EMC® XDS Repository Connectors

Version 1.9

Administration 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 ECD.Documentation.Feedback@emc.com

Table of Contents

Revision History

Revision Date	Description
February 2016	Initial publication.

Chapter 1

About XDS Repository Connectors

This chapter provides an overview of Healthcare Integration Portfolio (HIP) Cross-enterprise Document Sharing (XDS) Repository Connectors, their architecture, workflow, and endpoints.

Overview

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

HIP provides the following XDS Repository Connectors:

- XDS Repository Connector for Documentum
- XDS Repository Connector for xDB

XDS Repository Connector for each repository consists of a server that supports:

- A database that serves 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
- Trusted Node Authentication through TLS certificates

The XDS Repository Connector for Documentum specifically supports:

- Documentum specific processors to register or 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)
- 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.

The XDS Repository Connector for xDB supports:

- Storing and managing healthcare content, both XML and non-XML documents, in xDB or Swift-based Elastic Cloud Service (ECS) blobstores.
- Storing and managing Clinical Document Architecture (CDA) or XML documents in the xDB XML store
- Utilizing xDB capabilities, if required, to search and perform analytics, on XML documents by using XML tools such as xquery, xpath and so on
- Performing IHE XDS related transactions, that is, ITI-41, ITI-42, ITI-43
- Registering all clinical documents that are submitted to the xDB Repository through the xDB Repository Server, with XDS Registry
- Trusted Node Authentication through TLS certificates
- A blob store, either xDB or ECS, in which all documents are stored

Architecture

The XDS Repository Connectors consist of the following components:

- A data server that stores all XDS document content
- An XDS Repository Server

The following are the data servers for different XDS Repository Connectors:

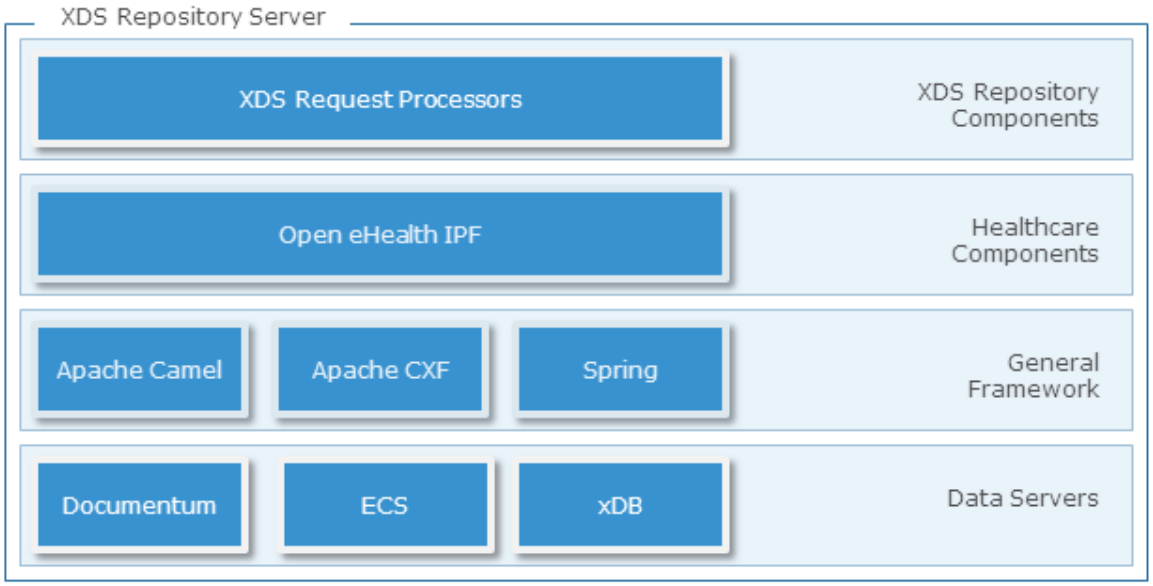
- **Documentum**: The data server for XDS Repository Connector for Documentum
- **ECS**: The data server for XDS Repository Connector for ECS
- **xDB**: The data server for XDS Repository Connector for xDB

The XDS Repository Server is a J2EE web application built on the Apache Camel open-source routing and mediation framework.

You can deploy both components on the same system or on separate systems. EMC recommends deploying the components in separate systems to optimize performance and scalability.

Load Balancing and Scalability, page 18 provides more information on scalability and load balancing.

The following figure shows the XDS Repository Server architecture:



XDS Request Processors:

This layer contains XDS Request Processors that handle and process request messages.

For XDS Repository Connector for Documentum, these processors handle messages for generating Documentum Content Server requests using Documentum Foundation Classes (DFC) and for retrieving documents from the Documentum Content Server.

Openhealth IPF:

This 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.

Application Framework:

This 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 different formats. This layer also uses the Spring ApplicationContext to provide configuration properties to the application.

Data Server:

This layer consists of the data servers for the respective connector.

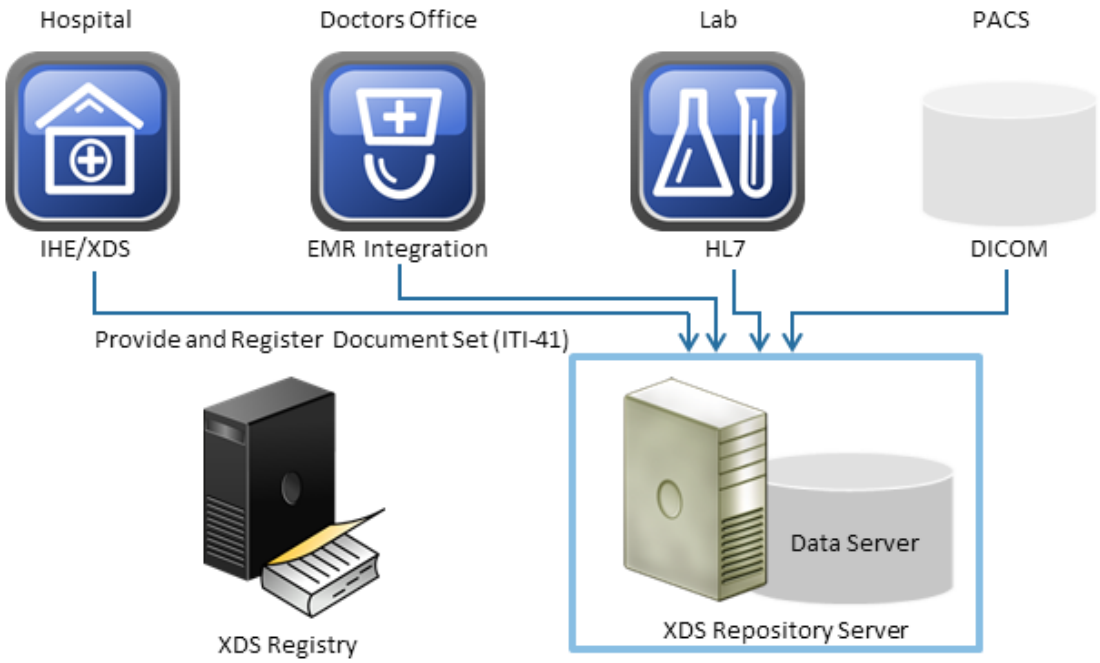
Workflow

A healthcare community consists of different healthcare consumers and providers who 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, and so on).

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).

