any trusted node authentication. This is optional.

User Authentication

The user authentication settings control the process of verifying an identity claimed by a user for accessing the product.

HIP provides user security through XUA.

System security authenticates systems, however, the server enterprise system is not aware of users defined on the client enterprise systems. XUA provides the ability to implement cross-enterprise user authentication to prevent data manipulation and provide data integrity. HIP systems provide XUA through WS-Security and WS-Trust standards.

The credentials for user authentication using SAML tokens and Security Token Server are configured in the `cs-repository.properties` file.

Access Control Settings

The access control settings enable the protection of resources against unauthorized access.

Configuring the ACL Property, page 69 provides details on configuring ACL used by HIP server to create documents.

Patient Privacy Policy Enforcement

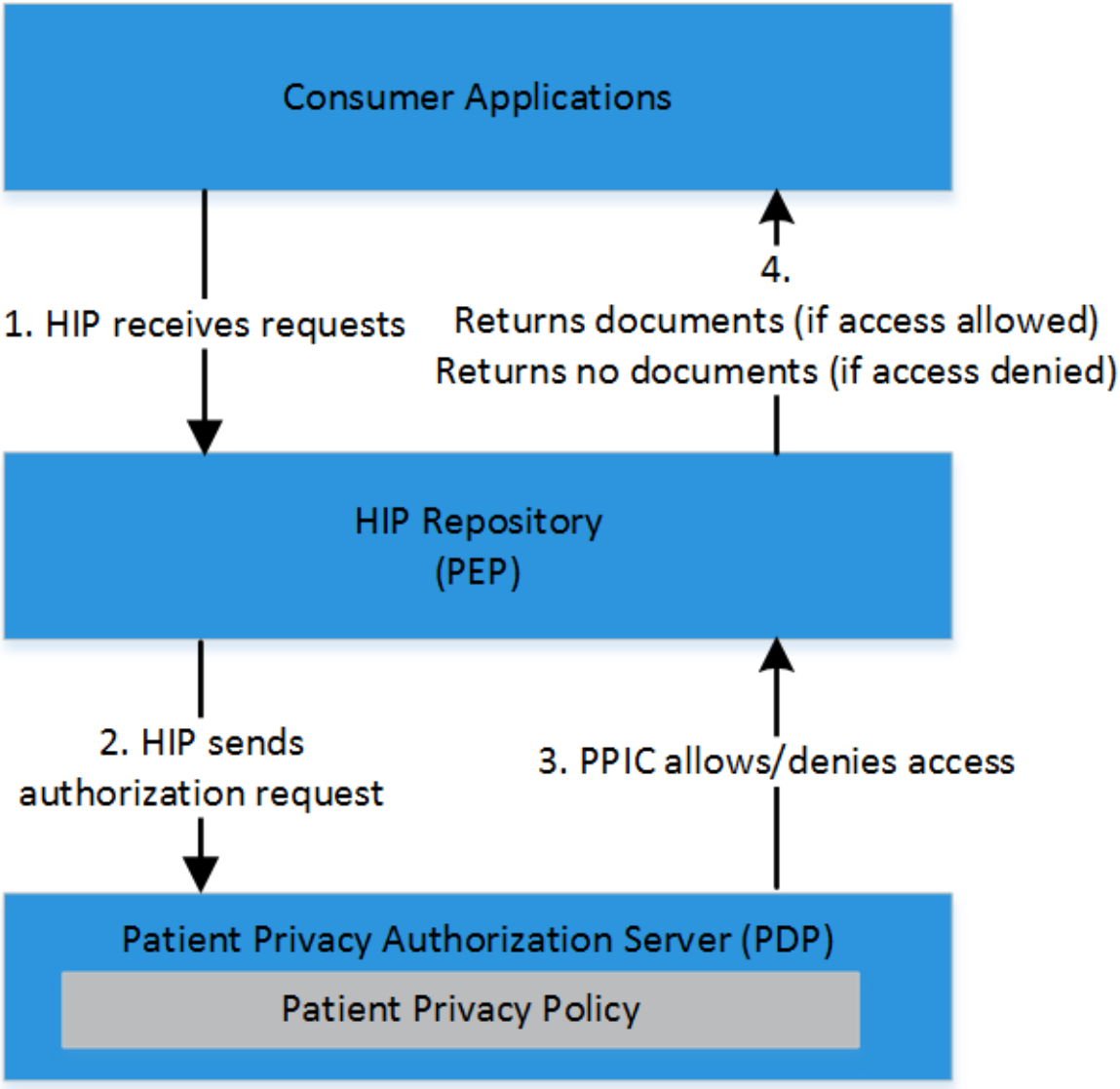
Users require authorization to access a patient's healthcare metadata and documents in an XDS Affinity Domain. A user is permitted to view only those documents for which permission is granted by the patient privacy policy. HIP implements the authorization feature through Patient Privacy and Informed Consent (PPIC) server. The PPIC server maintains the patient privacy policies that determine the users' rights to view patient documents. Patient Privacy Policy (PPP) enforcement enables XDS Repository Server to perform authorization for specific requests by user.

Whenever a Document Consumer sends a retrieve request to XDS Repository, the Policy Enforcement Point (PEP) component within XDS Repository, prepares a Policy Decision Request containing the Document UniqueIDs of documents for which authorization is required.

In an XUA enabled environment, the PEP component retrieves Subject ID and Subject Role information from SAML token. The PEP components passes these details in the Policy Decision Request to PPIC Server. PPIC Server evaluates all policies and performs lookup for additional metadata using XDB Registry PIP, if required. Based on the response received from PPIC Server, XDS Repository returns the document to the Document Consumer.

If the response from PPIC Server is 'Deny', no documents are returned.

The following figure shows the PPIC authorization flow:



This feature can be enabled or disabled according to the requirement.

Configuring the PPIC Server Properties, page 80 provides details on the PPIC configuration properties.

EMC Documentum PPIC Installation Guide provides details about the configuration of PPIC policies.

Communication Security Settings

The communication security settings enable the establishment of secure communication channels between the product components as well as between product components and external systems or components.

Port Usage

The XDS Repository Server endpoint URLs are configured for each request type, which includes the optional port. No default port is provided.

The ATNA audit server port default is 514 for ATNA SOAP requests.

XDS Repository Server creates audit messages for every request and response that it processes. The server prefixes audit messages with a unique audit source ID to easily identify the origin of the messages. The Audit Trail and Node Authentication Auditing (ATNA) properties in the `cs-repository.properties` file provide XDS Repository Server with connection information for the ATNA audit server.

The following table shows the ATNA port details:

Component	Service	Protocol	Port	Description
ATNA Audit Server	ITI-20 Audit Event.	SOAP	514	ATNA audit server

Configuring the ATNA Properties, page 75 provides details on ATNA configuration.

Network Encryption

XDS Repository Server uses the HTTPS protocol for HIP Server endpoints to achieve network encryption.

Data Security Settings

XDS Repository Server does not provide any data security settings, as all sensitive healthcare data is stored in the Documentum Repository. Documentum Content Server is responsible for securing this data internally.

Encryption of Data at Rest

The XDS Repository Server data is stored in Documentum, which is responsible for data encryption of the data at rest. The Documentum provides this ability OOTB.

Secure Deployment Settings

You must follow the instructions given below to securely deploy the product:

- Use HTTPS for all network communications with the XDS Repository Server SOAP requests.
- Configure TLS keystore and truststore credentials in the appropriate `<server>.properties` file.
- Use XUA for user authentication with a Secure Token Server. Configure the STS server credentials in the `<server>.properties` file.
- Use the HIP encryption password to encrypt sensitive user credentials.
- For the XDS Repository Documentum healthcare database, follow the Documentum recommended procedures to secure your Documentum healthcare database.

Chapter 4

Before You Install

Before beginning installation, ensure that your system meets the requirements.

The *EMC XDS Repository Connector for Documentum Release Notes* contains information on the system requirements for your product. This documentation is available from EMC Online Support.

Chapter 5

Installation

This chapter contains the following topics:

Pre-Installation Tasks

Before deploying XDS Repository Server, ensure that you have prepared the installation environment with the following components:

Component	Task
XDS Repository Installation Files	Download the following files from EMC Download Center: • Files for XDS Repository deployment: `hip-cs-repository-1.7.0.zip` • Files for deploying HIM Object types: `hip-dctm-1.7.0.dar` • Files for deploying XDS Object types: `hip-xds-1.7.0.dar` • Files for password encryption: `hip-encryptpassword-1.7.0.zip`
J2EE Web Application	Install either one of the following application servers: • Apache Tomcat 7 – 7.0.42 • Oracle WebLogic 12.1.1 or Oracle WebLogic 12.1.2

Component	Task
Registry	Install XDS Registry 1.7. You can use the EMC XDS Registry or any IHE enabled registry.
PPIC Server	Install PPIC Server 1.7, if PPIC is enabled for XDS Repository. *EMC Documentum PPIC Server Installation Guide* provides the instructions for installing and configuring PPIC Server.
Firewall Access	Provide XDS Repository Server with access outside the firewall for communicating with other healthcare community members. When configuring the XDS Repository, create a record of the ports you use and the direction in which you want to allow connectivity.
Documentum Content Server	Install Documentum Content Server 7.0 or 7.1.
Documentum Composer	Install the Documentum Headless Composer or the DAR Deployer to install the HIP Healthcare Information Model in the Documentum Repository.
Documentum Administrator (DA)	Install Documentum Administrator 7.0 or 7.1. This application enables you to update the custom job arguments in the Documentum repository.
Open eHealth Integration Platform (IPF)	Ensure that the IPF version is 2.6.3.
Apache Ant	Install Apache Ant 1.9.3.
JDK	Install JDK 1.7.x.
RabbitMQ	Install RabbitMQ 3.3.5 (www.rabbitmq.com) RabbitMQ server is required only if asynchronous HL7 inbound message processing is enabled.

EMC XDS Repository Connector for Documentum Release Notes provides a complete list of system requirements.

Creating a Documentum User Account for XDS Server

XDS Repository Server requires a user account to access the Documentum repository.

To create a Documentum User Account:

1. In the Documentum Content Server, create a Documentum user account named **`XdsServer`**.
2. Ensure that the `/Patients` cabinet exists in Documentum and the new user has unrestricted access and DELETE privileges on the `/Patients` cabinet in the newly created repository. The *EMC Documentum Administrator User Guide* provides the instructions for adding users.

Configuring the HIM Documentum Metadata Properties, page 72 provides details on cabinet configuration.

3. Note down the XDS Server user name and password for reference. You need these values for the next configuration step.

Deploying Object Types

The default object types used by HIP to install repository server are stored in the following DAR files:

- `hip-dctm-1.7.0.dar`—Stores the Documentum Healthcare Information Model object types
- `hip-xds-1.7.0.dar`—Stores the HIP XDS object types

If you are deploying `hip-dctm-1.7.0.dar` as `dmadmin` on a Windows 2012 machine, execute the DAR deployer with 'Run As Administrator' privilege to avoid getting the following post install failure error '`Unable to create dmh indices`'.

To deploy the DAR file:

1. Go to `<DCTM_install_dir>\product\7.1\install\composer\ComposerHeadless\`.
2. Double-click `dardeployer.exe`.
3. Define the following settings in the **DAR Details** and **Docbroker Details** section:
   - **DAR**: Click **Browse** and select the DAR file to be deployed.
   - **Docbroker Host**: Select the Docbroker host.
   - **Docbroker Port**: Define the port number for the Docbroker host.
4. Click **Connect**.
5. Define the following settings in the **Repository Details** section:
   - **Repository**: Select the Documentum Repository from the list.
   - **User Name**: Type the Documentum Content Server **User Name** that you have already created.

     Creating a Documentum User Account for XDS Server, page 42 provides the instructions to create a user.
   - **Password**: Type the password for the Documentum Content Server user name.
6. Click **Install**.

   The DAR Installer shows a message after you successfully install the DAR file.

   Deploy both the DAR files as mentioned above.

Installing Third-party Library Dependencies

You must install the following third-party library dependencies for successful deployment of the Repository Server WAR file:

- Camel
- HL7 Application Programming Interface (HAPI)
- IHE

Obtaining the Library Dependencies

To obtain the Camel jar files:

1. Create a folder in the local path to copy the Camel jar files.
   For example:
   `C:\jarfiles\camel`
2. Go to www.camel.apache.org.
3. Download the following files:
   For Windows:
   `apache-camel-2.12.1.zip`
   For Linux:
   `apache-camel-2.12.1.tar.gz`
4. Extract the zip file to a local path.
   For example:
   `C:\apache-camel-2.12.1`
5. Go to `<local path>\apache-camel-2.12.1`.
6. Copy the `lib` folder containing the jar files to `C:\jarfiles\camel`.
   For example:
   `C:\jarfiles\camel\lib`

To obtain the HAPI jar files:

1. Create a folder in the local path to copy the HAPI jar files.
   For example:
   `C:\jarfiles\hapi`
2. Go to www.sourceforge.net.
3. Download `hapi-dist-2.0-all.zip`.
4. Extract the Zip file to a local path.
   For example,
   `C:\hapi-dist-2.0-all`
5. Go to `<local path>\hapi-dist-2.0-all` folder.

6. Copy the `lib` folder containing the jar files to `C:\jarfiles\hapi`.
   For example:
   `C:\jarfiles\hapi\lib`

The HAPI jar files are required only if you want to integrate repository with Epic using C4E.

Due to licensing restrictions, HIP products do not deliver the required HAPI library jar files used when parsing HL7 feeds.

To obtain the IHE jar files:

1. Create a folder in the local path to copy the IHE jar files.
   For example:
   `C:\jarfiles\ihe`
2. Go to www.projects.openhealthtools.org.
3. Download the `org.openhealthtools.ihe_2.0.0.zip` file.
4. Extract the Zip to a local path.
   For example:
   `C:\openhealthtools\`
5. Go to `C:\openhealthtools`.
6. Copy the following jar files to the `C:\jarfiles\ihe` folder.
   - `org.openhealthtools.ihe.atna.context_2.0.0.jar`
   - `org.openhealthtools.ihe.atna.nodeauth_2.0.0.jar`
   - `org.openhealthtools.ihe.utils_2.0.0.jar`
7. Go to www.repo.openehealth.org.
8. Copy the following jar file to the `C:\jarfiles\ihe` folder.
   `org.openhealthtools.ihe.atna.auditor-2.0.0-p1.jar`

To verify the jar files:

1. Check if the checksum values of the downloaded jar files match with the checksum values mentioned in the following table for the corresponding file:

Filename	Checksum Value
apache-camel-2.12.1.zip	8aae88dec9558606cfd1b4bd6aaa2438
org.openhealthtools.ihe_2.0.0.zip	ac7bee0ac7d0db9becf71d29690d45d3
org.openhealthtools.ihe.atna.auditor-2.0.0-p1.jar	33c0fbc631ab539d70dd0bb6fd343fee
hapi-dist-2.0-all.zip	ee574cb7b458ea934a45d71a3c5da5e2

Installing the Library Dependencies

1. Download `hip-cs-repository-1.7.0.zip` from EMC Download Center.

2. Extract `hip-cs-repository-1.7.0.zip` to a local folder.

   For example:

   `C:\hip-cs-repository-1.7`

   You can find the `build.xml` file in the `C:\hip-cs-repository-1.7` folder.

3. Go to the command prompt and navigate to the directory where `build.xml` is located.

   For example:

   `C:\hip-cs-repository-1.7`

4. Run `build.xml` using the following command:

   `ant -f build.xml`

5. Type the complete path of the Camel home directory when the script prompts you to enter the Camel home directory.

   For example:

   `C:\jarfiles\camel`

6. Type the complete path of the HAPI home directory when the script prompts you to enter the HAPI home directory.

   For example:

   `C:\jarfiles\hapi`

7. Type the complete path of the OHT home directory when the script prompts you to enter the OHT home directory.

   For example:

   `C:\jarfiles\ihe`

After you run the `ant -f build.xml` command, you get the `install` folder as follows:

`C:\hip-cs-repository-1.7\install`

You can find the `hip-cs-repository-1.7.0.war` file in the `install` folder.