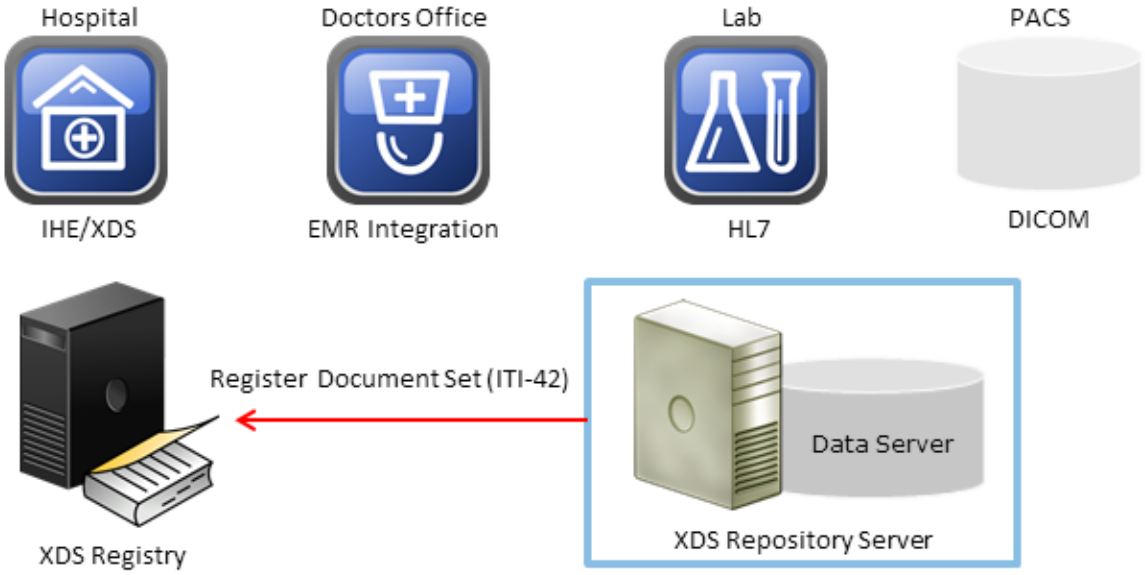
Healthcare providers use the `Provide and Register Document Set` transaction (ITI-41) to submit new healthcare records to XDS Repository. XDS Repository stores the submitted healthcare records in Documentum.

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



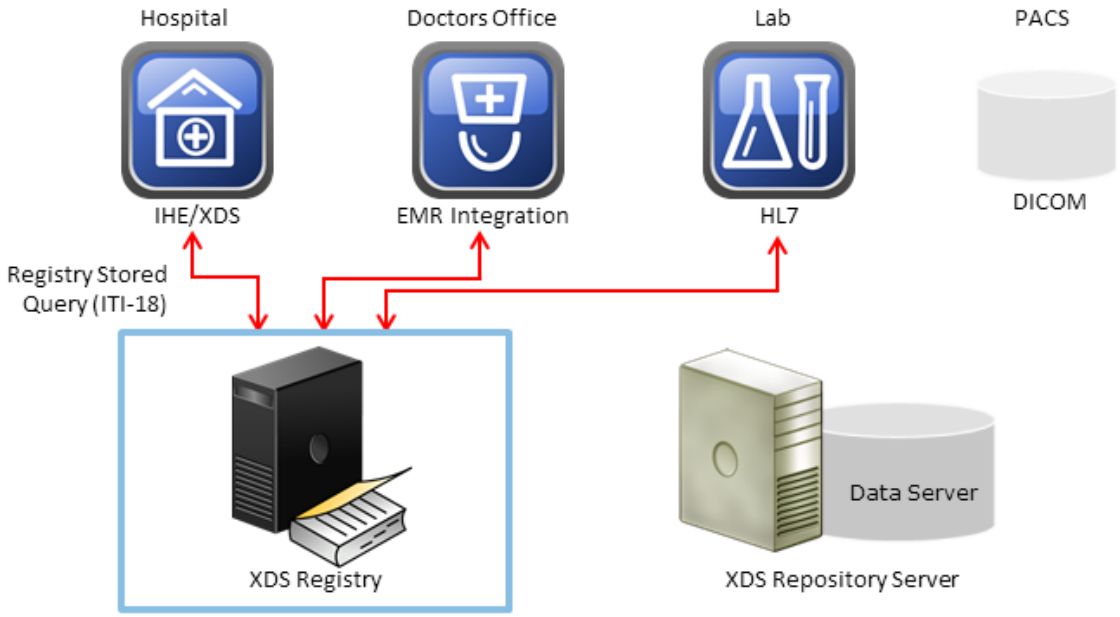
The XDS Repository Server then automatically registers the relevant document metadata with the XDS Registry. This is done using the `Register Document Set` transaction (ITI-42). Registering a healthcare record with the XDS Registry enables other healthcare providers to find the record.

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



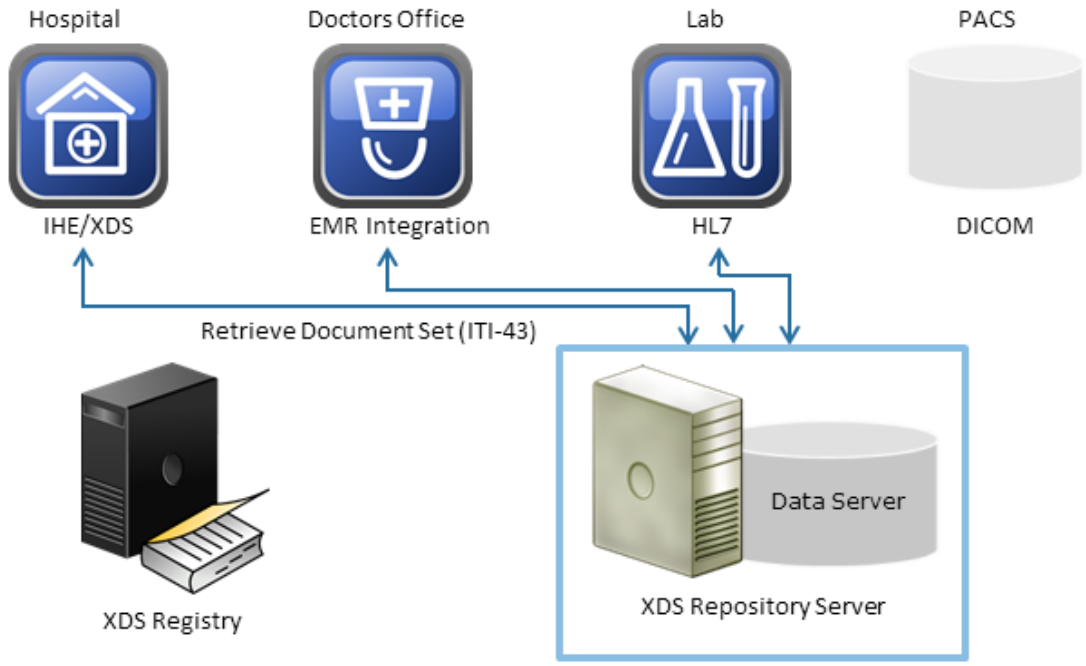
When healthcare providers need to obtain a patient's healthcare records, they query the Registry to get a list of healthcare records of that patient and their repository locations. The Registry is queried with the `Registry Stored Query` request (ITI-18). The Registry provides locations of the healthcare 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 that the XDS Repository Server provides for ITI-41 and ITI-43 transactions:

IHE Transaction	Description	Endpoint
ITI-41	Provide and Register Document Set	https://<*host:port*>/repository/services/xds-iti41
ITI-41as	Provide and Register Document Set Asynchronous	https://<*host:port*>/repository/services/xds-iti41as
ITI-43	Retrieve Document Set	https://<*host:port*>/repository/services/xds-iti43
ITI-43as	Retrieve Document Set Asynchronous	https://<*host:port*>/repository/services/xds-iti43as

For client applications such as XDS consumers that support only single endpoint for XDS operations, HIP provides an additional single SOAP endpoint for IHE transactions. This additional endpoint dispatches request to the original endpoint (as defined in SOAPEndpointRouteBuilder) based on the SOAP operation name in the request.

The following table lists the single endpoints (applicable to both http and https) provided by the XDS Repository Server for ITI-41 and ITI-43 transactions:

Type of Operation	Unified Endpoint URL	Supported Transactions
Synchronous operations	http://<*host:port*>/repository/services/xds-repo-svc	• **ITI-41**: http://<*host:port*>/repository/services/xds-iti41 • **ITI-43**: http://<*host:port*>/repository/services/xds-iti43
Asynchronous operations	http://<*host:port*>/repository/services/xds-repo-svc-as	• **ITI-41as**: http://<*host:port*>/repository/services/xds-iti41as • **ITI-43as**: http://<*host:port*>/repository/services/xds-iti43as

Chapter 2

Features

This chapter describes the features of XDS Repository Connectors.

Common Features of XDS Repository Connectors

The following features are common for all XDS Repository Connectors:

- XDS Repository Transactions
- Business Continuance
- Usage Reporting

XDS Repository Transactions

XDS Repository Connectors support 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 Document Repository, and to request Document Repository to store these documents, and register them with Document Registry.
- **ITI-42 Register Document Set**: Transaction [ITI-42] is used by a Document repository Actor to register a set of documents with Document Registry in XDS.b.

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

Business Continuance

The information provided in this topic applies only to the XDS Repository Server.

The Documentum product documentation provides more 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 the 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 the available XDS Repository Servers to ensure that the system remains available.
- Configuring server connections to the XDS repositories: Each XDS Repository Server connects to a single XDS repository, but a single XDS repository can accept connections from multiple XDS Repository Servers. You can instantiate multiple XDS Repository Servers to serve the same or different repositories.

Data Backup and Recovery

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

For example:

- **Spare Instance**: Create spare instances of the 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 fails and another server replaces it, the components may have difficulty connecting to the new server. Re-configure 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. Standard Usage Reports are scheduled to be automatically generated on the last day of each month. However, you can also generate adhoc reports.

HIP provides a Usage Reporting web application that provides the ability to view the monthly reports and generate ad hoc reports.

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

`localhost:port/repository/reports`

The default username and password to log in to the web application are configured in the `<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:

- For XDS Repository Connector for Documentum:
  - **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.

    The usage of Repository is calculated based on the number of documents stored and number of unique patients in the Repository.

    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.
- For XDS Repository Connector for xDB:

  Storage Volume: Volume occupied by the documents stored in the Blob store.

  The usage of xDB Repository is calculated based on the storage volume utilized by the documents in the Blob store.

  The monthly reports are stored in the `{Blob folder}/usagereports` folder.
- **Generated on**: Date and time when the report is generated.
- **Generated By**: Name of the user who generates the reports.

- **Host**: IP address of the host system 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 ad hoc reports that show the usage details from the day of product purchase.

The names 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 37 provides information on configuring the Usage Reporting properties.

Features of XDS Repository Connector for Documentum

The XDS Repository Connector for Documentum provides the following features:

- Patient Identity Notifications
- Patient Privacy Policy Enforcement
- HIP Customizations
- Message Queue
- Documentum Integrations
- EMR Integrations

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 by 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 (denoted by ADT^A40). The Patient Identity Source generates the **Merge Patient–Internal ID** notification 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. You must set the

HTTPS properties if you want to enable the secure HTTPS MLLP port. You can edit the `cs-repository.properties` file to configure these ports.

Patient Privacy Policy Enforcement

The Patient Privacy Policy Enforcement is applicable only for XDS Repository Connector for Documentum.

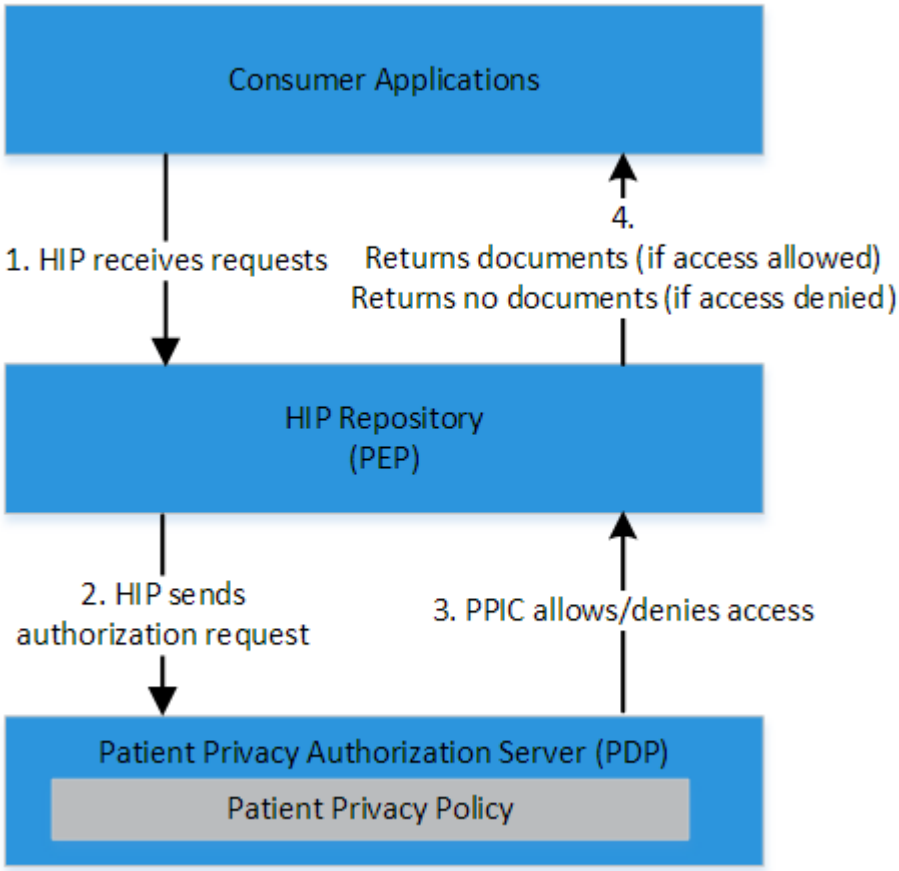
Users require authorization to access a patient's healthcare metadata and documents in an XDS Affinity Domain. A user is permitted to view only those documents for which permission is granted by the patient privacy policy. HIP implements the authorization feature through the Patient Privacy and Informed 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 the 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 the SAML token. The PEP components pass 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 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 PPIC, page 40 provides more information on the PPIC configuration properties.

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

HIP Customizations

HIP supports the following customizations:

- Customization for Message Routes
- Custom Extension for Document Creation
- PnR Custom Extensions
- Custom HL7 Message Processing

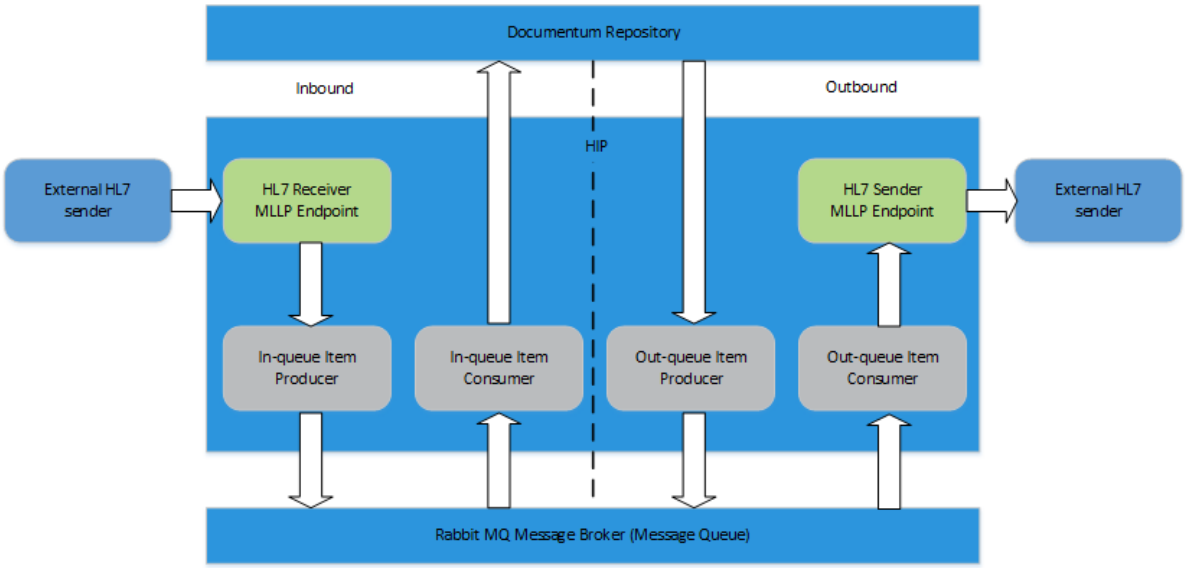
Chapter 4, Customizing XDS Repository Connector for Documentum provides more information on HIP customization.