Creating the HIP Configuration Directory

The HIP servers maintain configuration information in a directory called `HIP configuration directory` that is located outside the WAR file. This design enables the users to easily upgrade to latest versions of the software by replacing the server WAR file.

By default, HIP uses the following directory:

`<user.home>/.hip`
where <*user.home*> is the home directory of the user who runs the XDS Repository Server.

For example:

`C:\Users\username\`

If this directory does not already exist, create a folder named **`.hip`** in your `user.home` directory by typing:

`.hip.`

Ensure that you place a dot at the end of the name. Windows removes the trailing dot.

The **com.emc.Healthcare.com** Java system property defines the location of the configuration directory. If you want to override the default location of the HIP server configuration information, override the **com.emc.Healthcare.com** system property when you start the J2EE Web Application container.

Use the following syntax:

`-Dcom.emc.Healthcare.home=<hip_config_directory>`

Deploying the Properties Files in HIP Configuration Directory

After creating the HIP configuration directory, copy the properties files to the HIP configuration directory.

To copy the property files to the HIP directory:

1. Go to the `install` folder obtained after running the build command described in Step 4.

   For example:

   `C:\hip-cs-repository-1.7\install`
2. Extract `hip-cs-repository-1.7.0.war`.
3. Go to `\hip-cs-repository-1.7.0\config\`.
4. Copy the `cs-repository` folder to `C:\Users\username\.hip\`.
5. Ensure that the following files are present in the `C:\Users\username\.hip\cs-repository` folder.
   - `cs-repository.properties`
   - `cs-repository-context-extension.xml`
   - `hip-dctm-mapping.properties`
   - `hip-ppic-mapping.properties`
   - `hl7.properties`
   - `mimetype.properties`
6. Copy `dfc.properties` from `<Documentum installation directory>\config` to `C:\Users\username\.hip\cs-repository`.

You must configure the property files mentioned above according to your environment, for successful installation.

Chapter 7, Post-Installation Configuration describes the configuration of property files required for successful installation.

Deploying the HIP Repository WAR File on Windows

Before deploying the WAR file, ensure that you have completed all configuration tasks described in Chapter 7, Post-Installation Configuration.

HIP supports the following:

- Deploying the HIP Repository War File Using Tomcat, page 48
- Deploying the HIP Repository WAR File Using WebLogic, page 48

Follow the procedure described for the application server you have installed.

Deploying the HIP Repository War File Using Tomcat

1. Stop the J2EE Web Application container.
2. Copy the XDS Repository Server WAR file to the following directory:

   `<tomcat_install_dir>/webapps/`
3. Rename the WAR file to `cs-repository.war`.
4. Start the J2EE Web Application container to expand the WAR file.

The examples in this guide are provided with the assumption that the WAR file is deployed in `<tomcat_install_dir>/webapps/`.

Deploying the HIP Repository WAR File Using WebLogic

Before deploying the WAR file, perform the following tasks:

- Set the following environment variable:

  ```
  Name:  DOMAIN_HOME

  Value:  ~\Oracle\Middleware\user_projects\domains\domain{domain used
  for deployment}
  ```

- Set the following System Properties in the WebLogic startup script:

  ```
  com.sun.xml.ws.spi.db.BindingContextFactory=com.sun.xml.ws.db.glassfish
  .JAXBRIContextFactory javax.xml.bind.JAXBContext=com.sun.xml.bind.v2
  .ContextFactory javax.wsdl.factory.WSDLFactory=com.ibm.wsdl.factory
  .WSDLFactoryImpl
  ```

- Set the HIP home location in the startup script to get the log files generated in the `<user.home>/.hip/` folder.

  Example:

  ```
  set JAVA_OPTIONS=%JAVA_OPTIONS% -Dcom.emc.healthcare.home=C:\Users
  \<username>\.hip
  ```

To deploy the repository war file:

1. Log in to the WebLogic Admin console.
2. Go to **base_domain** > **deployment**.
3. Click **Install**.
4. From **Install Application Assistant**, click the **upload your file** link in the Note.
5. From **Deployment Archive**, browse and select the `hip-cs-repository-1.7.0.war` file.
6. Click **Next**.
7. Select **Install as application**.
8. Click **Next**.
9. Click **Finish**.
10. Check if the deployment state of HIP Repository is Active.
    The Active state shows a successful deployment.

Deploying the HIP Repository WAR File in Linux

1. Log in as **root** user.
2. Copy `hip-cs-repository-1.7.0.war` to the following location:
   `$CATALINA_HOME/webapps`
3. Run the following command to change tomcat installation owner to **dmadmin**.
   `chown –R $CATALINA_HOME dmadmin:dmadmin`
4. Set **dmadmin** environment variables.
5. Set HIP Java options in `$CATALINA_HOME/bin/setenv.sh`.
   `JAVA_OPTS="-Xms512m -Xmx1g -XX:MaxPermSize=512m -Dcom.emc.healthcare.home=/home/dmadmin/.hip"`
6. As **dmadmin**, copy the `.hip` folder to the following location:
   `/home/dmadmin`
7. As root user, create tomcat `startup/etc/init.d/tomcat` setting the values appropriately.
   For example,

```
#!/bin/bash
# description: Tomcat Start Stop Restart
# processname: tomcat
# chkconfig: - 90 10

# Source function library
. /etc/rc.d/init.d/functions

CATALINA_HOME=/app/apache-tomcat-7.0.53
TOMCAT_OWNER=dmadmin

#Set Startup Options for HIP
#See $CATALINA_HOME/bin/setenv.sh
#Check they have been used using ps-ef|grep tomcat
```

```
case $1 in
start)
        echo "Starting tomcat under dmadmin account..."
        echo "Note:xDB Must be Running or HIP Registry startup will fail..."
        su - $TOMCAT_OWNER -c "$CATALINA_HOME/bin/startup.sh"
        ;;
stop)
        echo "Stopping tomcat..."
        su - $TOMCAT_OWNER -c "$CATALINA_HOME/bin/shutdown.sh"
        ;;
restart)
        echo "Restarting tomcat under dmadmin account..."
        su - $TOMCAT_OWNER -c "$CATALINA_HOME/bin/shutdown.sh"
        sleep 2
        su - $TOMCAT_OWNER -c "$CATALINA_HOME/bin/startup.sh"
        sleep 2
        ;;
status)
        status tomcat
        ;;
*)
        echo "Usage: $0 {start|stop|restart|status}"
        exit 1
        ;;
esac
exit 0
```

8. Change permissions and set Tomcat to auto-start on reboot.

```
chmod +x /etc/init.d/tomcat
chkconfig tomcat on
```

Chapter 6

Epic Integration

This chapter contains the following topics:

This optional step applies to users who want to integrate the repository with Epic using Connector for Epic (C4E).

After installing XDS Repository Server, perform all the steps listed in the *EMC Documentum Connector for EPIC Installation Guide* and then perform the following additional configuration steps:

Configuring the HIP Merge Job and Method Arguments

The XDS Repository Server queues inbound HL7 ADT Merge Patient Identities requests in *dmh_hl7_in_queue_item*. HIP merges patient records through the `HipPatientMergeJob` method. By default, the Documentum server runs this job every 30 minutes and performs patient record merge for all queued items.

To use the merge feature, you must use Documentum Administrator and configure the following custom arguments:

Argument	Description

-patientsCabinetPath	This argument specifies the path to the root cabinet that contains all the patient records. The patient merge job execution fails if this argument is not specified or if it is incorrectly specified.
	
	This argument has no default value.
	
	For example:
	
	`-patientsCabinetPath<space>`
	`/<Patientsfolder>`

Argument	Description
-sendEmail	This optional argument specifies whether to send an email with merge status to the user. If set to true, the email is sent. The default value is **`false`**. For example: `-sendEmail<space>false`
-userNameToSendEmail	This optional argument specifies the e-mail address of the user to whom the merge status is sent. The default value is the name of the repository owner. For example: `-userNameToSendEmail<space><user email address>`
-sendEmailOnMergeCompletion	This argument specifies whether an e-mail must be sent to the user when the merge operation is completed. For example: `-sendEmailOnMergeCompletion<space>true`
-userAccountToSendMergeStatusEmail	This argument specifies the user account to which the e-mail must be sent when the merge operation is completed. For example: `-userAccountToSendMergeStatusEmail<space>Administrator` Ensure that the Administrator e-mail address is set to a valid e-mail address and the Content Server is configured with a valid SMTP Server name.
-hipRepositoryACL	This argument specifies the ACL used to create the HIP documents. For example: `-hipRepositoryACL<space>newACL`
-deleteRetiringFolderAfterMerge	This argument specifies whether to delete the retiring folder when the merge operation is completed. By default, the retiring folders are deleted after merge operation. However, by setting this argument to false, you can choose not to delete the retiring folders. For example: `-deleteRetiringFolderAfterMerge<space>false`

Configuring the HL7 Properties

This topic describes the following configurations:

Configuring the HL7 Time Zone Property

The following table shows the configuration of HL7 time zone property:

Property	Description
hl7.message.timestamp .includeTZOffset	This property specifies whether to include time zone offset in Time Stamp fields. HL7 recommends using time zone offset in Time Stamp fields. Since this time zone information is optional, certain applications prefer not to include the time zone offset. For example: `hl7.message.timestamp.includeTZOffset=true`

Configuring the HL7 Inbound Properties

The XDS Repository Server looks for all inbound properties in the `hl7.properties` file defined in the server classpath.

The following table shows the configuration of HL7 inbound properties:

Property	Description
hl7.inbound.mllp.port	This property specifies the port to which the XDS Repository Server listens for inbound HL7 requests. XDS Repository Server does not accept inbound HL7 messages unless you define this property. The default value is **0**. For example: `hl7.inbound.mllp.port=0`
hl7.inbound.mllp.securePort	This property specifies the secure port to which the XDS Repository Server listens for inbound HL7 requests. If the value of this property is set to **0**, Repository will not listen for incoming messages. For example: `hl7.inbound.mllp.securePort=0`
hl7.inbound.rejectDuplicateMessage	This property specifies whether the XDS Repository Server must reject the duplicate HL7 inbound messages. If set to **True**, the server rejects duplicate HL7 inbound messages. The default value is **True**. For example: `hl7.inbound.rejectDuplicateMessage=true`
hl7.inbound.queueitem.folderPath	This property specifies the folder path to save the inbound queue item. For example: `hl7.inbound.queueitem.folderPath=/Temp`
hl7.inbound.queueitem.acl	This property specifies the ACL for the inbound HL7 queue item. For example: `hl7.inbound.queueitem.acl=hip_repository_acl`

Configuring the HL7 Outbound Properties

The XDS Repository Server looks for all outbound properties in the `hl7.properties` file defined in the server classpath. All the properties are required unless otherwise mentioned.

The following table shows the configuration of HL7 outbound properties:

Property	Description
hl7.outbound.message.version	This required property specifies the supported versions. XDS Repository Server can send the output messages in different HL7 versions. The default value is **`2.3.1`**. For example: `hl7.outbound.message.version=2.3.1`
hl7.outbound.mllp.host	This required property specifies the external HL7 server host where the XDS Repository Server sends the HL7 messages. The default value is **`localhost`**. For example: `hl7.outbound.mllp.host=localhost`
hl7.outbound.mllp.port	This property specifies the port number of the HL7 server host where the XDS Repository Server sends the HL7 messages. The default value is **`zero`**. If set to **`0`**, the server will not send MDM messages. For example: `hl7.outbound.mllp.port=0`
hl7.outbound.requestTimeout	This optional property specifies the request timeout value in milliseconds. The default value is **`3000`**. For example: `hl7.outbound.requestTimeout=3000`
hl7.outbound.queue.pollingInterval	This property specifies the polling interval in seconds. For example: `hl7.outbound.queue.pollingInterval=30`

Property	Description
hl7.outbound.queue .reProcessInProgressItems OlderThanDays	This optional property specifies the number of days the system waits before resetting "in progress" items to "to be processed". The default value is **1**. For example: `hl7.outbound.queue .reProcessInProgressItemsOlderThanDays=1`
hl7.outbound.sendingApplication .namespaceId	This property specifies the name space ID of the sending application. For example: `hl7.outbound.sendingApplication.namespaceId =`
hl7.outbound.sendingApplication .universalId	This property specifies the universal ID of the sending application. For example: `hl7.outbound.sendingApplication.universalId =`
hl7.outbound.sendingApplication .universalIdType	This property specifies the universal ID type of the sending application such as ISO or HL7. For example: `hl7.outbound.sendingApplication .universalIdType=HL7`
hl7.outbound.sendingFacility .namespaceId	This property specifies the namespace ID of the sending facility. For example: `hl7.outbound.sendingFacility.namespaceId=`
hl7.outbound.sendingFacility .universalId	This property specifies the universal ID of the sending facility. For example: `hl7.outbound.sendingFacility.universalId=`
hl7.outbound.sendingFacility .universalIdType	This property specifies the universal ID type of the sending facility such as ISO or HL7. For example: `hl7.outbound.sendingFacility .universalIdType=HL7`

Configuring the HL7 Render Empty Segment Properties

The following table shows the configuration of HL7 render empty segment properties:

Property	Description

hl7.mdm.T01.renderEmptySegments
 	
hl7.mdm.T02.renderEmptySegments
 	
hl7.mdm.T11.renderEmptySegments
	This property specifies whether the MDM message contains a segment or not.
	
	The format of the property and its value is:
	
	`<property>=<segmentName>-<fieldSequence>`
	
	The recommended value for <fieldSequence> is 1.
	
	For example:
	
	`hl7.mdm.T01.renderEmptySegments=PV1-1`
	
	`hl7.mdm.T02.renderEmptySegments=PV1-1`
	
	`hl7.mdm.T11.renderEmptySegments=PV1-1`
	
	If
	
	`hl7.mdm.T01.renderEmptySegments =<segmentName>-<fieldSeq>`
	and the segment has NO value, then the outgoing MDM will have a segment with no value.
	
	If
	
	`hl7.mdm.T01.renderEmptySegments =<segmentName>-<fieldSeq>`
	and segment HAS A value, then the outgoing MDM will have a segment with the value of the segment.
	
	If
	
	`hl7.mdm.T01.renderEmptySegments=NONE`
	and segment HAS NO value, then the outgoing MDM will NOT have any segment.
	
	If
	
	`hl7.mdm.T01.renderEmptySegments=NONE`
	and segment HAS A value, then the outgoing MDM will have a segment with the value of the segment.

Configuring the HL7 MDM Redelivery Properties

HIP tries to redeliver the outbound message if the message delivery is a failure in the first attempt due to network issues.