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 the 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 73 provides more information on support for custom HL7 messages.

Configuring the HL7 Message Queue Properties, page 59 provides more information on message queue configuration.

HL7 Messages

The HL7 messaging standards define the standards to package and communicate information 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

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),ADT A01,ADT A04,ADT A05,ADT A08 and ADT A31`

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

Some clinical systems send medical records in the OBX segment of the HL7 MDM T02 message. HIP Repository receives the MDM T02 message, extracts medical record from the OBX segment and stores it as an instance of the `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 Process HL7 MDM T02 Inbound Messages, page 75 provides more information 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 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.
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. The 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 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 59 provides more information about the configuration of HL7 EOB messages.

HL7 Out-Queue Item Object Type

The XDS Repository Server can send HL7 messages to the 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 then 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.

Name	Type	Description
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

This section describes the Documentum integrations supported by HIP Repository.

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` 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 50 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 value 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`.

mimetype.properties, page 90 provides a sample of the `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.

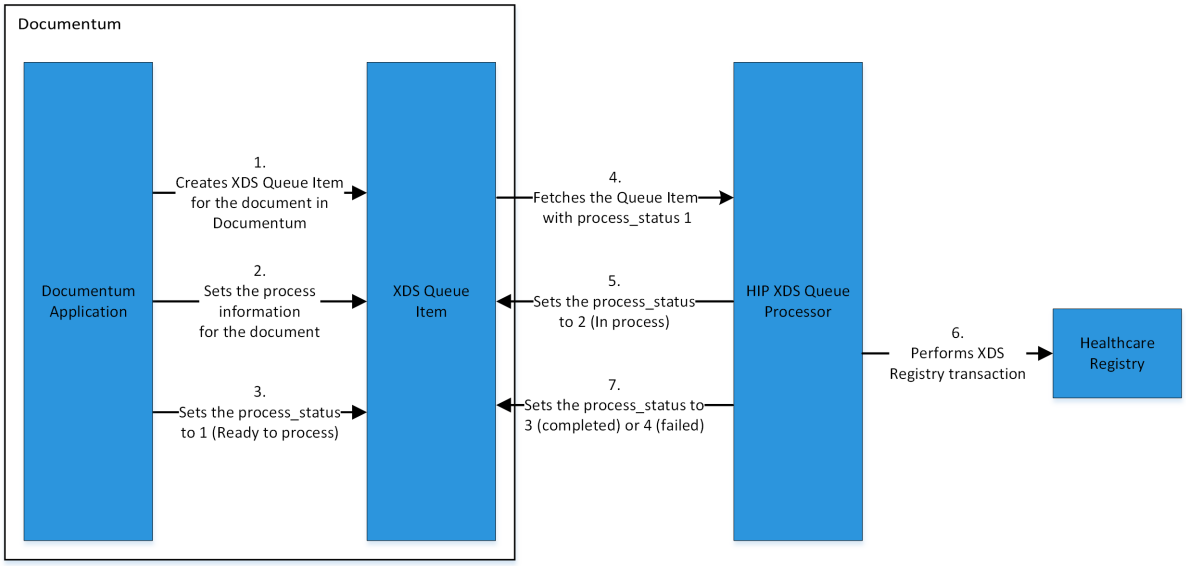
The Documentum XDS Queue currently supports the following XDS processes:

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

The `dmh_xds_queue_item` object type stores the registration, 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 deletes 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 by using the XML Registration data contained in the `dmh_xds_queue_item` content.

For the delete registration process, all document entries in XDS Registry that have 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 XDS Registry that has the document unique ID (`dmh_xds_queue_item.document_unique_id`) is marked for deletion.

- **Name**: dmh_xds_queue_item
- **Supertype**: dm_document
- **Object type tag**: 09
- **Indexes**: none

The `dmh_xds_queue_item` object type 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 the 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` is 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`

Name	Type	Description
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. You must configure the `patient_id` property 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 by 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 for DCTM Repository, page 45 provides more information on configuring the `Register Document Set` properties.

Configuring the XDS Queue Item Processor Properties, page 47 provides more information on configuring the registration polling properties.

XDS Registration XML, page 90 provides a sample content of the XDS Registration XML file.

XDS Registration Schema, page 93 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 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 for DCTM Repository, page 45 provides more information 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 XDS Registry.

Any Documentum application or process can create a '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, the 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 for DCTM Repository, page 45 provides more information on configuring the `Deprecate Document Set` properties.

EMR Integrations

Epic Integration

HIP supports integration of the Repository with Epic by 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 Reference Guide* provides details about the properties of `dmh_xds_queue_item`.

Configuring Epic Integration, page 50 provides details about the configuration that you must perform to enable integration of repository with C4E.

Features of XDS Repository Connector for xDB

XDS Repository Connector for xDB provides the following features:

- A blob store, either xDB or ECS, in which all documents are stored
- Optional replication of XML documents into xDB for analytics and search capability

Blob Store

The XDS Repository Connector for xDB serves as an XDS repository for storing and maintaining healthcare content, both XML and non-XML documents. By default, the XDS Repository Connector for xDB supports the following types of blob stores:

- xDB-based blob store
- Swift-based ECS blob store

Analytics and Search Capability

The XDS Repository Connector for xDB, besides from providing the capability to manage content in blobstore, provides the capability to store selected type of validated XML documents in a separate xDB library. You can utilize the xDB capabilities to search and perform analytics on the XML documents using xQuery or other XML tools.

Configuring the xDB XML Store, page 65 provides information about configuring the XML configuration file for schema validation.

Chapter 3

Configuring XDS Repository Connectors

This chapter describes the steps to configure the XDS Repository Connectors.

Common Configurations for all XDS Repository Connectors

The following configuration tasks are common for all XDS Repository connectors:

- Creating the HIP configuration directory
- Deploying the property files in the HIP configuration directory
- Securing the repository server property file
- Configuring the server properties file
- Configuring SSL

Creating the HIP Configuration Directory

The HIP servers maintain configuration information in a directory called `hip` 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 all connectors.

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. The Windows system 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 the HIP Configuration Directory

1. Go to the `install` folder obtained after running the build command.

   The section *Installing the Library Dependencies* in the *XDS Repository Connectors Installation Guide* provides instructions to run the build command.

   For example:

   For XDS Repository Connector for Documentum:

   `C:\hip-cs-repository-1.9\install`

   For XDS Repository Connector for xDB:

   `C:\xdb-repository-1.9\install`

2. Extract the WAR file.
3. Go to the `\config` folder obtained after extracting the WAR file.

   For example:

   `\repository\config\`

4. Copy the folder containing the properties files to `C:\Users\username\.hip\`.

   For example:

   For XDS Repository Connector for Documentum, copy the `cs-repository` folder to `C:\Users\username\.hip\`.

   For XDS Repository Connector for xDB, copy the `xdb-repository` folder to `C:\Users\username\.hip\`.