The following table shows the configuration of HL7 MDM redelivery properties:

Property	Description
hl7.outbound.queue.maxRedeliveries	This property specifies the number of times HIP tries to redeliver the hl7 outbound message if the message delivery is a failure in the first attempt. If this property is set to 0, HIP will not try to redeliver the message. The default value is **3**. For example: `hl7.outbound.queue.maxRedeliveries=3`
hl7.outbound.queue.reDeliveryDelay	This property specifies the delay between each redelivery of message in seconds. For example, if this property value is set to 60 seconds, it indicates that HIP will wait for 60 seconds to resend the message, when the message delivery fails for the first time. The default value is **60 seconds**. For example: `hl7.outbound.queue.reDeliveryDelay=60`

Configuring the HL7 MDM Mapping

The following table shows the configuration of HL7 MDM mapping:

Property	Description

HL7 MDM T01 Mapping	`hl7.mdm.T01.MSH-7=import_date`
	
	`hl7.mdm.T01.MSH-6=station_name`
	
	`hl7.mdm.T01.PID-7=patient_birth_date`
	
	`hl7.mdm.T01.PID-18=csn_number`
	
	`hl7.mdm.T01.PID-2=NONE`
	
	`hl7.mdm.T01.PID-3=patient_id`
	
	`hl7.mdm.T01.PID-8=patient_sex`
	
	`hl7.mdm.T01.PID-5=patient_name`
	
	`hl7.mdm.T01.PV1-3=healthcare_facility_code`
	
	`hl7.mdm.T01.PV1-19=account_number`

Property	Description
	hl7.mdm.T01.TXA-2=record_type hl7.mdm.T01.TXA-4=admission_date hl7.mdm.T01.TXA-5=provider_mcr hl7.mdm.T01.TXA-6=investigation_date hl7.mdm.T01.TXA-17=completion_status hl7.mdm.T01.TXA-12=r_object_id hl7.mdm.T01.TXA-16=r_object_id
HL7 MDM T02 Mapping	hl7.mdm.T02.MSH-7=r_creation_date hl7.mdm.T02.MSH-6=station_name hl7.mdm.T02.PID-7=patient_birth_date hl7.mdm.T02.PID-18=csn_number hl7.mdm.T02.PID-3=patient_id hl7.mdm.T02.PID-2=NONE hl7.mdm.T02.PID-8=patient_sex hl7.mdm.T02.PID-5=patient_name hl7.mdm.T02.PV1-3=healthcare_facility_code hl7.mdm.T02.PV1-19=account_number hl7.mdm.T02.TXA-2=dmh_record_type.record_type hl7.mdm.T02.TXA-4=admission_date hl7.mdm.T02.TXA-5=provider_mcr hl7.mdm.T02.TXA-6=investigation_date hl7.mdm.T02.TXA-17=completion_status hl7.mdm.T02.TXA-16=r_object_id hl7.mdm.T02.TXA-12=r_object_id hl7.mdm.T02.OBX-5=r_object_id hl7.mdm.T02.NTE-3=

Property	Description

	The `hl7.mdm.T02.NTE-3` attribute can have any one of the following values:
	• **title**: Configure as follows if the document description is Free Form:
	`hl7.mdm.T02.NTE-3=title`
	• **dmh_record_type.subtype**: Configure as follows if the document description is categorized and retrieved from a drop-down list:
	`hl7.mdm.T02.NTE-3=dmh_record_type.subtype`
	• **title|dmh_record_type.subtype**: Configure as follows if you want the MDM Message Builder to search for the `title` first, and if found blank, search for the `dmh_record_type.subtype`.
	`hl7.mdm.T02.NTE-3=title|dmh_record_type.subtype`
	• **dmh_record_type.subtype|title**: Configure as follows if you want the MDM Message Builder to search for the `dmh_record_type.subtype` first, and if found blank, search for the `title`.
	`hl7.mdm.T02.NTE-3=dmh_record_type.subtype|title`
	• **NONE**: Configure as follows if you do not want to include NTE-3 field in the MDM T02 message:
	`hl7.mdm.T02.NTE-3=NONE`

HL7 MDM T11 Mapping	`hl7.mdm.T11.MSH-7=r_creation_date`
	`hl7.mdm.T11.MSH-6=station_name`
	`hl7.mdm.T11.PID-7=patient_birth_date`
	`hl7.mdm.T11.PID-18=NONE`
	`hl7.mdm.T11.PID-3=patient_id`
	`hl7.mdm.T11.PID-2=NONE`
	`hl7.mdm.T11.PID-8=patient_sex`
	`hl7.mdm.T11.PID-5=patient_name`
	`hl7.mdm.T11.PV1-3=NONE`
	`hl7.mdm.T11.PV1-19=NONE`
	`hl7.mdm.T11.TXA-2=record_type`

Property	Description
	hl7.mdm.T11.TXA-4=NONE hl7.mdm.T11.TXA-5=NONE hl7.mdm.T11.TXA-6=NONE hl7.mdm.T11.TXA-17=NONE hl7.mdm.T11.TXA-16=r_object_id hl7.mdm.T11.TXA-12=r_object_id

Configuring the HL7 Message Queue Properties

The following table shows the configuration of HL7 Message Queue properties:

Property	Description
hl7.queue.enabled	This property specifies whether to enable or disable the Message Queue support. The default value is **`False`**. For inbound messages, the Message Queue enables the HIP system to store the messages in a Message Queue and send acknowledgement to the external systems before the messages are processed. For outbound messages, the Message Queue enables the HIP system to store the messages that HIP failed to deliver to the external systems even after multiple retries. The failure can be due to network issues or any other reason. For example: `hl7.queue.enabled=True`
hl7.merge.exchange	This property specifies the name of the exchange in RabbitMQ that accepts HL7 inbound messages from an in-queue publisher. This exchange routes the messages to a message queue where they are stored before they are sent to the Documentum in-queue items for processing. You need to configure this property only if **hl7.queue.enabled** is enabled.

Property	Description
hl7.merge.queue	This property specifies the name of the message queue in RabbitMQ where the HL7 inbound messages are stored before they are sent to the Documentum in-queue items for processing. You need to configure this property only if **hl7.queue.enabled** is enabled.
hl7.outbound.exchange	This property specifies the name of the exchange in RabbitMQ that accepts HL7 outbound messages from an out-queue publisher and routes these messages to a message queue where they are stored before they are sent to the Documentum out-queue items for processing. You need to configure this property only if **hl7.queue.enabled** is enabled.
hl7.failed.queue	This property specifies the name of the queue that you want to create in RabbitMQ where the HL7 outbound messages are stored when the external systems fail to receive the outbound messages even after multiple retries, due to network issues. You need to configure this property only if **hl7.queue.enabled** is enabled.

Configuring the HL7 EOB Message for C4E

The following table shows the configuration of HL7 EOB message properties:

Property	Description
repository.eob.scanType	This property specifies the type of the EOB document scanned to the Documentum repository. For example: `repository.eob.scanType=Professional Billing`
repository.eob.hospitalLocation	This property specifies the location of the hospital where the scanning was performed.
repository.eob.batchUser	This property specifies the user ID for which the batch must be created.

Property	Description
repository.eob.batchMode	This property specifies the batch type that must be created. For example: `repository.eob.batchMode=Insurance`
repository.eob.postingDepartment	This property specifies the EOB posting department. For example: `repository.eob.postingDepartment =Cardiology`

Enabling TLS Support for Merge Patient Endpoint

You must configure the MLLP secure port in the `hl7.properties` file and the HTTPS properties in the `cs-repository.properties` file as follows, to enable TLS support for Merge Patient Endpoint (HL7 v2) in Repository.

The following table shows the configuration of mllp secure port and https properties for TLS support:

Property	Description
Properties to be configured in the `hl7.properties` file:	
hl7.inbound.mllp.securePort	`hl7.inbound.mllp.securePort=<port number>`
Properties to be configured in the `cs-repository.properties` file:	
https.keyStore	`https.keyStore=${com.emc.healthcare.home} /keystore.jks`
https.keyStorePassword	`https.keyStorePassword=xxxxxx`
https.server.privateKeyPassword	`https.server.privateKeyPassword=xxxxxx`
https.trustStore	`https.trustStore=${com.emc.healthcare.home} /truststore.jks`
https.trustStorePassword	`https.trustStorePassword=xxxxxx`

Chapter 7

Post-Installation Configuration

This chapter contains the following topics:

Configuring the Repository Server

The `cs-repository.properties` file contains user-definable properties for the XDS Repository Server. This file provides the XDS Repository Server with information to connect to other systems, Documentum repositories, and the XDS Repository itself.

cs-repository.properties, page 107 shows a sample of this file.

To configure `cs-repository.properties`, open `<hip_config_dir>/cs-repository/cs-repository.properties` and define the properties as described in the following topics:

Configuring the Repository Server Properties

The following table shows the configuration of the XDS Repository Server domain identity properties:

Property	Description
repository.homeCommunityId	This required property specifies the Home Community ID for this server. Each XDS Repository Server can be a part of an XDS Affinity Domain Community. This property has no default value. For example: `repository.homeCommunityId=urn:oid:1.19.6.24.109`
repository.uniqueId	This required property specifies the unique ID assigned to the XDS Repository. Each XDS Repository Server identifies itself with a unique ID. This property has no default value. For example: `repository.uniqueId= 1.3.6.1.4.1.2205`

Configuring the Content Server Properties

The following table shows the configuration of properties that define the basic information about Documentum Content Server:

Property	Description
description	This optional property specifies a description for the Content Server. The description is used only for display purposes. This property has no default value. For example: `description=CS Repository Server`
cs.docbaseName	This required property specifies the name of the Documentum Content Server that XDS Repository uses when storing and retrieving documents. This property has no default value. For example: `cs.docbaseName=Healthcare`
cs.userName	This property specifies the user name of the Documentum account you have already created. Deploying Object Types, page 43 provides the instructions to create the user name. This is the user name that XDS Repository uses to access Documentum Content Server. For example: `cs.userName=XdsServer`
cs.password	This property specifies the password of the Documentum account you have already created. Deploying Object Types, page 43 provides the instructions to create to the password. This is the user password that XDS Repository uses to access the Documentum Content Server. For example: `cs.password=xxxxxxx`

Property	Description

	Perform the following steps to encrypt the password:
	1. From the command prompt, execute the `HipEncryptPassword.bat` command as follows:
	`C:\EncryptPassword>HipEncryptPassword.bat`
	2. `Enter clear text password>Password`
	`HIP_ENCR_PASS=JxwGkf59eneCKgVhZljyEA==`
	After generating the encrypted password, the `hip.keystore` file is generated in the `C:\EncryptPassword\` folder.
	3. Copy the `hip.keystore` file to `{com.emc.healthcare.home}/cs-repository/`.
	4. Set the `cs.keystore` property as follows:
	`cs.keystore={com.emc.healthcare.home}/cs-repository/hip.keystore`
	After encryption, the `cs.password` property appears in the `cs-repository.properties` file as follows:
	`cs.password=HIP_ENCR_PASS=JxwGkf59eneCKgVhZljyEA==`

cs.keystore	This property specifies the path and filename for your keystore. If your password is encrypted, you must type the path and filename for your keystore. For example: `cs.keystore={com.emc.healthcare.home}/cs-repository/hip.keystore`

Configuring the ACL Property

The following table shows the configuration of ACL property that is used to configure the ACL name for the patient folder and the documents created in Content Server:

Property	Description
repository.acl.name	This property specifies the ACL name HIP assigns to the objects that it creates in the Content Server. The default value of ACL avoids recreation of an ACL for every document. For example: `repository.acl.name=hip_repository_acl`

Configuring the Registry Properties

The following table shows the configuration of register document set properties that enable the XDS Repository Server to register documents with your registry:

Property	Description
registry.registerDocumentSetUrl	This required property specifies the URL that XDS Repository uses when it registers a document set with the registry. This property has no default value. For example: `registry.registerDocumentSetUrl=http://localhost/registry/services/xds-iti42`
registry.registerDocumentSetAsyncUrl	This property specifies the URL that XDS Repository uses with asynchronous routes when it registers a document set with the registry. This property is not required. If undefined or commented out, XDS Repository Server uses the Register Document Set URL property instead. For example: `registry.registerDocumentSetAsyncUrl=http://localhost/registry/services/xds-iti42`

The Delete Document Set properties are required only when automatic registration and delete document set transactions are enabled.

The following table shows the configuration of delete document set properties that enable the XDS Repository Server to delete documents from the registry:

Property	Description
registry.deleteDocumentSetUrl	This property specifies the URL that XDS Repository uses to delete document set from the Registry. For example: `registry.registerDocumentSetUrl=http://localhost/registry/services/xds-iti62`
registry.deleteDocumentSetAsyncUrl	This property specifies the URL that XDS Repository uses when performing an asynchronous deletion of document set from the Registry. For example: `registry.registerDocumentSetUrl=http://localhost/registry/services/xds-iti62as`

The following table shows the configuration of RegistryStoredQuery transactions with the registry:

Property	Description

registry.storedQueryUrl	This property specifies the URL that the XDS Repository uses when performing RegistryStoredQuery transactions with the Registry.
	
	To delete/deprecate documents from Registry, it is mandatory to configure this property.
	
	For example:
	
	`registry.storedQueryUrl=http://localhost/registry/services/xds-iti18`
	
	If PPIC is enabled on EMC XDS Registry, that is, if **ppic.enabled** property is set to **`true`**, in the `registry.properties` file, you must also:
	• Set the **registry.trusted.hosts.enabled** property in the `registry.properties` file to **`true`**.
	
	and
	
	• Configure **registry.storedQueryUrl** in `cs-repository.properties` file as follows:
	
	`registry.storedQueryUrl=https://<host:port>/registry/services/trustedhosts/xds-iti18`

Property	Description
registry.storedQueryAsyncUrl	This property specifies the URL that the XDS Repository uses when performing asynchronous RegistryStoredQuery transactions with the Registry. For example: `registry.storedQueryAsyncUrl=http://localhost/registry/services/xds-iti18as` To delete/deprecate documents from Registry, it is mandatory to configure this property.

The following table shows the configuration of deprecate document set properties that enable the XDS Repository Server to deprecate documents from the Registry:

Property	Description
registry.updateDocumentSetUrl	This property specifies the URL that the XDS Repository uses to update document set from the Registry. For example: `registry.registerDocumentSetUrl=http://localhost/registry/services/xds-iti57`
registry.updateDocumentSetAsyncUrl	This property specifies the URL that the XDS Repository uses when performing an asynchronous update of document set from the Registry. For example: `registry.registerDocumentSetUrl=http://localhost/registry/services/xds-iti57as`

The MLLP host configuration is used by the XDS Queue Item Processor when registering documents in Documentum. For registration, Patient Identity Source must trigger the New Patient Identity notification to the Registry first, before the documents are registered for a new or unknown patient.

The following table shows the configuration of MLLP host and MLLP port for patient registration:

Property	Description
registry.mllp.host	This property specifies the MLLP listening host where the Registry server is running. This property has no default value.
registry.mllp.port	This property specifies the MLLP listening port of Registry. The default value is `0`. For example: `registry.mllp.port=0`

Configuring the HIM Documentum Metadata Properties

The following table shows the configuration of HIM properties that determine whether to enable the metadata storage in Content Server and the patient folder location:

Property	Description
store.document.metadata	This property specifies whether to enable the storage of complete document metadata in Documentum Content Server. The default value is **`false`**. • If set to **`True`**: Automatic registration and PnR are done for `dmh_document` only. • If set to **`False`**: Automatic registration and PnR are done for `dm_document` only. If set to **`False`**, the **Unique Id** attribute should not be empty in dm_document. You can configure this property to enable automatic registration and PnR for only one type of document at a time. For example: `store.document.metadata=false`
patient.folder.location	This property specifies the cabinet where the patient folders are created. The default cabinet is `Patients`. For example: `patient.folder.location=/Patients`

Configuring the XDS Queue Item Processor Properties

The registration polling properties enable the XDS Repository Server to query the Documentum repository for new healthcare content that is added to the repository through a separate channel. These properties are optional.

The following table shows the configuration of registration polling properties:

Property	Description
repository.queueItemPollingInterval	This property specifies in seconds, how often the XDS Repository Server queries the Repository XDS Queue Item for new documents that need to be registered with the registry. To stop the server from querying for new documents, set the interval to 0. The default value is **0**. For example: `repository.queueItemPollingInterval=0`
repository.queueItemPollingBatchSize	This property specifies the size of the batch of documents that the XDS Repository sends to the registry. The number value indicates the number of documents in the batch. Set the value to 0 to cease queueing documents for batch. The default value is **0**. For example: `repository.queueItemPollingBatchSize=0`
repository.queueItemResubmitPollingInterval	This property is for documents that failed to complete the registration process. This property specifies the duration to wait after the failed document has been modified, before making another attempt to register. The number value represents days. One hour is equivalent to a value of .0416. Set the value to 0 to process all the documents. The default value is **0**. For example: `repository.queueItemResubmitPollingInterval=0`

Configuring the HTTPS Properties

The HTTPS properties enable the XDS Repository Server to use secure HTTPS communication. These properties are optional and do not have default values. If you are not using HTTPS, comment out this property by placing a pound sign (#) in front of it.