5. Ensure that the all property files are present in the respective configuration folders.

   For XDS Repository Connector for Documentum, the `cs-repository` folder contains the following files:

   - `cs-repository.properties`
   - `cs-repository-context-extension.xml`
   - `hip-dctm-mapping.properties`
   - `hip-ppic-mapping.properties`
   - `hl7.properties`
   - `mimetype.properties`

For XDS Repository Connector for xDB, the `xdb-repository` folder contains the following files:

- `xdb-repository.properties`
- `hip-ppic-mapping.properties`
- `hip.xmlstorage.xdb.config.xml`

6. For XDS Repository Connector for Documentum, copy the `dfc.properties` file from `<Documentum installation directory>\config` to `C:\Users\username\.hip\cs-repository`.

   Skip this step if you are installing XDS Repository Connector for xDB.

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

Securing the Repository Server Properties File

The properties files for Documentum Repository, and xDB Repository contain information to access the respective XDS Repositories.

Secure the property file as follows:

- Restrict access to the system where the XDS Repository Server reside.
- Restrict access to the system where the property file reside.
- Restrict access to the property file by providing the read/write access only to the HIP administrator. Visit www.wiki.apache.org/tomcat/FAQ/Password.
- Provide an encrypted user password for the Repository. You can create an encrypted password using the encryptPassword utility.

  For Documentum, you can create an encrypted password during Documentum installation. The *EMC Documentum Content Server Administration and Configuration Guide* provides more information on password encryption.

Configuring the Server Properties File

You must configure the following properties for all XDS Repositories in the respective properties file:

- XDS Repository Server
- HTTP
- ATNA
- SOAP
- Usage Reporting
- Register Document Set URL
- Request and Response Validator
- XUA
- PPIC

You must configure the above mentioned properties in the following files:

- The `cs-repository.properties` file for XDS Repository Connector for Documentum
- The `xdb-repository.properties` file for XDS Repository Connector for xDB

Configuring the XDS Repository Server Properties

Property	Description
repository.homeCommunityId	The home community ID for the XDS Repository server. Each XDS Repository Server can be a part of an XDS Affinity Domain Community. This property is mandatory and has no default value.
repository.uniqueId	The unique ID assigned to the XDS Repository. Each XDS Repository Server identifies itself with a unique ID. This property is mandatory and has no default value.

Configuring the HTTPS Properties

The HTTPS properties enable the XDS Repository Server to use secure HTTPS communication. These properties are used for keystore and truststore configuration.

Property	Description
https.keyStore	The location of the keystore on the system where you keep the private SSL certificates for the system.
https.keyStorePassword	The password to access the keystore. This property is optional and has no default value.
https.server.keyAlias	The alias used for the server certificate in the keystore. If not specified, the first key read in the keystore is used.
https.server.privateKeyPassword	The private key password for encrypting the data.
https.trustStore	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.
https.trustStorePassword	The password to access the truststore.
https.ciphersuites	The cipher suite used to encrypt the session.

Configuring the ATNA Properties

Property	Description
audit.host	The name of the system that hosts the ATNA audit repository. **Default value**: localhost.
audit.port	The port number of the ATNA audit repository. **Default value**: 514.
audit.transport	The transport type for the ATNA audit repository, for example BSD, TLS, or UDP. **Default value**: UDP.
audit.sourceId	The source ID of the event. This property has no default value.

Configuring the Custom SOAP Routes Properties

Property	Description
soap.routeBuilderScriptSource	The location of custom groovy file that you can specify overriding the default SoapRouteBuilder script. **Note:** The `SoapRouteBuilder.groovy` file must have the `xuaEnabled` property defined in the file. The server fails to start if this property is missing in the file.

Configuring the Usage Report Properties

Property	Description
usagereport.username	The username to login to the Usage Reporting web application. **Default value**: Administrator.
usagereport.password	The password to login to the Usage Reporting web application. **Default value**: password.

Configuring the Register Document Set URL Properties

Property	Description
registry.registerDocumentSetUrl	The URL that XDS Repository uses when it registers a document set with the registry. This property is mandatory and has no default value.
registry.registerDocumentSetAsyncUrl	The URL that XDS Repository uses with asynchronous routes when it registers a document set with the registry. If undefined or commented out, the XDS Repository Server uses the Register Document Set URL property instead.

Configuring the Request and Response Validator Properties

The Request and Response validator flags are enabled by default. You must enable these flags to avoid the problems that may occur if the request is not correct.

Property	Description
repository.iti41.requestValidator.enabled **repository.iti43.requestValidator.enabled**	A flag that specifies whether to disable the incoming message validation for the specified IHE transaction. This property is optional.
repository.iti41.responseValidator.enabled **repository.iti43.responseValidator.enabled**	A flag that specifies whether to disable the outgoing message validation for the specified IHE transaction. This property is optional.

Configuring the XUA Properties

Property	Description
repository.xua.enabled	A flag that specifies whether to enable XUA for the XDS Repository Server. **Default value**: false.

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

After enabling XUA, configure the XUA policy, XUA SAML attributes, and trusted assertion provider properties in the repository properties file as described in the following topics.

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 directories:

- For XDS Repository Connector for Documentum: `/webapps/cs-repository/config/cs-repository/`
- For XDS Repository Connector for xDB: `/webapps/xdb-repository/config/xdb-repository/`

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

`/webapps/<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 the SAML Security Token attributes.

Property	Description
xua.saml2.token.validator	The validator that validates the SAML token in the request.
xua.crypto.provider	The encryption/decryption and signature validation.
xua.service.endpoint	The endpoint regular expression. The XDS Repository Server compares this value against the audience restriction attribute provided in the token.
xua.supported.authentication.methods	A comma-separated list of authentication methods supported by the XDS Repository.
xua.purposeOfUse.codeSystem	The supported code system for the PurposeOfUseCode attribute provided in the token.
xua.purposeOfUse.code.values	A comma-separated list of purpose of use code values supported by XDS Repository. If not configured, PurposeOfUseCode value provided in the token is not validated.
xua.role.codeSystem	The code system supported for the Role attribute provided in the token.
xua.role.code.values	A comma-separated list of code values supported by XDS Repository.

Configuring the XUA Attribute Validation Property

Property	Description
xua.authz.consent.option	A flag that specifies whether to enable or disable the validation of the patient consent attribute value of SAML token. **Default value**: false.

Configuring the Trusted Assertion Provider Properties

The configuration of trusted assertion provider certificates is a required configuration if XUA is enabled. The Trusted Assertion Provider properties and have no default value.

Property	Description
xua.assertion.provider.trustStore	The truststore file (for example, truststore.jks).
xua.assertion.provider.trustStorePassword	The truststore password.

Configuring PPIC

You must configure the PPIC properties for XDS Repository Connector for Documentum and XDS Repository Connector for xDB to enable the PPIC feature.

Note: The `hip-ppic-mapping.properties` file contains the PPIC properties. The `hip-ppic-mapping.properties` file contains the HIP PPIC mapping properties. Both the files are located in the `<hip_config_dir>/<repository>` folder.

Configuring the PPIC Server Properties

Property	Description
ppic.enabled	A flag that 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. **Default value**: False. **Note:** You must configure the **ppic.pdpServiceUrl** property if you enable this property.
ppic.pdpServiceUrl	The URL for making PDP service call for PPIC.

Configuring the HIP PPIC Mapping Properties

Property	Description
request.subjectId	The logical identifier of the user performing the original service request.
request.subjectRole	The relevant user subject roles from a locally defined code-set.

Configuring SSL for XDS Repository Connectors

HIP supports SSL configuration for Tomcat and WebLogic.