The following table shows the configuration of HTTPS properties:

Property	Description
https.keyStore	This property specifies the location of the keystore on the system where you keep the private SSL certificates for the machine. For example: `https.keyStore=<JAVA_HOME>/jre/lib/security`
https.keyStorePassword	This property specifies the password to access the keystore. For example: `https.keyStorePassword=xxxxxx`
https.server.keyAlias	This property specifies the alias used for the server certificate in the keystore. If not specified, the first key read in the keystore is used. For example: `https.server.keyAlias=hlc`
https.server.privateKeyPassword	This property specifies the private key password for encrypting the data. For example: `https.server.privateKeyPassword=xxxxxx`
https.trustStore	This property specifies the location of the truststore on the system where you keep the SSL certificates of machines trusted in TSL connections. This is where you keep the certificates of Document Consumers and XDS Repositories. For example: `https.trustStore=<JAVA_HOME>/jre/lib /security`
https.trustStorePassword	This property specifies the password to access the truststore. For example: `https.trustStorePassword=xxxxxx`

Property	Description
https.ciphersuites	This property specifies the Cipher Suite used to encrypt the session. For example: `https.ciphersuites=TLS_RSA_WITH_AES_128_CBC_SHA`

Configuring the ATNA Properties

The following table shows the configuration of ATNA properties:

Property	Description
audit.host	This property specifies the name of the machine that hosts the ATNA audit repository. The default value is **`localhost`**. For example: `audit.host=localhost`
audit.port	This property specifies the port number of the ATNA audit repository. The default value is **`514`**. For example: `audit.port=514`
audit.transport	This property specifies the transport type for the ATNA audit repository, for example BSD, TLS, or UDP. The default value is **`UDP`**. For example: `audit.transport=UDP`
audit.sourceId	This property specifies the source ID of the event. This property has no default value. For example: `audit.sourceId=${description}`

Configuring the XUA Properties

The following table shows the configuration of XUA properties:

Property	Description
repository.xua.enabled	This property specifies whether to enable XUA for XDS Repository Server. The default value is **`false`**. For example: `repository.xua.enabled=false`

If you are using both the HIP XDS Repository and HIP XDS Registry Servers, you must enable XUA separately for each component.

To enable XUA for HIP XDS Registry Server, open `registry.properties` and set the **`registry.xua.enabled`** property to **`true`**.

The default value is false.

After enabling XUA, you must configure the `cs-repository.properties` file as described in the following topics:

- Configuring the XUA Policy, page 76
- Configuring the XUA SAML Attribute Values, page 76
- Configuring the XUA Attribute Validation Property, page 78
- Configuring the Trusted Assertion Providers Properties, page 79

Configuring the XUA Policy

The `ws-policy.xml` file enables the Web Service security for the server. This policy file defines and enables the standard WS-Security features such as confidentiality (encryption), integrity (signing), and authentication (SAML token) for Web Services.

A sample `ws-policy.xml` file is located in the following XDS Repository directory:

`/webapps/cs-repository/config/cs-repository/`

Copy the sample `ws-policy.xml` file to:

`/webapps/cs-repository/WEB-INF/classes/`

Alternatively, you can place this file in a different folder and define the file location in the server classpath.

Configuring the XUA SAML Attribute Values

The XDS Repository Server uses the XUA SAML attribute properties to validate SAML Security Token attributes sent as part of a registry request.

The following table shows the configuration of XUA SAML attribute values in `cs-repository.properties`:

Property	Description
xua.saml2.token.validator	This property specifies the validator that validates the SAML token in the request. For example: `xua.saml2.token.validator=com.emc .Healthcare.xua.validator.XuaValidator`
xua.crypto.provider	This property specifies the encryption/decryption and signature validation. For example: `xua.crypto.provider=org.apache.ws .security.components.crypto.Merlin`
xua.service.endpoint	This property specifies the endpoint regular expression. The XDS Repository Server compares this value against the audience restriction attribute provided in the token. For example: `xua.service.endpoint=http://localhost:(\ \d)*/hip-webapps-xua-registry-1.6/`
xua.supported.self.issuer.names	This property specifies a comma-separated list of all X-Service-User names that are able to issue self assertions. The XDS Registry endpoints accept requests from only these X-Service-User names (Document Source and Document Consumers). For example: `xua.supported.self.issuer.names =ISC,LeafsproutSTP,http://ith-icoserve .com/senseSTS`
xua.supported.authentication.methods	This property specifies a comma-separated list of authentication methods supported by the XDS Registry. For example: `xua.supported.authentication .methods=urn:oasis:names:tc:SAML:2 .0:ac:classes:X509`

Property	Description
xua.purposeOfUse.codeSystem	This property specifies the supported system for the PurposeOfUseCode attribute. For example: `xua.purposeOfUse.codeSystem=1.3.6.1.4.1.21367.3000.4.1`
xua.purposeOfUse.code.values	This property specifies a comma-separated list of purpose of use codes supported by the XDS Repository. For example: `xua.purposeOfUse.code.values=99-101,99-102`
xua.role.codeSystem	This property specifies the supported code system value for the RoleCode attribute. For example: `xua.role.codeSystem=2.16.840.1.113883.6.96`
xua.role.code.values	This property specifies a comma-separated list of roles supported by the XDS Repository. For example: `xua.role.code.values=112247003,22515006`

Configuring the XUA Attribute Validation Property

The Attribute Validation options can enable or disable specific SAML attribute validations.

The following table shows the configuration of XUA attribute validation option in `cs-repository.properties`:

Property	Description
xua.authz.consent.option	This property specifies enables or disables the validation of the patient consent attribute value of SAML token. The default value is **`false`**. For example: `xua.authz.consent.option=false`

Configuring the Trusted Assertion Providers Properties

The configuration of trusted assertion providers' certificates is a required configuration if XUA is enabled and has no default value.

The following table shows the configuration of trusted assertion providers' properties:

Property	Description
xua.assertion.provider.trustStore	This property specifies the truststore file (for example, truststore.jks).
xua.assertion.provider.trustStorePassword	This property specifies the truststore password.

Configuring the Custom SOAP Routes

The following table shows the configuration of SOAP property:

Property	Description
soap.routeBuilderScriptSource	This property specifies the location of custom groovy file that you define overriding the SoapRouteBuilder script. For example: `soap.routeBuilderScriptSource=file:/C:/Users/Administrator/.hip/cs-repository/MllpRouteBuilder.groovy` where `MllpRouteBuilder.groovy` is the custom groovy file. **Note:** The `SoapRouteBuilder.groovy` file must have the `xuaEnabled` property defined in the file. The server fails to start if the property is missing in the file.

Configuring the RabbitMQ Properties

The RabbitMQ properties need to be configured only if you are using RabbitMQ to queue HL7 messages.

The following table shows the configuration of RabbitMQ properties:

Property	Description
rabbitmq.ssl.enabled	This property specifies whether SSL is enabled between HIP servers and RabbitMQ server. The default value is **`False`**. You must configure the HTTPS properties if this property is enabled.
rabbitmq.host	This property specifies the name of the machine that hosts RabbitMQ server.
rabbitmq.port	This property specifies the RabbitMQ server port.
rabbitmq.username	This property specifies the user name of for RabbitMQ server.
rabbitmq.password	This property specifies the password of for RabbitMQ server.
rabbitmq.virtualhost	This property specifies the virtual host name for RabbitMQ server. This host should be pre-created in the RabbitMQ server to prevent deployment failure.

Configuring the PPIC Server Properties

The following table shows the configuration of PPIC properties:

Property	Description
ppic.enabled	This property specifies whether to enable the PPIC server for user authorization. If enabled, the PPIC server checks if the user has permission to view the documents that a retrieve query returns. The default value is **`False`**. For example: `ppic.enabled=false` **Note:** You must configure the **ppic.pdpServiceUrl** property if you enable this property.
ppic.pdpServiceUrl	This property specifies the URL for making PDP service call for PPIC. For example: `ppic.pdpServiceUrl=http://localhost:8080/ppic/pdp`

Configuring the Usage Report Properties

The following table shows the configuration of usage report properties:

Property	Description
usagereport.username	This property specifies the user name to log in to the Usage Reporting web application. For example: `usagereport.username=Administrator` The default value is **`Administrator`**
usagereport.password	This property specifies the password you need to type to log in to the Usage Reporting web application. For example: `usagereport.password=password` The default value is **`password`**.

Securing the Repository Server Properties File

The `cs-repository.properties` file contains access information for the XDS Repository.

Secure the file as follows:

- Restrict access to the system where the XDS Repository Server and the `cs-repository .properties` file reside.
- Restrict access to the `cs-repository.properties` file by providing the read/write access only to the HIP Administrator. See www.wiki.apache.org/tomcat/FAQ/Password.
- Provide an encrypted Documentum user password. You can create an encrypted password during Documentum installation or through the encryptPassword utility. The *EMC Documentum Content Server Administration and Configuration Guide* provides more details on password encryption.

Configuring the Repository Server Context Extensions

Configuration to Support Custom HL7 Message Processing

The HIP XDS Repository supports handling of additional HL7 messages (that is, messages other than ADT Merge) and MDM T02 inbound messages. You need to do this configuration only if you want to process additional HL7 messages.

To enable processing of custom HL7 messages:

1. Create a custom route builder.

   For example:

   ```
   CustomRouteBuilder.groovy
   ```

2. Define custom routes in the custom route builder.

   For example:

   ```
   public class CustomRouteBuilder extends SpringRouteBuilder
   {
       @Override
       public void configure () throws Exception
       {
           from( 'direct:customEndpoint')
               .process() {log.info('it hits this endpoint')}
       }
   private final static  Logger log = LoggerFactory.getLogger(CustomRouteBuilder.class);
   }
   ```

3. Type the id and path of the custom route builder in the `cs-repository-context-extension.xml` file.

   For example:

   ```
   <lang:groovy id="CustomRouteBuilder"
   script-source=
   "classpath:com/emc/healthcare/xds/repository/commons/CustomRouteBuilder.groovy"/>
   ```

4. Inject or register the custom route defined above by defining a bean in the `cs-repository-context-extension.xml` file.

   For example:

   ```
   <bean
   id="RouteBuilderReg"
   class="com.emc.healthcare.commons.core.hl7.RouteBuilderRegistration"
   init-method="register">
   <constructor-arg ref="CustomRouteBuilder"/>
   </bean>
   ```

5. Map the incoming message to a custom route by specifying the type of the incoming message and the version to a specific end point.

   For example, consider the following MSH segment of an incoming message:

   ```
   MSH|^~\&|ICW-MPI@SHG|DOMAIN1_ADMITTING|MESA_XREF|XYZ_HOSPITAL|
   200310011100-0600||ADT^A40|10506116|P|2.3.1
   ```

Here, the value of the 2nd component of the 9th field `(ADT^A40)`, that is, `A40` is the message type and value of the twelfth field, that is, `2.3.1` is the version of the message.

6. Specify the message type and version for the `entry key` and `value` parameters respectively in the `cs-repository-context-extension.xml` file.

   For example:

```
<bean id="MessageToRouteMap"
class="com.emc.healthcare.commons.core.hl7.MessageToRouteMap"
init-method="register">
   <constructor-arg>
     <map>
       <entry key="A40-2.3.1"
       value="direct:customEndpoint"/>
     </map>
   </constructor-arg>
</bean>
```

   In the above example, A40 indicates the message type and 2.3.1 indicates the version of the message. `direct:customEndpoint` indicates the endpoint to which the message is mapped.

Configuration to Support Unknown Message Processing

By default, HIP Repository rejects message of types that are not supported by users or by HIP. However, HIP provides you the provision to process unknown messages by adding a new messagetoroute map with entry key configured as **`defaultRoute`** and value configured as any custom route that can process unknown messages.

For example:

```
<bean id="MessageToRouteMap"
class="com.emc.healthcare.commons.core.hl7.MessageToRouteMap"
init-method="register">
   <constructor-arg>
     <map>
       <entry key="defaultRoute"
       value="user-defined endpoint"/>
     </map>
   </constructor-arg>
</bean>
```

Customization to Support HL7 MDM T02 Inbound Messages Processing

To enable HIP Repository to process MDM T02 inbound messages, you must customize the spring bean by mapping the fields in the MDM T02 inbound message with the corresponding properties defined in the Documentum Object Model.

You can customize the `cs-repository-context-extension.xml` file with required values as mentioned in the following procedure:

To enable processing of HL7 MDM T02 Inbound Messages

1. Customize the spring bean with required key-value pair as shown in the following example:

```
<bean id="MDMT02MessageDctmMappingConfig" class=
        "com.emc.healthcare.xds.repository.cs.dfc.hl7.HL7v2MessageDctmMappingConfig">
        <property name="documentType" value="dmh_document"/>
        <property name="csRepositoryConfig" ref=
        "com.emc.healthcare.xds.repository.cs.RepositoryConfig"/>
        <property name="patientFolderHL7v2MessageFieldsMap">
            <map>
                <entry key="patient_id" value="PID-3"/>
                <entry key="patient_name" value="PID-5"/>
                <entry key="patient_sex" value="PID-8"/>
                <entry key="date:patient_birth_date" value="PID-7"/>
            </map>
        </property>

        <property name="documentHL7v2MessageFieldsMap">
            <map>
                <entry key="object_name" value="TXA-16"/>
                <entry key="unique_id" value="TXA-16"/>
                <entry key="availability_status" value="TXA-19"/>
                <entry key="record_type" value="TXA-2"/>
                <entry key="patient_id" value="PID-3"/>
                <entry key="date:origination_date" value="TXA-6-1"/>
                <entry key="date:transcription_date" value="TXA-7-1"/>
                <entry key="account_number" value="PID-18"/>
                <entry key="completion_status" value="TXA-18"/>
                <entry key="mime:mime_type" value=""/>
            </map>
        </property>
    </bean>


<util:map  id="MDMT02_EDSubType_MimeMapping" map-class="java.util.HashMap" >
        <entry key="xml" value="text/xml"/>
        <entry key="pdf" value="application/pdf"/>
        <entry key="x-hl7-cdalevel-one" value="text/xml"/>
        <entry key="x-hl7-cdalevel-three" value="text/xml"/>
        <entry key="x-hl7-cdalevel-two" value="text/xml"/>
        <entry key="jpeg" value="image/jpeg"/>
        <entry key="rtf" value="application/msword"/>
        <entry key="tiff" value="image/tiff"/>
        <entry key="html" value="text/html"/>
        <entry key="gif" value="image/gif"/>
    </util:map>
```

Configuration to Support PnR Custom Extensions

1. Implement the **CSContentObjectHandler** interface either directly or by extending the **DefaultCSContentObjectHandler**.

   The **DefaultCSContentObjectHandler** provides many default methods which can be overridden to tweak its behavior.

2. Create a jar file with your custom implementation and copy the jar to `WEB-INF/lib` of the Repository web application.

3. In the `cs-repository-context-extension.xml` file, define a bean with your custom implementation.

4. Specify the id for the bean as follows:

```
<bean
   id="com.emc.healthcare.xds.repository.cs.api.pnrext.CSContentObjectHandler"
   class="my.own.pnr.customizations.CustomCsContentObjectHandler">
   constructor-arg name="config"
                   ref="com.emc.healthcare.xds.repository.cs.RepositoryConfig"/>
</bean>
```

5. Save the file.

Configuring DFC

The server DFC components require connection information for the Documentum Content Repository. You can provide these settings through a properties file located in the HIP configuration directory:

<*hip_config_dir*>/cs-repository/dfc.properties

This file directs the server to another location for the properties. This is done to keep the XDS Repository Server `dfc.properties` file separate from the main `dfc.properties` file.

The `dfc.properties` file located in the configuration directory contains connection information for the Documentum Content Server. Ensure that the HIP configuration directory contains the `dfc.properties` file with connection information for your Documentum Content Server.

Configuring the HIP Documentum Mapping Properties

The `hip-dctm-mapping.properties` file provides the mapping of the Documentum object types and attributes to the following IHE object types and attributes:

- Healthcare document object type and attributes
- Repository folder location (defaults to `/Patients` cabinet)
- Patient folder object type and attributes

This configuration uses `Key`=**`Value`** pairs, where `Key` is the IHE attribute name and the **`Value`** is the mapped object type attribute name.

For example,

`document.object.type=dmh_document`
where `document.object.type` is the IHE attribute and `dmh_document` is the mapped object type attribute name.

If the `store.document.metadata` property in the `cs-repository.properties` file is configured as **`false`**, the Repository does not consider the mapping provided in the `hip-dctm-mapping.properties` file while creating XDS documents. In this case, the object is stored as `dm_document`.

If the `store.document.metadata` property in the `cs-repository.properties` file is configured as **`true`**, the Repository maps the object types according to the configuration given in the `hip-dctm-mapping.properties` file, by default.

However, if the user has customized the `hip-dctm-mapping.properties` file with user-specific object types and attributes, the Repository maps the object types according to the configuration given by the user.

The hip-dctm-mapping.properties, page 116 shows a sample `hip-dctm-mapping.properties` file.

Configuring the HIP PPIC Mapping Properties

The following table shows the configuration of hip-ppic-mapping.properties:

Property	Description
request.subjectId	This property specifies the logical identifier of the user performing the original service request. For example: `request.subjectId =urn:oasis:names:tc:xacml:1 .0:subject:subject-id`
request.subjectRole	This property specifies the relevant user subject roles from a locally defined code-set. For example: `request.subjectRole =urn:oasis:names:tc:xacml:2 .0:subject:role`

Enabling the Request and Response Validator Properties

The Request and Response validator flags are located in the Spring Configuration file inside the deployed WAR. These flags are enabled by default. These validators must be enabled to avoid the problems that may occur if the request is not correct.

The following table shows the configuration of request and response validator properties:

Property	Description
repository.iti41.requestValidator .enabled **repository.iti43.requestValidator .enabled**	This optional property specifies whether to disable the incoming message validation for the specified IHE transaction. For example: `repository.iti41.requestValidator.enabled =true repository.iti43.requestValidator .enabled=true`
repository.iti41.responseValidator .enabled **repository.iti43.responseValidator .enabled**	This optional property specifies whether to disable the outgoing message validation for the specified IHE transaction. For example: `repository.iti41.responseValidator.enabled =true repository.iti43.responseValidator .enabled=true`

Configuring the Web Container Heap Memory

You must configure the initial and maximum heap size settings for the J2EE Web Application container according to the memory allocated to the server. You must also monitor the J2EE Web Application container during testing to ensure that the settings are sufficient.

For Tomcat, EMC recommends:

Running the server as a service:

```
#Set initial heap size Xms and maximum heap size -Xmx JAVA_OPTS="
-Xms512m -Xmx1024m -XX:MaxPermSize=512m"
```

Running the server as a standalone system:

```
Set "JAVA_OPTS"=-Xms256m -Xmx1g -XX:MaxPermSize=256m"
```

Configuring SSL for Tomcat

To configure Tomcat for SSL, add the paths for the keystores and truststores to the following file:

`<Tomcat_install_dir>/conf/server.xml`

For example:

```
<Connector port="8443"
    protocol="org.apache.coyote.http11.Http11NioProtocol"
    SSLEnabled="true" maxThreads="150" scheme="https"
    secure="true" clientAuth="false" sslProtocol="TLS"
```

```
keystoreFile="C:/Users/Administrator/.hip/keystore.jks"
keystorePass="changeit"
truststoreFile="C:/Users/Administrator/.hip/truststore.jks"
truststorePass="changeit"/>
```

Configuring SSL for WebLogic

1. Log in to the WebLogic console.

   For example:

   ```
   http://server:7001/console
   ```

2. Ensure that the SSL port is enabled for each server.

   You can find the setting **SSL Listen Port Enabled** under **Configuration -> General** for each server. Enable this setting and make sure you use a correct and unused port.

3. For each server, change the KeyStore configuration as follows:

   a. Go to **Configuration -> Keystores**.

   b. In the **Keystores** section, click **Change**.

   c. Type the values for the following fields as given in the example below:

   Custom Identity Keystore=C:/Users/Administrator/.hip/keystore.jks

   Custom Identity Keystore Type=JKS

   Custom Identity Keystore Passphrase=changeit

   Custom Trust keystore=C:/Users/Administrator/.hip/truststore.jks

   Custom Trust Keystore Type=JKS

   Custom Trust Keystore Passphrase=changeit

4. For each server, change the SSL configuration as follows:

   a. Go to **Configuration -> SSL**.

   b. Type the values for the following fields as given in the example below:

   Private Key Alias=serverXX

   Private Key Passphrase=changeit

5. Click **Save**.

Chapter 8

Verifying Installation

This chapter contains the following topics:

Verifying the Installation Using Tomcat

1. Ensure that the library dependencies are installed.

   Installing Third-party Library Dependencies, page 44 provides details on installing library dependencies.

2. Ensure that the `.hip` folder is available in `C:\Users\<username>` folder.

   If you want to override the default location of the HIP configuration folder, override the `com.emc.Healthcare.com` system property when you start the J2EE Web Application container.

   Use the following syntax:

   ```
   Dcom.emc.Healthcare.home=<hip_config_directory>
   ```

3. Start the XDS Repository Server using the normal start procedure for the J2EE web application container.

   For example, start the server on Tomcat using the following command:

   ```
   [root]# service Tomcat start
   ```

4. Check the log file for errors.

   For example:

   ```
   /usr/share/apache-Tomcat-7.0.25/logs/catalina.out
   ```

5. Open a web browser and type the following URL:

   ```
   http://<host:port>/cs-repository/services
   ```

   The WSDL page appears, which indicates that the server installation is successful.

Verifying the Installation Using WebLogic

1. Log in to the WebLogic Admin console.
2. Ensure that the library dependencies are installed as described in Installing Third-party Library Dependencies, page 44.
3. Ensure that the `.hip` folder is available in `C:\Users\<username>` folder.

   If the user does not have rights to access the `C:\Users\<username>` folder, perform the following steps:

   a. Create `.hip` folder in any other location.

   b. Update the `startWebLogic.cmd` file located at `~\Oracle\Middleware\user_projects\domains\domain{domain used for deployment}` by adding the following line:

   ```
   set JAVA_OPTIONS=-Dcom.emc.healthcare.home=C:\.hip (".hip location")
   ```

4. Ensure that the HIP configuration properties files are present in the `.hip` folder.
5. Restart the system for the above changes to take effect.
6. Start the WebLogic server.
7. Deploy the Repository WAR file as described in Deploying the HIP Repository WAR File Using WebLogic, page 48.
8. Open a web browser and type the following URL:

   ```
   http://<host:port>/cs-repository/services
   ```

   The WSDL page appears, which indicates that the server installation is successful.

Chapter 9

Upgrade

This chapter contains the following topic:

Upgrading XDS Repository from 1.6B250714_update to 1.7

1. Deploy `hip-xds-1.7.0.dar`.
2. Deploy `hip-dctm-1.7.0.dar`.
3. Delete the previous version of Repository WAR file from the deployed location.
4. Build new repository WAR from the `hip-cs-repository-1.7.0.zip` file.
5. Go to the `\cs-repository\config\` folder in the WAR file.
6. Copy the `cs-repository` folder containing the properties file to the HIP_HOME directory.
7. Configure the properties file in the HIP_HOME directory.
8. Deploy the `hip-cs-repository-1.7.0.war` file.
9. Verify the upgrade.

Chapter 10

Troubleshooting

This chapter contains the following topics:

Log Settings

A log is a chronological record of system activities that is sufficient to enable the reconstruction and examination of the sequence of environments and activities surrounding or leading to an operation, procedure, or event in a security-relevant transaction from inception to final results.

Log Description

The log file for XDS Repository Server is located in `<hip_config_directory>\logs`.

For Apache Tomcat:

`C:\Users\<username>\.hip\logs\cs-repository.log`

For Oracle WebLogic:

`C:\Users\<username>\.hip\logs\cs-repository.log`

Log Management and Retrieval

XDS Repository Server uses the Simple Logging Facade for Java (SLF4J) combined with a `Log4j` logging provider implementation.

The default log level setting is INFO.

You can increase the trace messages by setting the log level to DEBUG in

`<tomcat installation directory>\webapps\cs-repository\WEB-INF\classes\log4j.xml` file.

For example:

```
<--!Set to DEBUG to see detailed HIP message information -->
<logger name="com.emc.healthcare">
<level value="DEBUG"/>
</logger>
```

Issues and Resolutions

This section describes the following XDS Repository errors and their solutions:

ERROR Exception During ProvideDocumentSetProcessor Processing

Problem

The cs-repository webapp is successfully deployed but transaction fails with the following error message:

```
: 2014-05-16 16:13:26,906 INFO Object protocol version 2 2014-05-16
16:13:27,141 ERROR Exception during ProvideDocumentSetProcessor
processing com.documentum.fc.common.DfRuntimeException:  [DM
_SESSION_E_AUTH_FAIL]error:  "Authentication failed for user
```

```
Administrator with docbase Healthcare." at com.documentum.fc.common
.DfRuntimeException.convertToRuntimeException(DfRuntimeException.java:35)
at com.emc.healthcare.commons.cs.dfc.SimpleDfSessionManager.get
(SimpleDfSessionManager.java:83) at com.emc.healthcare.commons.cs
.dfc.SimpleDfSessionManager.get(SimpleDfSessionManager.java:33) at
com.emc.healthcare.commons.cs.dfc.ExchangeDfSessionManager.getNew
(ExchangeDfSessionManager.java:43)
```

Cause

The user name and password for authenticating the Content Server docbase is not correctly set.

Resolution

Check following properties in `cs-repository.properties` file:

```
# The user name for authenticating to the Content Server docbase.
# REQUIRED, NO DEFAULT
cs.userName=<user name>
# The password for authenticating to the Content Server docbase
# REQUIRED, NO DEFAULT
cs.password=<password>
```

Error Creating Bean with Name ’mllpSSLFilter’

Problem

You get the following error message:

```
2014-05-16 14:58:30,143 ERROR Context initialization failed
org.apache.camel.RuntimeCamelException:  org.springframework.beans.factory
.BeanCreationException:  Error creating bean with name 'mllpSSLFilter'
defined in ServletContext resource [/WEB-INF/spring/context.xml]:
Cannot resolve reference to bean 'sslContext' while setting constructor
argument; nested exception is org.springframework.beans.factory
.BeanCreationException:  Error creating bean with name 'sslContextFactory'
defined in ServletContext resource [/WEB-INF/spring/context.xml]:
Cannot resolve reference to bean 'keyStore' while setting bean
property 'keyManagerFactoryKeyStore'; nested exception is org
.springframework.beans.factory.BeanCreationException:  Error creating
bean with name 'keyStoreFactory' defined in ServletContext resource
[/WEB-INF/spring/context.xml]:  Error setting property values; nested
exception is org.springframework.beans.PropertyBatchUpdateException;
nested PropertyAccessExceptions (1) are:  PropertyAccessException 1:
org.springframework.beans.MethodInvocationException:  Property 'dataFile'
```

```
threw exception; nested exception is java.lang.NullPointerException
at org.apache.camel.util.ObjectHelper.wrapRuntimeCamelException
(ObjectHelper.java:1344) at org.apache.camel.spring.SpringCamelContext
.onApplicationEvent(SpringCamelContext.java:120)
```

Cause

You are trying to use MLLP secure ports without specifying the HTTPS parameters.

Resolution

If you plan to use `hl7.inbound.mllp.securePort` then `https.server.privateKeyPassword` must be specified. The following HTTPS properties in `cs-repository.properties` must also be specified.

- https.keyStore
- https.keyStorePassword
- https.server.keyAlias
- https.server.privateKeyPassword
- https.trustStore
- https.trustStorePassword
- https.ciphersuites

For example:

```
https.keyStore=${com.emc.healthcare.home}/keystore.jks
https.keyStorePassword=changeit
https.trustStore=${com.emc.healthcare.home}/truststore.jks
https.trustStorePassword=changeit
https.ciphersuites=TLS_RSA_WITH_AES_128_CBC_SHA
https.server.privateKeyPassword=changeit
```

Context Initialization Failing when Deploying Server WAR Files

Issue with HIP Configuration

Problem

When you try to install the Repository WAR files, you get an error message as follows:

```
o.s.web.context.ContextLoader - Context initialization failed
org.springframework.beans.factory.BeanInitializationException:
```

```
Could not load properties; nested exception is java.io
.FileNotFoundException:  C:\Users\Administrator\.hip\cs-repository\
cs-repository.properties (The system cannot find the path specified)at
org.springframework.beans.factory.config.PropertyResourceConfigurer
.postProcessBeanFactory (PropertyResourceConfigurer.java:87)
~[spring-beans-3.2.4.RELEASE.jar:3.2.4.RELEASE]
```

Cause

The `.hip` folder is not available in `<user home>`.

Resolution

Ensure that the HIP configuration properties are available in the `%HIP HOME%`.

Creating the HIP Configuration Directory, page 46 and Deploying the Properties Files in HIP Configuration Directory, page 47 provide more details on HIP configuration.

Issue with Camel Jar Files

Problem

When you try to install the Repository WAR files, you get an error message as follows:

```
10:57:01.592 [localhost-startStop-1] INFO o.s.b.f.xml
.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [META-INF/spring/xua-context.xml]10:57:01.895
[localhost-startStop-1] ERROR o.s.web.context.ContextLoader
- Context initialization failed org.springframework.beans
.factory.parsing.BeanDefinitionParsingException:  Configuration
problem:  Unable to locate Spring NamespaceHandler for XML schema
namespace [http://camel.apache.org/schema/spring] Offending
resource:  ServletContext resource [/WEB-INF/spring/context.xml] at
org.springframework.beans.factory.parsing.FailFastProblemReporter.error
(FailFastProblemReporter.java:68) ~[spring-beans-3.2.4.RELEASE.jar:3.2.4
.RELEASE]
```

Cause

Camel jar files are not added to the tomcat classpath.

Resolution

Install Camel Library Dependencies.

Installing Third-party Library Dependencies, page 44 describes the steps to install Camel library dependencies.

Error while Installing HAPI Jar Files

Problem

Installation fails.

Cause

- You downloaded an incorrect version of HAPI.
- You copied the HAPI jar files directly to the `cs-repository/WEB-INF/lib` folder.

Resolution

Download the correct version of HAPI (HAPI 2.0) files and refer the files from a separate folder defined in web applications server's context classpath.

In tomcat 7.0 the hapi folder is referred from `conf/context.xml` file.

If you are using WebLogic, verify if the `HAPI 2.0 jar` files are successfully installed.

The topic Installing Third-party Library Dependencies, page 44 describes the steps to install the jar files.

Also, check the log files in the logs folder (`~\Oracle\Middleware\user_projects\domains\domain{domain used for deployment}\servers\AdminServer\logs`) for any errors.

Cannot Connect to the XDS Repository Server

Problem

You are unable to connect to the XDS Repository Server.

Cause

You are using incorrect URL or the Repository installation is incomplete.

Resolution

- Ensure that the endpoint (URL/port) used to connect to the XDS Repository Server is correct.

  You can also validate the URL by accessing the XDS Repository Server WSDL.

  `http://localhost:<port>/cs-repository/services`.
- Ensure that the server is up and running by performing the steps in Chapter 8, Verifying Installation.

  Access the XDS Repository Server WSDL page at `http://localhost:<port>/cs-repository/services/`.