Configuring SSL for Tomcat

Add the paths for the keystore and truststore 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. For each server, enable the **Configuration > General > SSL Listen Port Enabled** setting.
   Ensure that you use a correct and unused port.
3. For each server, change the Keystore configuration as follows:
   a. From the **Configuration > Keystores** panel, click **Change**.
   b. 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. In the **Configuration -> SSL** panel, configure the following fields:
      Private Key Alias=serverXX
      Private Key Passphrase=changeit
5. Click **Save**.

Configuring XDS Repository Connector for Documentum

The configuration of XDS Repository Connector for Documentum consists of the following tasks:

- Configuring DFC
- Configuring the Repository Server
- Configuring the Web Container heap memory
- Configuring the HIP Documentum mapping properties
- Configuring Epic Integration

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

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 the information to connect to other systems, Documentum repositories, and XDS Repository itself.

The cs-repository.properties, page 79 topic in Appendix A provides sample content of the `cs-repository.properties` file. You can refer this file for configuration examples.

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

Configuring the Content Server Properties

Property	Description
description	The description for Documentum Content Server. This description is used only for display purposes. This property is optional and has no default value.
cs.docbaseName	The name of Documentum Content Server that XDS Repository uses when storing and retrieving documents. This property is mandatory and has no default value.
cs.userName	The username of the Documentum account you have already created. The section *Deploying Object Types* in the *XDS Repository Connector for Documentum Installation Guide* provides the instructions to create the username. This is the username that XDS Repository uses to access Documentum Content Server.

Property	Description

cs.password	The password of the Documentum account you have already created.
	
	The section *Deploying Object Types* in the *XDS Repository Connector for Documentum Installation Guide* provides the instructions to create to the password.
	
	This is the user password that XDS Repository uses to access Documentum Content Server.
	
	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. When the system prompts to enter the password, type the password as follows:
	
	`Enter clear text password`>Password
	
	The encrypted password is generated as follows:
	
	`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	The path and filename for your keystore. If your password is encrypted, you must type the path and filename for your keystore.

Configuring the ACL Property

Property	Description
repository.acl.name	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.

Configuring the Registry Properties for DCTM Repository

Property	Description
registry.registerDocumentSetUrl	The URL that XDS Repository uses when it registers a document set with the registry. This property is mandatory and has no default value.
registry.registerDocumentSetAsyncUrl	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, the XDS Repository Server uses the Register Document Set URL property instead.
registry.deleteDocumentSetUrl	The URL that XDS Repository uses to delete document set from the Registry. The Delete Document Set properties are required only when automatic registration and delete document set transactions are enabled.
registry.deleteDocumentSetAsyncUrl	The URL that XDS Repository uses when performing an asynchronous deletion of document set from the Registry.
registry.storedQueryAsyncUrl	The URL that XDS Repository uses when performing asynchronous RegistryStoredQuery transactions with Registry. You must configure this property to delete or deprecate documents from Registry.

Property	Description
registry.storedQueryUrl	The URL that XDS Repository uses when performing RegistryStoredQuery transactions with the Registry. You must configure this property to delete or deprecate documents from Registry. 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 the `cs-repository.properties` file as follows: `registry.storedQueryUrl=https://<host:port>/registry/services/trustedhosts/xds-iti18`
registry.updateDocumentSetUrl	The URL that XDS Repository uses to update document set from Registry.
registry.updateDocumentSetAsyncUrl	The URL that XDS Repository uses when performing an asynchronous update of a document set from Registry.

The XDS Queue Item processor uses the MLLP host configuration when registering documents in Documentum.

Property	Description
registry.mllp.host	The MLLP listening host where the Registry server runs. This property has no default value.
registry.mllp.port	The MLLP listening port of Registry. **Default value**: 0.

Configuring the HIM Documentum Metadata Properties

Property	Description
store.document.metadata	A flag that specifies whether to enable the storage of complete document metadata in Documentum Content Server. **Default value**: false. • **`True`**: Automatic registration and PnR are done only for `dmh_document`. • **`False`**: Automatic registration and PnR are done only for `dm_document`. The **Unique Id** attribute must not be empty in `dm_document`. At a time, you can configure this property to enable automatic registration and PnR for only one type of document.
patient.folder.location	The cabinet where the patient folders are created. The default cabinet is `Patients`. The cabinet name should be preceded by a '/'.

Configuring the XDS Queue Item Processor Properties

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

Property	Description
repository.queueItemPollingInterval	The frequency in seconds, at which 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`**. **Default value**: 0.

Property	Description
repository.queueItemPollingBatchSize	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. **Default value**: 0.
repository.queueItemResubmitPollingInterval	This property is for documents that failed to complete the registration process. 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. **Default value**: 0.

Configuring the RabbitMQ Properties

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

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

Configuring the Unified Endpoint Properties

Property	Description
unifiedEndpoint .routeBuilderScriptSource	The UnifiedRouteBuilder script path. **Default value**: classpath:com/emc/healthcare/xds /repository/commons/UnifiedEndpointRouteBuilder .groovy.
unifiedEndpoint.enabled	A flag that specifies whether to enable or disable the unified endpoint on the XDS Repository Server. **Default value**: true. **Note:** If **unifiedEndpoint.enabled** is set to **`true`**, it is mandatory to enable HTTP. If HTTP is not enabled, the requests to XDS Repository will fail.
unifiedEndpoint.xdsServer.port	The http port of the application server where the XDS Repository Server is deployed. This property is mandatory and can take any value greater than zero. **Default value**: 0.
unifiedEndpoint.xdsServer.contextPath	The context name of the XDS Repository Server in the application server. **Default value**: /cs-repository.
unifiedEndpoint.xdsServer.servicePath	The service path of the XDS Repository Server in the application server. **Default value**: /services/.

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 the HIP Documentum Mapping Properties

The hip-dctm-mapping.properties file, located in the <hip_config_dir>/cs-repository folder, 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 **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.

hip-dctm-mapping.properties, page 88 shows a sample hip-dctm-mapping.properties file.

Configuring Epic Integration

Configuring Epic Integration is an optional step that applies only to the users who want to integrate their repository with Epic using Connector for Epic (C4E).

After installing the XDS Repository Server, perform all the steps listed in *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	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>`

-sendEmail	Specifies whether to send an email with merge status to the user. If set to true, the email is sent. This argument is optional. **Default value**: false. For example: `-sendEmail<space>false`
-userNameToSendEmail	The email address of the user to whom the merge status is sent. This argument is optional. The default value is the name of the repository owner. For example: `-userNameToSendEmail<space><user email address>`
-sendEmailOnMergeCompletion	Specifies whether an email must be sent to the user when the merge operation is completed. For example: `-sendEmailOnMergeCompletion<space>true`

-userAccountToSendMergeStatusEmail	The user account to which the email must be sent when the merge operation is completed.
	
	For example:
	
	`-userAccountToSendMergeStatusEmail<space>`
	`Administrator`
	
	Ensure that the Administrator email address is set to a valid email address and the Content Server is configured with a valid SMTP Server name.

Argument	Description
-hipRepositoryACL	The ACL used to create the HIP documents. For example: `-hipRepositoryACL<space>newACL`
-deleteRetiringFolderAfterMerge	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

You must perform the above configurations in the `hl7.properties` file located in the `<hip_config_dir>/cs-repository` folder.

Configuring the HL7 Time Zone Property

Property	Description
hl7.message.timestamp.includeTZOffset	Specifies whether to include time zone offset in Time Stamp fields. HL7 recommends using time zone offset in Time Stamp fields. Since the time zone information is optional, certain applications prefer not to include the time zone offset.

Configuring the HL7 Inbound Properties

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

Property	Description
hl7.inbound.mllp.port	The port to which the XDS Repository Server listens for inbound HL7 requests. The XDS Repository Server does not accept inbound HL7 messages unless you define this property. **Default value**: 0.