  If the WSDL loads then the XDS Repository Server is up.
- Check the `cs-repository.log` file for startup errors.

Java Errors at Startup

Problem

You get an error message as follows in the startup:

```
java.lang.NoClassDefFoundError:  Lca/uhn/hl7v2/parser/Parser; at
java.lang.Class.getDeclaredFields0(Native Method) at java.lang
.Class.privateGetDeclaredFields(Unknown Source) at java.lang.Class
.getDeclaredFields(Unknown Source) at org.codehaus.groovy.vmplugin.v5
.Java5.configureClassNode(Java5.java:313)
```

Cause

The server cannot find the HAPI jar files because they were not deployed or were deployed incorrectly.

Resolution

Install the HAPI jar files as described in Installing Third-party Library Dependencies, page 44.

DFC Error at Startup

Problem

You get an error message as follows:

```
2013-06-20 10:08:02,270 ERROR DfException::THREAD: Thread-8; MSG:
[DFC_API_W_ATTEMPT_TO_USE_DEPRECATED_CRYPTO_API] WARNING: Program attempts
to use deprecated non-FIPS compliant cryptography API. This is not
recommended.  Please consult the documentation for more detail.
```

Cause

NA

Resolution

You can ignore this message as this is a harmless message.

Cannot Connect to the Documentum Repository

Problem

XDS Repository Server is unable to connect to the Documentum repository and you get the following error message in the log file:

```
2011-04-27 21:17:47,134 WARN [DFC_DOCBROKER_EXCLUDED] Docbroker
"Healthcare/2001:0:4137:9e76:306f:1b44:3f57:cd9a:1489" excluded from
active docbroker list due to "3" connection failed attempts with exception
"Connection refused:  connect" java.net.ConnectException:  Connection
refused:  connect at java.net.PlainSocketImpl.socketConnect(Native Method)
at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:333) at
java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:195)
```

Cause

XDS Repository Server is unable to connect to the Documentum repository because:

- The Documentum repository is not currently running
- The Content Server database and log in information are incorrect

Resolution

Ensure that the Documentum repository is running properly.

Ensure that the `cs-repository.properties` file uses the correct database and log in information. The user names are case-sensitive.

Registering a Document More Than Once

Problem

If you submit a duplicate Provide and Register Document Set transaction after an initial successful transaction, the duplicate transaction fails.

Cause

The document already exists.

Resolution

You can re-register or replace the existing document, but you cannot replace or use version control on a document in the repository using the same unique ID.

XUA Policy File Error

Problem

You get the following error in the log file during initialization:

```
Context initialization failed org.apache.camel.RuntimeCamelException:
org.apache.cxf.ws.policy.PolicyException:  Policy reference
classpath:ws-policy.xml could not be resolved.
```

Cause

XDS Repository Server cannot find the `ws-policy.xml` file defined in the classpath.

Resolution

Ensure that you copy the sample `ws-policy.xml` file from

`/webapps/cs-repository/config/cs-repository/`

and place it in

`/webapps/cs-repository/WEB-INF/classes/`

Alternatively, you can place this file in a different folder and define the file location in the server classpath.

servicestore.jks File Not Found Error

Problem

You get an error message as follows in the log file:

```
java.io.FileNotFoundException:  certificates\servicestore.jks (The system
cannot find the path specified)
```

Cause

XDS Repository Server is unable to find the keystore file specified in the `serviceKeystore.properties` file.

If you are using HTTPS to connect to the XDS Repository, the appropriate TLS certificates (keystore.jks, truststore.jks) must be located in the HIP_HOME directory.

Resolution

- Copy the keystore file to the location specified in `serviceKeystore.properties`.
- Verify Tomcat is configured for SSL as follows:
  — Edit `tomcat\conf\server.xml` to add the paths for the keystores/truststores as follows:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="C:/Users/Administrator/.hip/keystore.jks" keystorePass="changeit"
truststoreFile="C:/Users/Administrator/.hip/truststore.jks" truststorePass="changeit"/>
```

Required Header Not Present Error

Problem

XDS Repository Server writes the following error to the log file:

```
org.apache.cxf.interceptor.Fault:  A required header representing a
Message Addressing Property is not present
```

Cause

An XUA enabled XDS Repository Server received a request without a security header.

Resolution

With XUA enabled on the server, all requests must contain a security header. Either disable XUA on the server or instruct the sending application to send requests with security headers.

To disable XUA, open the `cs-repository.properties` file and set the `repository.xua .enabled` property to false.

Failure to Authenticate Security Token

Problem

Authentication of Security Token fails.

Cause

Incorrect configuration of XUA related properties in the `cs-repository.properties` file.

Resolution

Ensure that the **xua** related properties in `cs-repository.properties` file are configured correctly.

Verify if the security options are enabled and they are re configured correctly.

Check the following properties:

- role
- purpose of use
- authentication methods
- token issuer names
- audience restriction
- certificate configurations

java.lang.OutOfMemoryError: PermGen Space Error

Problem

You get `java.lang.OutOfMemoryError:PermGen space` error while verifying the deployment of HIP Repository.

Cause

The permanent generation heap is full.

Resolution

Increase the Permgen space.

For Tomcat:

```
Set JAVA_OPTS=-Xms256m -Xmx512m -XX:PermSize=256m -XX:MaxPermSize=512m
```

For WebLogic:

Replace the following lines in the `setDomainEnv.cmd` files located at `C:\Oracle\Middleware\user_projects\domains\base_domain\bin`

Replace:

```
set WLS_MEM_ARGS_64BIT="-Xms256m -Xmx512m"
   set WLS_MEM_ARGS_32BIT="-Xms256m -Xmx512m"
```

With:

```
set WLS_MEM_ARGS_64BIT=-Xms256m -Xmx512m -XX:MaxPermSize=512m
  set WLS_MEM_ARGS_32BIT=-Xms256m -Xmx512m -XX:MaxPermSize=512m
```

Errors Related to ADT Merge Patient Identities Requests

Problem

Retiring Patient Records are not moving to the Surviving Patient Folder.

Cause

- Java Method Server is not running
- HipPatientMergeJob is inactive
- ADT Merge Identities related configuration is incorrect

Resolution

Verify that:

- The Documentum Java Method Server is running
- The HipPatientMergeJob is installed and active
- The `patientsCabinetPath` argument is set to valid Patients Records cabinet in HipPatientMergeJob custom Arguments

The Configuring the HIP Merge Job and Method Arguments, page 51 provides more details on the configuration of Merge Job and Method Arguments.

Errors Related to HL7 MDM Message Processing

Problem

MDM T01 messages with the content object ID as unique document identifier are not sent.

Cause

Incorrect configuration of HL7 Outbound Properties.

Resolution

Verify that:

- `hl7.properties` is located in `<hip_home>/cs-repository` location
- `hl7.outbound.mllp.host` property is set to correct HL7 External Engine
- `hl7.outbound.mllp.port` is set to correct port

The Configuring the HL7 Outbound Properties, page 55 provides more details on the configuration of HL7 Outbound Properties.

Appendix A

Sample Configuration Files

This appendix contains the following sample files:

cs-repository.properties

EMC ships a sample `cs-repository.properties` file in your installation package. After deploying the XDS Repository WAR file, the sample can be found in `$CATALINA_BASE/webapps/cs-repository/config/`.

A sample file is reproduced below.

```
# User settable properties are listed in this file.
# The order of property evaluation is as follows:
# 1: ${com.emc.healthcare.home}/cs-repository.properties is consulted first
# 2: System.properties is consulted second.
#
# Property values are set in the order they are encountered. If the same
# property is defined in multiple locations, the final setter takes precedence
#
# All settable properties for this server are enumerated in this file.
# If a property defined and commented out this documents its default value.

# The description property is used to contain a description of this
# server instance for display purposes
# NO DEFAULT
description=CS Repository Server

# Content Server Properties.
#
# The following three properties have NO DEFAULT and MUST BE SET.
# The server will not boot without these properties being defined.
#
```

```
# The Content Server docbase name.
# REQUIRED, NO DEFAULT
# cs.docbaseName=

# The user name for authenticating to the Content Server docbase.
# REQUIRED, NO DEFAULT
# cs.userName=

# The password for authenticating to the Content Server docbase
# REQUIRED, NO DEFAULT
# cs.password=

# cs.keystore property is required if HIP encrypted password is configured
# DEFAULT=${com.emc.healthcare.home}/cs-repository/hip.keystore
# cs.keystore=${com.emc.healthcare.home}/cs-repository/hip.keystore

# The Home Community ID for this server
# Example: repository.homeCommunityId=urn:oid:1.19.6.24.109.42.1.2
# REQUIRED, NO DEFAULT
# repository.homeCommunityId=

# The repository unique ID.
# Example: repository.uniqueId=1.3.6.1.4.1.2205.2154.3.1.2
# REQUIRED, NO DEFAULT
# repository.uniqueId=

# HIP applies this ACL name to the objects that it creates in Content Server
# DEFAULT hip_repository_acl
# repository.acl.name=hip_repository_acl

# The URL for performing RegisterDocumentSet transactions with the registry.
# Example: registry.registerDocumentSetUrl=http://localhost/registry/xds-iti42
# REQUIRED, NO DEFAULT
# registry.registerDocumentSetUrl=

#The URL for performing asynchronous RegisterDocumentSet transactions
# with the registry.
# Example:
# registry.registerDocumentSetUrl=http://localhost/registry/xds-iti42as
# DEFAULT:  The synchronous URL will be used in Async routes
# registry.registerDocumentSetAsyncUrl=

# The URL for performing DeleteDocumentSet transactions with the registry.
# Example:
# registry.registerDocumentSetUrl=http://localhost/registry/xds-iti62
# registry.deleteDocumentSetUrl=

#The URL for performing asynchronous DeleteDocumentSet transactions
# with the registry.
# Example:
# registry.registerDocumentSetUrl=http://localhost/registry/xds-iti62as
# registry.deleteDocumentSetAsyncUrl=


# The URL for performing RegistryStoredQuery transactions with the registry.
# Example:
# registry.registerDocumentSetUrl=http://localhost/registry/xds-iti18
# registry.storedQueryUrl=

#The URL for performing asynchronous RegistryStoredQuery transactions
# with the registry.
# Example:
# registry.registerDocumentSetUrl=http://localhost/registry/xds-iti18as
# registry.storedQueryAsyncUrl=
```

```
# The URL for performing Update transactions with the registry.
# Example: registry.updateDocumentSetUrl=http://localhost/registry/xds-iti57
# registry.updateDocumentSetUrl=

#The URL for performing asynchronous update transactions with the registry.
# Example:
# registry.updateDocumentSetAsyncUrl=http://localhost/registry/xds-iti57as
# registry.updateDocumentSetAsyncUrl=


# Flag to enable/disable storing of complete document metadata in
# Content Server
# DEFAULT false
#store.document.metadata=false

# Creates patient folders under the location configured below
# DEFAULT /Patients
# patient.folder.location=/Patients


# Registry Mllp Listening host to register patient.
# NO DEFAULT
# registry.mllp.host=

#Registry Mllp Listening port to register patient. Set to 0 to turn off.
# DEFAULT 0
# registry.mllp.port=0

# The Repository Xds Queue Item polling interval to register/delete/deprecate
# XDS documents - default is 0 seconds. Set to 0 to turn off.
# DEFAULT 0
# repository.queueItemPollingInterval=0

# The batch size for the Repository Xds Queue Item poller to query for
# XDS documents to register/delete/deprecate - default is 0 for no batching.
# DEFAULT 0
# repository.queueItemPollingBatchSize=0

# The XDS queue items resubmit polling interval to query queue items that
# were submitted for registration/delete/deprecate, but did not
# complete (registry_status=2).
# Set to 0 to process all documents.  (1 hour is .0416) (description)
# DEFAULT 0
# repository.queueItemResubmitPollingInterval=0

# ---------------------- HTTPS Related Properties ---------------------- #
#
#   The following properites must be configured for secure communication
#   These are used for keystore and truststore configuration.
# Keystore configurations (first 4 https configurations) are required
# if XUA is enabled
#
#  The final property in this section, https.ciphersuites is used
# to configure the suite used, a suitable value for this parameter
# is: TLS_RSA_WITH_AES_128_CBC_SHA
#
#   The properties in this section have NO DEFAULTS

# https.keyStore
# https.keyStorePassword
# https.server.keyAlias
# https.server.privateKeyPassword
# https.trustStore
# https.trustStorePassword
# https.ciphersuites
```

```
# ------------------  ATNA Audit Related Parameters ----------------- #
#
# Host name for the ATNA audit repository
# DEFAULT=localhost
# audit.host=localhost

# Port number for the ATNA audit repository.
# DEFAULT=514
# audit.port=514

# Transport type for the ATNA audit repository (BSD, TLS, UDP)
# DEFAULT=UDP
# audit.transport=UDP

# An optional source identifier for ATNA audit messages.
# NO DEFAULT
  audit.sourceId=${description}

# --------------------- XUA Related Properties ------------------------#

# flag to enable/disable Cross Enterprise User Assertion Validation
# Default value is false
# repository.xua.enabled=false

# The following properties are not required to be configured if
# XUA validation is disabled

# The validator to validate the SAML token in the request.
# The default value is com.emc.healthcare.xua.validator.XuaValidator
# xua.saml2.token.validator=com.emc.healthcare.xua.validator.XuaValidator

# The Crypto provider to be used for encryption/decryption and
# signature validation.
# The default value is org.apache.ws.security.components.crypto.Merlin
# xua.crypto.provider=org.apache.ws.security.components.crypto.Merlin

# The service endpoint regular expression to match against service
# endpoint attribute provided in the token.
# If not configured, the service endpoint provided in the token is
# not validated
# xua.service.endpoint


# The list of "," separated authentication methods supported by the
# XDS Repository.
# if not configured, Authentication method provided in the token is
# not validated
# xua.supported.authentication.methods

# The code system and the list of "," separated code values supported
# for the PurposeOfUseCode attribute provided in the token.
# if not configured, PurposeOfUseCode value provided in the token
# is not validated
# xua.purposeOfUse.codeSystem
# xua.purposeOfUse.code.values

# The code system and the list of "," separated code values supported
# for the Role attribute provided in the token.
# if not configured, Role value provided in the token is not validated
# xua.role.codeSystem
# xua.role.code.values

# The property to enable/disable Authorization Consent validation
# Default value is false
```

```
# xua.authz.consent.option=false

# Configuration of trusted assertion providers' certificates.
# It is a required configuration and has no default value.
# xua.assertion.provider.trustStore
# xua.assertion.provider.trustStorePassword

#---------------------- Soap Route Builder ----------------------------#
#
# An optional specification that allows users to override the
# default Soap RouteBuilder script provided with the product. The actual
# location of this within the classpath is within one of the Jar files
# shipped with the product. See description for soap.routeBuilderScriptSource
# above for details on how this can be used.
#
# DEFAULT=classpath:com/emc/healthcare/xds/repository/commons/SoapRouteBuilder.groovy
#soap.routeBuilderScriptSource=classpath:com/emc/healthcare/xds/repository/commons/SoapRouteBuilder.groovy


# Optional flags to disable incoming message validation. These flags
# may be used in special situations to disable
# incoming message validation. These flags are intended mainly for
# Connectathon testing in case a testing partner does
# not pass message validation. Messages that do not pass validation
# have a high likelyhood of failing for other reasons
# later in the processing cycle. These flags should always be set to
# "true" in production scenarios.
#
# DEFAULT=true
# repository.iti41.requestValidator.enabled=true

# DEFAULT=true
# repository.iti41.responseValidator.enabled=true

# DEFAULT=true
# repository.iti43.requestValidator.enabled=true

# DEFAULT=true
# repository.iti43.responseValidator.enabled=true

# RabbitMQ Configurations required if RabbitMQ is used to queue
# HL7 messages
# rabbitmq.host
# rabbitmq.port
# rabbitmq.username
# rabbitmq.password
# rabbitmq.virtualhost

# property to enable/disable TLS for Rabbitmq communication.
# If enabled, https properties should be configured.
# Default false
# rabbitmq.ssl.enabled

# ------------------------ PPIC Related Properties --------------------#

# flag to enable/disable PPIC
# Default value is false
#ppic.enabled=false


#The URL for making pdp service call for PPIC.
# Example: repository.pdpServiceURL=http://localhost:8080/ppic/pdp
ppic.pdpServiceUrl=http://localhost/ppic/pdp

# -------------------- Usage Report Properties ----------------------#
```

```
# User name to login to usage report User Interface
# Default is Administrator
#usagereport.username=

# User password to login to usage report User Interface
# Default is password
#usagereport.password=
```

hl7.properties

```
# -------------------- hl7.properties --------------------- #
# All settable properties for HL7 Processing are enumerated in
# this file and their default values.
#
# The order of property evaluation is as follows:
#
#   1: This file is consulted first.
#   2: ${com.emc.healthcare.home}/hl7.properties is consulted second
#   3: System.properties is consulted last.
#
# Property values are set in the order they are encountered. If
# the same property is defined in all three locations, the last takes
# precedence.
#
# ---------------------------------------------------------------- #

#HL7 General Properties

#--------------------------------------------------------------------
# HL7 recommends to use timezone offset in TimeStamp fields.
# Since this timezone information is optional so some application prefer
# to not include timezone offset.
#
# default =true
#--------------------------------------------------------------------

hl7.message.timestamp.includeTZOffset=true

#HL7 Inbound Message related properties.
#
# Currently Repository supports following HL7 Versions
# ADT^A34(V23), ADT^A40( V231,V24,V25,V251)
#--------------------------------------------------------------------

#--------------------------------------------------------------------
#InBound Mllp Listening port. If this is set to 0 ( Zero ) then
# Repository will not listen for incoming Messages.
# default =0
#--------------------------------------------------------------------

hl7.inbound.mllp.port=0

#--------------------------------------------------------------------

#--------------------------------------------------------------------
#InBound Secure Mllp Listening port. If this is set to 0 ( Zero )
# then Repository will not listen for incoming Messages.
# default =0
#--------------------------------------------------------------------
```

```
hl7.inbound.mllp.securePort=0


#-------------------------------------------------------------------------
# Due to some network disconnect/timeout issues, external systems can
# send same HL7 message twice to repository. If rejectDuplicateMessage
# flag is set to true then Repository will reject duplicate
# message and send AE( Application Error ) response code to Sender.
# default =true
#-------------------------------------------------------------------------
hl7.inbound.rejectDuplicateMessage=true

#-------------------------------------------------------------------------
# Folder Path to save inbound queue item.
# Default : /Temp
#
#-------------------------------------------------------------------------

hl7.inbound.queueitem.folderPath=/Temp

#-------------------------------------------------------------------------
# ACL for inbound HL7 queue item.
# Default : hip_repository_acl
#
#-------------------------------------------------------------------------

hl7.inbound.queueitem.acl=hip_repository_acl

#-------------------------------------------------------------------------
# ****** HL7 OutBound Message related properties. *******
#-------------------------------------------------------------------------

#-------------------------------------------------------------------------
# Outbound messages are in version specified in this property.
# Currently repository supports only
# MDM T01 message for following HL7V2 version.
# Supported versions : V23,V231,V24,V25,V251
# Supported MessageType/Event: MDM_T01,MDM_T02, MDM_T11
# DEFAULT = 2.3.1
#-------------------------------------------------------------------------
hl7.outbound.message.version=2.3.1

#-------------------------------------------------------------------------
# Outbound MLLP Server host. All outbound message will be sent on this host.
#-------------------------------------------------------------------------
hl7.outbound.mllp.host=localhost

#-------------------------------------------------------------------------
# Outbound MLLP Server port. All outbound message will be sent on this port.
# If port is set to 0( Zero ) then outbound feature will be disabled and
# no messages will be sent to external Systems.
#
#DEFAULT =0
#-------------------------------------------------------------------------
hl7.outbound.mllp.port=0

#-------------------------------------------------------------------------
# Request timeout. Repository will wait for configured milliseconds to
# received acknowledgement from External Systems. If response is not
# received then request will timeout.
#
#DEFAULT =3000
#-------------------------------------------------------------------------
hl7.outbound.requestTimeout=3000
```

```
#-------------------------------------------------------------------------------
# HL7 Out Queue Polling Interval. Repository will poll Docbase for new items
# as per interval set in this property.
# If set to Zero then Repository will not send any message to external system
#
#DEFAULT =0
#-------------------------------------------------------------------------------

hl7.outbound.queue.pollingInterval=0

#-------------------------------------------------------------------------------
# In Progress Items which are older than following value will be reset
# to "pending" and processed again.
# This helps reprocessing "in-progress" jobs which are lying in systems
# due to app server crash etc.
#DEFAULT =1
#-------------------------------------------------------------------------------

hl7.outbound.queue.reProcessInProgressItemsOlderThanDays=1

#-------------------------------------------------------------------------------
# HIP will try to redeliver outbound message as per count specified below
# if there was some network issues and message was not delivered in first
# attempt. for example, if count is 3 then HIP will try to redeliver
# message 3 times before it marks that item failed.
#
# DEFAULT = 3
#-------------------------------------------------------------------------------
hl7.outbound.queue.maxRedeliveries=3

#-------------------------------------------------------------------------------
# HIP will try to redeliver outbound message after some time as given in
# this property if message was not delivered
# in first attempt.
# DEFAULT = 60 ( in Seconds )
#-------------------------------------------------------------------------------
hl7.outbound.queue.reDeliveryDelay=60

#-------------------------------------------------------------------------------
# Sending Application Details.
#-------------------------------------------------------------------------------
hl7.outbound.sendingApplication.namespaceId=
hl7.outbound.sendingApplication.universalId=
hl7.outbound.sendingApplication.universalIdType=

#-------------------------------------------------------------------------------
# Sending Facility Details.
#-------------------------------------------------------------------------------
hl7.outbound.sendingFacility.namespaceId=
hl7.outbound.sendingFacility.universalId=
hl7.outbound.sendingFacility.universalIdType=


#-------------------------------------------------------------------------------
#
# Render Segments in HL7 Message even if there is no content
#-------------------------------------------------------------------------------

hl7.mdm.T01.renderEmptySegments=PV1-1
hl7.mdm.T02.renderEmptySegments=PV1-1
hl7.mdm.T11.renderEmptySegments=PV1-1

#-------------------------------------------------------------------------------
# HL7 MDM T01 Mapping
#-------------------------------------------------------------------------------
```

```
hl7.mdm.T01.MSH-7=import_date
hl7.mdm.T01.MSH-6=station_name
hl7.mdm.T01.PID-7=patient_birth_date
hl7.mdm.T01.PID-18=csn_number
hl7.mdm.T01.PID-2=NONE
hl7.mdm.T01.PID-3=patient_id
hl7.mdm.T01.PID-8=patient_sex
hl7.mdm.T01.PID-5=patient_name
hl7.mdm.T01.PV1-3=healthcare_facility_code
hl7.mdm.T01.PV1-19=account_number
hl7.mdm.T01.TXA-2=record_type
hl7.mdm.T01.TXA-4=admission_date
hl7.mdm.T01.TXA-5=provider_mcr
hl7.mdm.T01.TXA-6=investigation_date
hl7.mdm.T01.TXA-17=completion_status
hl7.mdm.T01.TXA-12=r_object_id
hl7.mdm.T01.TXA-16=r_object_id

#-----------------------------------------------------------------------------
# HL7 MDM T02 Mapping
#-----------------------------------------------------------------------------
hl7.mdm.T02.MSH-7=r_creation_date
hl7.mdm.T02.MSH-6=station_name
hl7.mdm.T02.PID-7=patient_birth_date
hl7.mdm.T02.PID-18=csn_number
hl7.mdm.T02.PID-3=patient_id
hl7.mdm.T02.PID-2=NONE
hl7.mdm.T02.PID-8=patient_sex
hl7.mdm.T02.PID-5=patient_name
hl7.mdm.T02.PV1-3=healthcare_facility_code
hl7.mdm.T02.PV1-19=account_number
hl7.mdm.T02.TXA-2=record_type
hl7.mdm.T02.TXA-4=admission_date
hl7.mdm.T02.TXA-5=provider_mcr
hl7.mdm.T02.TXA-6=investigation_date
hl7.mdm.T02.TXA-17=completion_status
hl7.mdm.T02.TXA-16=r_object_id
hl7.mdm.T02.TXA-12=r_object_id
hl7.mdm.T02.OBX-5=r_object_id
hl7.mdm.T02.NTE-3=title
#-----------------------------------------------------------------------------
# HL7 MDM T11 Mapping
#-----------------------------------------------------------------------------

hl7.mdm.T11.MSH-7=r_creation_date
hl7.mdm.T11.MSH-6=station_name
hl7.mdm.T11.PID-7=patient_birth_date
hl7.mdm.T11.PID-18=NONE
hl7.mdm.T11.PID-3=patient_id
hl7.mdm.T11.PID-2=NONE
hl7.mdm.T11.PID-8=patient_sex
hl7.mdm.T11.PID-5=patient_name
hl7.mdm.T11.PV1-3=NONE
hl7.mdm.T11.PV1-19=NONE
hl7.mdm.T11.TXA-2=record_type
hl7.mdm.T11.TXA-4=NONE
hl7.mdm.T11.TXA-5=NONE
hl7.mdm.T11.TXA-6=NONE
hl7.mdm.T11.TXA-17=NONE
hl7.mdm.T11.TXA-16=r_object_id
hl7.mdm.T11.TXA-12=r_object_id

#-----------------------------------------------------------------------------
# Queue Related Properties
#-----------------------------------------------------------------------------
```

```
#Default is false
#hl7.queue.enabled

#Below properties are used when queue flag is enabled.
#Default is hl7MergeExchange
#hl7.merge.exchange
#Default is hl7OutboundExchange
#hl7.outbound.exchange

#Default is hl7MergeQueue
#hl7.merge.queue=
#Default is hl7FailedQueue
#hl7.failed.queue

#HL7 EOB Message Configurations. Defaults to empty values
#repository.eob.scanType
#repository.eob.hospitalLocation
#repository.eob.batchUser
#repository.eob.batchMode
#repository.eob.postingDepartment
```

hip-dctm-mapping.properties

```
#Document Object Type
document.object.type=dmh_document

#Document Properties
document.authors=authors
document.availabilityStatus=availability_status
document.class.code=class_code
document.comments=comments
document.confidentiality.code=confidentiality_code
document.creationTime=creation_time
document.entryUuid=entry_uuid
document.event.code=event_code
document.format.code=format_code
document.healthcareFacility.code=healthcare_facility_code
document.language.code=language_code2
document.legalAuthentication=legal_authenticator_id
document.mimeType=mime_type
document.patientId.id=patient_id
document.patientId.assigningAuthority=patient_id_issuer
document.practiceSetting.code=practice_setting_code
document.serviceStartTime=service_start_time
document.serviceStopTime=service_stop_time
document.source.patientId=source_patient_id
document.title=title
document.type.code=type_code
document.uniqueId=unique_id
document.uri=uri

#Author Object Type
author.object.type=dmh_author

#Author Properties
author.person.id=person_id
author.familyName=family_name
author.givenName=given_name
author.secondName=second_name
```

```
author.prefixName=prefix_name
author.suffixName=suffix_name
author.degree=degree
author.specialties=specialties
author.institutions=organizations
author.contacts=contacts
author.roles=roles

#Person Object Type
person.object.type=dmh_person

#Person Properties
person.id=person_id
person.familyName=family_name
person.givenName=given_name
person.secondName=second_name
person.prefixName=prefix_name
person.suffixName=suffix_name
person.degree=degree

#Code Object Type
code.object.type=dmh_code

#Code Properties
code.value=code
code.name=code_name
code.scheme=code_schema

#Patient Folder Object Type
folder.object.type=dmh_patient_folder

#patient Folder Properties
folder.patient.name=patient_name
folder.patient.birthDate=patient_birth_date
folder.patient.sex=patient_sex
folder.patient.id=patient_id
```

hip-ppic-mapping.properties

```
request.subjectId=urn:oasis:names:tc:xacml:1.0:subject:subject-id
request.subjectRole=urn:oasis:names:tc:xacml:2.0:subject:role
```

mimetype.properties

```
#sample mimetype mapping for Documentum Format values.
#mimeType=format
text/asp=asp
text/plain=crtext
text/xml=xml
text/css=css
text/dtd=dtd
text/html=html
text/hdml=hdml
text/jhtml=jhtml
text/java=java
```

```
text/js=js
text/opml=opml
text/php=php
text/phtml=phtml
text/wsdl=wsdl
application/pdf=pdf
audio/basic=audio
application/x-macbinary=bin
image/bmp=bmp
application/cgi=cgi
image/cgm=cgm
java/*=class
image/x-canon-cr2=cr2
image/x-fpx=fpx
image/gif=gif
image/jpeg=jpeg
image/x-pcd=pcd
image/png=png
image/x-portable-anymap=pnm
image/x-portable-pixmap=ppm
image/x-rle=rle
image/svg+xml=sv
image/x-targa=tga
image/tiff=tiff
image/xbm=xbm
application/photoshop=photoshop8
video/x-gxf=gxf
application/msword=msw6
```

XDS Registration XML

```
<submissionSet>
   <author>
      <authorInstitution>
  <id>Hospital^^^^^^^^^^1.2.3.4.5.6.7.8.9.1789.45</id>
  <name>My Hospital</name>
      </authorInstitution>
      <authorPerson>
   <id>123</id>
    <name>
       <given>Tom</given>
       <family>Thumb</family>
    </name>
      </authorPerson>
      <authorRole>name of role</authorRole>
      <authorSpecialty>specialty of author</authorSpecialty>
      <authorTelecommunication>telephone of author</authorTelecommunication>
   </author>
   <availabilityStatus>Approved</availabilityStatus>
   <comments>Annual physical</comments>
   <contentTypeCode>
      <code>123</code>
      <systemName>content</systemName>
      <displayName>Content Type</displayName>
   </contentTypeCode>
   <entryUuid>urn:uuid:fd6f1187-bad7-417a-8b3b-9f58d229eca4</entryUuid>
   <homeCommunityId> urn:oid:1.19.6.24.109.42.1.2</homeCommunityId>
   <patientId>144ba3c4aad24e9^^^&1.3.6.1.4.1.21367.2005.3.7&ISO"</patientId>
   <sourceId>1.3.6.1.4.1.21367.2.2</sourceId>
   <submissionTime>2004-12-25T23:50:50Z</submissionTime>
   <title language="en-US">Physical</title>
```

```
   <uniqueId>1.42.20131114150734.7</uniqueId>
   <version>
      <versionName>1</versionName>
   </version>
</submissionSet>

<association>
   <associationType>HasMember</associationType>
   <entryUuid>urn:uuid:c0ff55a7-e0eb-4f27-bb22-90c46dc766b3</entryUuid>
   <label>Original</label>
   <sourceUuid>urn:uuid:fd6f1187-bad7-417a-8b3b-9f58d229eca4</sourceUuid>
   <targetUuid>urn:uuid:00287f52-c884-4ab0-8867-a6c761b6a298</targetUuid>
</association>

<folder>
   <availabilityStatus>Approved</availabilityStatus>
   <codeList>
      <code>123</code>
      <systemName>system name</systemName>
      <displayName>Code Name</displayName>
   </codeList>
   <comments>comments</comments>
   <entryUuid>urn:uuid:f7f401ac-45f7-4be8-bac6-e16117a231d4</entryUuid>
   <homeCommunityId> urn:oid:1.19.6.24.109.42.1.2</homeCommunityId>
   <lastUpdateTime>2014-01-03T19:01:51Z</lastUpdateTime>
   <patientId>144ba3c4aad24e9^^^&1.3.6.1.4.1.21367.2005.3.7&ISO"</patientId>
   <title>title</title>
   <uniqueId>1.2.3.6.555845363.4362715096841694297</uniqueId>
   <version>
      <versionName>1</versionName>
   </version>
</folder>