Property	Description
hl7.inbound.mllp.securePort	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.
hl7.inbound.onReceiveDuplicateMessage	Specifies whether Repository should accept or reject duplicate HL7 messages, if any, are sent by the external systems. Due to network disconnect or timeout issues, external systems may the send same HL7 message twice to Repository. If the value of this property is set to **reject**, Repository rejects the duplicate message and sends an application error response code (AE) to the sender. If the value of this property is set to **ignore**, Repository ignores the duplicate message and sends a positive acknowledgement to the sender. **Default value**: reject.
hl7.inbound.onExceptionBehaviour	The HL7 inbound message response that Repository should send to the external systems when message processing fails or an exception occurs during processing. If the value of this property is set to **NAK**, Repository sends an application error response code (AE) to the sender. If the value of this property is set to **ACK**, Repository sends a positive acknowledgement to the sender. **Default value**: NAK.
hl7.inbound.queueitem.folderPath	The folder path to save the inbound queue item.
hl7.inbound.queueitem.acl	The ACL for the inbound HL7 queue item.
hl7.inbound.message.encoding	The HL7 MLLP encoding format. **Default value**: UTF-8. HIP Repository supports only **ISO 8859-1** and **UTF-8** encoding.

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.

Property	Description
hl7.outbound.message.version	The supported versions of HL7 outbound messages. The XDS Repository Server can send output messages in different HL7 versions. This property is mandatory. **Default value**: 2.3.1.
hl7.outbound.mllp.host	The external HL7 server host where the XDS Repository Server sends the HL7 messages. This property is mandatory. **Default value**: localhost.
hl7.outbound.mllp.port	The port number of the HL7 server host where the XDS Repository Server sends the HL7 messages. **Default value**: zero. If set to `0`, the server will not send MDM messages.
hl7.outbound.requestTimeout	The request timeout value in milliseconds. This property is optional. **Default value**: 3000.
hl7.outbound.mllpConnection .keepAlive	Specifies whether the Repository connection with external HL7 systems should remain open for the entire lifetime of Repository or whether the connection should be closed after sending each outbound message. **Default value**: true. If set to **`false`**, the Repository connection with external HL7 systems will be closed after sending each outbound message. Keeping the connection open improves performance because it permits processing a large number of messages in a single connection. Keeping the connection closed affects the performance negatively because the connection has to be closed after sending each outbound message and reopened again.
hl7.outbound.queue.pollingInterval	The polling interval in seconds.
hl7.outbound.queue .reProcessInProgressItems OlderThanDays	The number of days the system waits before resetting "in progress" items to "to be processed". This property is optional. **Default value**: 1.

Property	Description
hl7.outbound.sendingApplication .namespaceId	The name space ID of the sending application.
hl7.outbound.sendingApplication .universalId	The universal ID of the sending application.
hl7.outbound.sendingApplication .universalIdType	The universal ID type of the sending application, such as ISO or HL7.
hl7.outbound.sendingFacility .namespaceId	The namespace ID of the sending facility.
hl7.outbound.sendingFacility .universalId	The universal ID of the sending facility.
hl7.outbound.sendingFacility .universalIdType	The universal ID type of the sending facility such as ISO, HL7 and so on.

Configuring the HL7 Render Empty Segment Properties

Property	Description
hl7.mdm.T01.renderEmptySegments **hl7.mdm.T02.renderEmptySegments** **hl7.mdm.T11.renderEmptySegments**	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 If **hl7.mdm.T01.renderEmptySegments** =`<segmentName>-<fieldSeq>` and the segment has no value, the outgoing MDM will have a segment with no value. If **hl7.mdm.T01.renderEmptySegments** =`<segmentName>-<fieldSeq>` and the segment has a value, the outgoing MDM will have a segment with the value of the segment. If **hl7.mdm.T01.renderEmptySegments**=`NONE` and the segment has no value, the outgoing MDM will not have any segment. If **hl7.mdm.T01.renderEmptySegments**=`NONE` and the segment has a value, 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.

Property	Description
hl7.outbound.queue.maxRedeliveries	The number of times HIP tries to redeliver the HL7 outbound message if the message delivery fails in the first attempt. If the value is set to 0, HIP will not redeliver the message. **Default value**: 3.
hl7.outbound.queue.reDeliveryDelay	The duration in seconds, for which HIP waits before each redelivery of message. **Default value**: 60 seconds.

Configuring the 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 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

Property	Description
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= 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

Property	Description

	• **title|dmh_record_type.subtype**: Configure as follows if you want MDM Message Builder to search for `title` first, and if found blank, search for `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 MDM Message Builder to search for `dmh_record_type.subtype` first, and if found blank, search for `title`.
	`hl7.mdm.T02.NTE-3=dmh_record_type .subtype|title`
	• **NONE**: Configure as follows if you do not want to include the 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`
	`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

Property	Description
hl7.queue.enabled	A flag that specifies whether to enable or disable the Message Queue support. **Default value**: False. For inbound messages, the Message Queue enables the HIP system to store the messages in a Message Queue and send an 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.
You need to configure the following properties only if **hl7.queue.enabled** is enabled.	
hl7.merge.exchange	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.
hl7.merge.queue	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.
hl7.outbound.exchange	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.
hl7.failed.queue	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.

Configuring the HL7 EOB Message for C4E

Property	Description
repository.eob.scanType	The type of the EOB document scanned to the Documentum repository.
repository.eob.hospitalLocation	The location of the hospital where the scanning is performed.

Property	Description
repository.eob.batchUser	The user ID for which the batch must be created.
repository.eob.batchMode	The batch type that must be created.
repository.eob.postingDepartment	The EOB posting department.

Configuring 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.

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`

Configuring XDS Repository Connector for xDB

The configuration of XDS Repository Connector for xDB consists of the following:

- Configuring the xDB repository property file
- Configuring the XML configuration file

Configuring xDB Repository Properties File

The `xdb-repository.properties` file contains user-definable properties for the XDS Repository Server. This file provides the XDS Repository Server with the information to connect to xDB.

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

Configuring the xDB Repository Server Properties

Property	Description
description	A description for the xDB Repository Server. This description is used only for display purposes.

Configuring the xDB Properties

Property	Description
xdb.libraryPath	The location in the xDB database where the XDS Repository Server stores the data. **Default value**: repository

Configuring the xDB Read and Write Node Properties

Property	Description
xdb.readNode.bootstrapFileName	The connection to a dedicated page server that runs behind the specified TCP/IP port. A bootstrap specifies a connection to a federation.
xdb.readNode.databaseName	The name of the xDB database.
xdb.readNode.userName	The username that the XDS Repository Server uses to access xDB.
xdb.readNode.password	The password of the user that the XDS Registry Server uses to access xDB.
xdb.readNode.keystore	The path to the Read node keystore file. This property is required if the HIP encrypted password is configured.
xdb.writeNode.bootstrapFileName	The bootstrap filename for the Write node.
xdb.writeNode.databaseName	The database name for the Write node.
xdb.writeNode.userName	The username for the Write node.
xdb.writeNode.password	The password for the Write node.

Property	Description
xdb.writeNode.keystore	The path to the Write node keystore file. This property is required if the HIP encrypted password is configured.
xdb.backup.readNode .bootstrapFileName	The bootstrap filename for the backup Read node.
xdb.backup.readNode.databaseName	The database name for the backup Read node.
xdb.backup.readNode.userName	The username for the backup Read node.
xdb.backup.readNode.password	The password for the backup Read node.
xdb.backup.readNode.keystore	The path to the backup Read node keystore file. This property is required if the HIP encrypted password is configured.
xdb.backup.writeNode .bootstrapFileName	The bootstrap filename for the backup Write node.
xdb.backup.writeNode.databaseName	The database name for the backup Write node.
xdb.backup.writeNode.userName	The username for the backup Write node.
xdb.backup.writeNode.password	The password for the backup Write node.
xdb.backup.writeNode.keystore	The path to the backup Write node keystore file. This property is required if the HIP encrypted password is configured.