<documentEntry>
   <author>
      <authorInstitution>
  <id>Hospital^^^^^^^^^1.2.3.4.5.6.7.8.9.1789.45</id>
  <name>My Hospital</name>
      </authorInstitution>
      <authorPerson>
   <id>123</id>
    <name>
       <given>Tom</given>
       <family>Thumb</family>
    </name>
      </authorPerson>
      <authorRole>name of role</authorRole>
      <authorSpecialty>specialty of author</authorSpecialty>
      <authorTelecommunication>telephone of author</authorTelecommunication>
   </author>
   <availabilityStatus>Approved</availabilityStatus>
   <classCode>
      <code>34133</code>
      <systemName>2.16.840.1.113883.6.1</systemName>
      <displayName>Summary of Episode Note</displayName>
   </classCode>
   <comments>comments</comments>
   <confidentialityCode>
      <code>N</code>
      <systemName >2.16.840.1.113883.5.25</systemName >
      <displayName>Normal sensitivity</systemName>
   </confidentialityCode>
   <creationTime>20061224</creationTime>
   <entryUuid>urn:uuid:14a9fdec-0af4-45bb-adf2-d752b49bcc7d</entryUuid>
```

```
  <eventCodeList>
     <code>evc-23</code>
     <systemName>event code list</systemName>
     <displayName>Events</displayName>
  </eventCodeList>
  <formatCode>
     <code>urn:ihe:lab:xd-lab:2008</code>
     <systemName>1.3.6.1.4.1.19376.1.2.3</systemName>
     <displayName>XD-Lab</displayName>
  </formatCode>
  <hash>e543712c0e10501972de13a5bfcbe826c49feb75</hash>
  <healthcareFacilityTypeCode>
     <code>394802001</code>
     <systemName>2.16.840.1.113883.6.96</systemName>
     <displayName>General Medicine</displayName>
  </healthcareFacilityTypeCode>
  <homeCommunityId>urn:oid:1.19.6.24.109.42.1.2</homeCommunityId>
  <languageCode>en-us</languageCode>
  <legalAuthenticator>
     <id></id>
     <name>
 <given>Tom</given>
 <family>Thumb</family>
     </name>
  </legalAuthenticator>
  <mimeType>text/xml</mimeType>
  <patientId>144ba3c4aad24e9^^^&1.3.6.1.4.1.21367.2005.3.7&ISO</patientId>
  <practiceSettingCode>
     <code>394802001</code>
     <systemName>2.16.840.1.113883.6.96</systemName>
     <displayName>General Medicine</displayName>
  </practiceSettingCode>
  <referenceIdList>
     <referenceId>144ba3c4aad24e9^^^&1.3.6.1.4.1.21367.2005.3.1&ISO</referenceId>
     <referenceId>144ba3c4aad24e9^^^&1.3.6.1.4.1.21367.2005.3.2&ISO</referenceId>
     <referenceId>144ba3c4aad24e9^^^&1.3.6.1.4.1.21367.2005.3.3&ISO</referenceId>
  </referenceIdList>
  <repositoryUniqueId>1.2.3.4</repository.uniqueId>
  <serviceStartTime>200612230800</serviceStartTime>
  <serviceStopTime>200612230900</ServiceStopTime>
  <size>350</size>
  <sourceId>4567856789^^^&3.4.5&ISO</sourceId>
  <sourcePatientId>4567856789^^^&3.4.5&ISO</sourcePatientId>

  <sourcePatientInfo>
     <id>123^^^&1.3.6.1.4.1.21367.2005.3.3</id>
     <name>
  <given>Tom</given>
  <family>Thumb</family>
     </name>
     <gender>M</gender>
     <birthDate>20/10/1990</birthDate>
     <birthPlace>Pleasanton, CA</birthPlace>
     <address>
        <street>123 Main St.</street>
        <city>Pleasanton</city>
        <stateOrProvince>CA</stateOrProvince>
        <postalCode>94566<postalCode>
        <country>USA</country
     </address>
     <telecom>19256001234</telecom>
     <email>tom.thumb@gmail.com<email>
  </sourcePatientInfo>
  <title>document title</title>
  <typeCode>
```

```
      <code>1</code>
      <systemName>1</systemName>
      <displayName>1</displayName>
   </typeCode>
   <uniqueId>1.2009.0827.08.33.5074</uniqueId>
   <uri>uri</uri>
   <version>
      <versionName>1</versionName>
   </version>
</documentEntry>
```

XDS Registration Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.emc.com/healthcare/commons/automaticRegistrationModel"
   elementFormDefault="unqualified"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:tns="http://www.emc.com/healthcare/commons/automaticRegistrationModel">

    <xs:element name="automaticRegistrationConfig">
   <xs:complexType>
    <xs:sequence>
   <xs:element ref="tns:submissionSet" minOccurs="1" maxOccurs="1"/>
   <xs:element ref="tns:association"  minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="tns:folder" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="tns:documentEntry"  minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
   </xs:complexType>
    </xs:element>

    <!-- submission set schema -->
    <xs:element name="submissionSet">
   <xs:complexType>
  <xs:sequence>
   <xs:element ref="tns:author" minOccurs="1" maxOccurs="1"/>
   <xs:element ref="tns:availabilityStatus" maxOccurs="1" minOccurs="0"></xs:element>
   <xs:element ref="tns:comments" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="contentTypeCode" type="tns:codes" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element ref="tns:customMetadata" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="tns:entryUuid" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element ref="tns:homeCommunityId" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="patientId" type="tns:LongName" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="sourceId" type="xs:string" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="submissionTime" type="xs:string" maxOccurs="1" minOccurs="0"></xs:element>
   <xs:element ref="tns:title" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element ref="tns:uniqueId" minOccurs="1" maxOccurs="1"></xs:element>
   <xs:element ref="tns:version" minOccurs="0" maxOccurs="1"></xs:element>
  </xs:sequence>
   </xs:complexType>
    </xs:element>

    <!-- association set element -->
    <xs:element name="association">
   <xs:complexType>
  <xs:sequence>
   <xs:element name="associationType" type="xs:string" minOccurs="1" maxOccurs="1"></xs:element>
   <xs:element ref="tns:customMetadata" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="tns:entryUuid" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="label" type="xs:string" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="sourceUuid" type="xs:string" maxOccurs="1" minOccurs="1"></xs:element>
```

```
 <xs:element name="targetUuid" type="xs:string" maxOccurs="1" minOccurs="1"></xs:element>
</xs:sequence>
 </xs:complexType>
  </xs:element>

  <!-- folder element -->
  <xs:element name="folder">
 <xs:complexType>
<xs:sequence>
 <xs:element ref="tns:availabilityStatus" maxOccurs="1" minOccurs="0"/>
 <xs:element name="codeList" type="tns:codes" maxOccurs="unbounded" minOccurs="0"></xs:element>
 <xs:element ref="tns:comments" minOccurs="0" maxOccurs="1"></xs:element>
 <xs:element ref="tns:customMetadata" minOccurs="0" maxOccurs="unbounded"/>
 <xs:element ref="tns:entryUuid" minOccurs="1" maxOccurs="1"></xs:element>
 <xs:element ref="tns:homeCommunityId" minOccurs="0" maxOccurs="1"></xs:element>
 <xs:element name="lastUpdateTime" type="xs:string" maxOccurs="1" minOccurs="0"/>
 <xs:element name="patientId" type="tns:LongName" maxOccurs="1" minOccurs="1"/>
 <xs:element ref="tns:title" minOccurs="0" maxOccurs="1"></xs:element>
 <xs:element ref="tns:uniqueId" minOccurs="1" maxOccurs="1"/>
 <xs:element ref="tns:version" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
 </xs:complexType>
  </xs:element>

  <!-- Document entry element -->
  <xs:element name="documentEntry">
 <xs:complexType>
  <xs:sequence>
 <xs:element ref="tns:author" minOccurs="1" maxOccurs="1"></xs:element>
   <xs:element ref="tns:availabilityStatus" maxOccurs="1" minOccurs="0"/>
   <xs:element name="classCode" type="tns:codes" minOccurs="1" maxOccurs="1"/>
   <xs:element ref="tns:comments" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="confidentialityCode" type="tns:codes" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="creationTime" type="xs:string" minOccurs="1" maxOccurs="1"></xs:element>
   <xs:element ref="tns:customMetadata" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="tns:entryUuid" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="eventCodeList" type="tns:codes" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="formatCode" type="tns:codes" maxOccurs="1" minOccurs="1"/>
   <xs:element name="hash" type="tns:LongName" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="healthcareFacilityTypeCode" type="tns:codes" maxOccurs="1" minOccurs="1"/>
   <xs:element ref="tns:homeCommunityId" minOccurs="0" maxOccurs="1"></xs:element>
   <xs:element name="languageCode" type="xs:string" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="legalAuthenticator" type="tns:authorPerson" maxOccurs="1" minOccurs="0"/>
   <xs:element name="mimeType" type="xs:string" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="patientId" type="tns:LongName" maxOccurs="1" minOccurs="1"/>
   <xs:element name="practiceSettingCode" type="tns:codes" maxOccurs="1" minOccurs="1"/>
   <xs:element name="referenceIdList" maxOccurs="1" minOccurs="0">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="referenceId" type="tns:LongName" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
   </xs:element>
   <xs:element name="repositoryUniqueId" type="xs:string" maxOccurs="1" minOccurs="0"></xs:element>
   <xs:element name="serviceStartTime" type="xs:string" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="serviceStopTime" type="xs:string" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="size" type="xs:long" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="sourceId" type="xs:string" maxOccurs="1" minOccurs="0"></xs:element>
   <xs:element name="sourcePatientId" type="tns:LongName" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="sourcePatientInfo" maxOccurs="1" minOccurs="1">
  <xs:complexType>
    <xs:sequence>
   <xs:element ref="tns:id"  maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="name" type="tns:name" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="gender" type="tns:String16" maxOccurs="1" minOccurs="1"></xs:element>
```

```
  <xs:element name="birthDate" type="tns:String16" maxOccurs="1" minOccurs="1"></xs:element>
  <xs:element name="birthPlace" type="tns:String32" maxOccurs="1" minOccurs="1"></xs:element>
  <xs:element name="address" maxOccurs="1" minOccurs="1">
   <xs:complexType>
  <xs:sequence>
   <xs:element name="street" type="tns:ShortName" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="city" type="tns:String32" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="stateOrProvince" type="tns:String32" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="postalCode" type="tns:String16" maxOccurs="1" minOccurs="1"></xs:element>
   <xs:element name="country" type="tns:String32" maxOccurs="1" minOccurs="1"></xs:element>
  </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="telecom" type="xs:string"></xs:element>
  <xs:element name="email" type="xs:string"></xs:element>
   </xs:sequence>
 </xs:complexType>
  </xs:element>
  <xs:element ref="tns:title" minOccurs="0" maxOccurs="1"></xs:element>
  <xs:element name="typeCode" type="tns:codes" minOccurs="1" maxOccurs="1"/>
  <xs:element ref="tns:uniqueId" minOccurs="1" maxOccurs="1"></xs:element>
  <xs:element name="uri" type="tns:LongName" minOccurs="1" maxOccurs="1"></xs:element>
  <xs:element ref="tns:version" minOccurs="0" maxOccurs="1"/>
 </xs:sequence>
</xs:complexType>
 </xs:element>
 <xs:element name="uniqueId" type="xs:string"></xs:element>
 <xs:element name="entryUuid" type="tns:LongName"></xs:element>
 <xs:element name="homeCommunityId" type="xs:string" ></xs:element>
 <xs:element name="logicalUuid" type="xs:string"></xs:element>

 <!--Define Author type -->
 <xs:element name="author">
<xs:complexType>
 <xs:sequence>
<xs:element name="authorInstitution" maxOccurs="unbounded" minOccurs="1">
 <xs:complexType>
   <xs:sequence>
  <xs:element ref="tns:id" maxOccurs="1" minOccurs="1"></xs:element>
  <xs:element name="name" type="xs:string" maxOccurs="1" minOccurs="1"></xs:element>
   </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="authorPerson" type="tns:authorPerson" maxOccurs="1" minOccurs="1"/>
<xs:element name="authorRole" type="tns:LongName" maxOccurs="unbounded" minOccurs="1"></xs:element>
<xs:element name="authorSpecialty" type="tns:LongName" maxOccurs="unbounded" minOccurs="1"></xs:element>
<xs:element name="authorTelecommunication" type="tns:LongName" maxOccurs="unbounded"
 minOccurs="1"></xs:element>
 </xs:sequence>
</xs:complexType>
 </xs:element>

 <!--Define Author Person -->
 <xs:complexType name="authorPerson">
<xs:sequence>
 <xs:element ref="tns:id" maxOccurs="1" minOccurs="1"></xs:element>
 <xs:element name="name" type="tns:name" maxOccurs="1" minOccurs="1"></xs:element>
</xs:sequence>
 </xs:complexType>
 <xs:element name="id" type="tns:LongName"></xs:element>
 <xs:complexType name="name">
<xs:sequence>
 <xs:element name="given" type="tns:ShortName"></xs:element>
 <xs:element name="family" type="tns:ShortName"></xs:element>
</xs:sequence>
```

```
 </xs:complexType>

 <xs:element name="comments" type="tns:LongName"></xs:element>
 <xs:element name="title" type="tns:LongName"></xs:element>

 <!-- Define availability status -->
 <xs:element name="availabilityStatus" type="tns:String16"></xs:element>

 <!-- Defining Codes -->
 <xs:complexType name="codes">
<xs:sequence>
 <xs:element ref="tns:code" maxOccurs="1" minOccurs="1"/>
 <xs:element ref="tns:systemName" maxOccurs="1" minOccurs="1"/>
 <xs:element ref="tns:displayName" maxOccurs="1" minOccurs="1"/>
</xs:sequence>
 </xs:complexType>
 <xs:element name="code" type="tns:LongName"></xs:element>
 <xs:element name="systemName" type="tns:LongName" ></xs:element>
 <xs:element name="displayName" type="tns:LongName"></xs:element>

 <!--custom metadata element-->
 <xs:element name = "customMetadata" type = "tns:customMetadataType"/>
 <xs:complexType name = "customMetadataType">
<xs:sequence>
 <xs:element ref = "tns:ValueList" minOccurs = "1" maxOccurs="1"/>
</xs:sequence>
<xs:attribute name = "name" use = "required" type = "xs:string"/>
 </xs:complexType>

 <xs:complexType name = "ValueListType">
<xs:sequence minOccurs = "0" maxOccurs = "unbounded">
 <xs:element ref = "tns:Value" />
</xs:sequence>
 </xs:complexType>
 <xs:element name = "ValueList" type = "tns:ValueListType"/>
 <xs:element name = "Value" type = "xs:string"/>

 <!--version element-->
 <xs:element name="version">
<xs:complexType>
 <xs:sequence>
<xs:element name="versionName" type="xs:string" minOccurs="1" maxOccurs="1"></xs:element>
 </xs:sequence>
</xs:complexType>
 </xs:element>

 <!-- Define Data Types -->
 <xs:simpleType name="LongName">
<xs:restriction base="xs:string">
 <xs:maxLength value="256"/>
</xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="ShortName">
<xs:restriction base="xs:string">
 <xs:maxLength value="64"/>
</xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="String16">
<xs:restriction base="xs:string">
 <xs:maxLength value="16"/>
</xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="String32">
<xs:restriction base="xs:string">
 <xs:maxLength value="32"/>
```

```
   </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

Index

L

M

N

O

P

R

S

T

U

V

W

X