Configuring the Blob Storage Property

Property	Description
blobstorage.type	The type of the blob library where XDS Repository Connector stores both XML and non-XML content. **Default value**: xdb **Possible values**: xdb (if the content is stored in the xdb blobstore), ecs (if the content is stored in the ecs blobstore) If you configure, the **blobstorage.type** as **`ecs`**, you must also configure the ECS related properties described in Configuring the ECS Blob Store Properties, page 64.

Configuring the xDB Blob Container Properties

Property	Description
blobstorage.xdb.partition.strategy	The storage partition strategy used for the blob library. **Possible values**: default, hash **Default value**: default • **`default`**: Indicates that all objects are kept in the library specified by **blobstorage.xdb.partition .default.library** • **`hash`**: Indicates that container segmentation is required. If a repository is deployed using **blobstorage.xdb .partition.strategy**=default, you cannot change the segmentation to hash at a later time. Similarly, If a repository is deployed using **blobstorage.xdb.partition.strategy**=hash, you cannot change the segmentation to default at a later time.
blobstorage.xdb.partition.hash.library	The library folder used for blob storage when the storage partition strategy used is **`hash`**.
blobstorage.xdb.partition.default .library	The library folder used for blob storage when the storage partition strategy used is **`default`**.
blobstorage.xdb.partition.hash.range	The hash partition range used by the blob storage container when the library resolver strategy used is **`hash`**. **Default value**: 10

Configuring the xDB XML Library Properties

Property	Description
xmlstorage.type	The type of the XML library where XDS Repository Connector stores only the XML content. **Default value**: xdb
xmlstorage.xdb.config.file	The configuration file used to validate the input xml. **Default value**: ${com.emc.healthcare.home}/xdb -repository/hip.xmlstorage.xdb.config.xml

Property	Description
xmlstorage.xdb.partition.strategy	The storage partition strategy used for the XML library. **Possible values**: default, hash **Default value**: default
xmlstorage.xdb.partition.hash.library	The library folder used for xml storage when the storage partition strategy used is **`hash`**.
xmlstorage.xdb.partition.default .library	The library folder used for xml storage when the storage partition strategy used is **`default`**.
xmlstorage.xdb.partition.hash.range	The hash partition range used by the xml storage container when the storage partition strategy used is **`hash`**. **Default value**: 10

Configuring the ECS Blob Store Properties

Property	Description
ecs.container.strategy	The ECS container segmentation strategy. **Possible values**: default, hash **Default value**: default • **`default`**: Indicates that all objects are kept in the container specified by **ecs.container.name**. • **`hash`**: Indicates that container segmentation is required. If a repository is deployed using **ecs.container.strategy** =default, you cannot change the segmentation to hash at a later time. Similarly, If a repository is deployed using **ecs.container.strategy**=hash, you cannot change the segmentation to default at a later time.
ecs.container.segment.range	The hash partition range used by the ECS container when the container segmentation strategy used is **`hash`**. **Default value**: 10
ecs.container.prefix	The prefix used by the ECS container when the container segmentation strategy used is **`hash`**. **Default value**: hip

Property	Description
ecs.container.suffix	The suffix used by the ECS container when the container segmentation strategy used is **`hash`**. **Default value**: hip
ecs.container.name	The name of the container in ECS where the content is stored **Default value**: hip
ecs.autocreate.container	The flag to decide whether the application should create the container. If set to **`false`**, the container should already exist in the ECS. **Default value**: true
ecs.swift.username	The ECS Swift username. This property has no default value.
ecs.swift.password	The ECS Swift password. This property has no default value.
ecs.swift.keystore	The path to the ECS password keystore file. This property has no default value. This property is required if **ecs.swift.password** is encrypted using HIP EncryptPassword.
ecs.swift.endpoint	The ECS Swift endpoint URL. This property has no default value.

Configuring the xDB XML Store

The configurations in the `hip.xmlstorage.xdb.config.xml` file defines the behavior of the XML storage.

Schema of the configuration file: `hip.xdbrepo.config.xsd`

You can configure the following in the xDB XML configuration file:

- Filters
- Validations
- Metadata storage
- Transformations

Configuring the Filters

The **filters** element consists of a list of filters that define the XML documents that should be stored in the XML library.

Mediatype Filter

The **mediatype** filter defines the mediatypes that must be stored in the XML storage. This is typically only **application/xml**.

Example:

```
<mediatype>
  <set>
    <entry>application/xml</entry>
    <entry>text/xml</entry>
  </set>
</mediatype>
```

Namespace Filter

The **root-element-namespace** filter consists of non-empty set entries that define the acceptable root element namespaces.

The documents where the root element does not have one of the specified namespaces are not stored in the XML storage.

Example:

```
<root-element-namespace>
  <set>
  <entry>urn:emc:healthcare:hip:xds:xdbrepository:config.xsd.1.9</entry>
  <entry>urn:hl7-org:v3</entry>
  </set>
</root-element-namespace>
```

Metadata Filter

The **metadata** filter consists of a field definition and a set of values for that field.

Example:

```
<metadata field="mimeType">
  <set>
  <entry>application/xml</entry>
  </set>
</metadata>
```

Custom Filter

The **custom** filter consists of a class definition and a set of configurations specific to the custom filter.

Example:

```
<custom class="a.fully.qualified.classname">
   ..any configuration specific to the custom filter
</custom>
```

The class referenced by the class attribute must exist in the class path and must implement the **com.emc.healthcare.repository.extensions.api.CustomFilter** interface located in the `hip-repository-extensions-api-1.9.0.jar` file.

Combining the Filters

You can combine two or more filters by using the following elements that represents logical AND, logical OR or logical NOT:

- <and>: Logical AND
- <or>: Logical OR
- <not>: Logical NOT

For example, the following filter stores only the documents that have **application/xml** mediatype (mimetype) and where the namespace of the root element is `urn:emc:healthcare:hip:xds:xdbrepository:config.xsd.1.9`.

```
<and>
  <mediatype>
    <set>
      <entry>application/xml</entry>
      </set>
  </mediatype>
  <root-element-namespace>
    <set>
      <entry>urn:emc:healthcare:hip:xds:xdbrepository:config.xsd.1.9</entry>
    </set>
  </root-element-namespace>
</and>
```

Configuring the Validations

The **validations** element consists of a list of validations that must be applied to the XML document prior to storing it in the XML library. A document is stored in the XML library only if it passes all the configured validations.

Schema Validation

The **schema** element defines a schema validation, wherein the XML Document is checked against an XSD.

The schema validation uses a schema locator to determine the schema against which the document must be validated.

Example:

```
<validations>
  <schema ignoreMissingSchema="false" ignoreValidationFailures="false">
    <schema-locator>
```

```
      <by-root-element-namespace/>
      <by-metadata/>
    </schema-locator>
  </schema>
</validations>
```

Schema Locator by Namespace

The **by-root-element-namespace** element defines the configuration to locate the schema by namespace of the root element of the XML document.

You can define multiple <map> elements to map different namespaces to different schema files.

The following example uses the schema specified in `C:/.hip/xdb-repository/hip.xdbrepo.config.xsd` to validate a document if the root element namespace is `urn:emc:healthcare:hip:xds:xdbrepository:config.xsd.1.9`.

Example:

```
<by-root-element-namespace>
  <map namespace="urn:emc:healthcare:hip:xds:xdbrepository:config.xsd.1.9"
       schema="C:/.hip/xdb-repository/hip.xdbrepo.config.xsd" />
</by-root-element-namespace>
```

Schema Locator by Metadata

The **by-metadata** element defines the configuration to locate the schema by the value of a metadata field.

You can define multiple <map> elements to map different values to different schema files.

The following example uses the schema specified in `C:/.hip/xdb-repository/hip.xdbrepo.config.xsd` to validate a document if the value of the metadata field **mimeType** is `text/xml`.

Example:

```
<by-metadata field="mimeType">
   <map value="text/xml" schema="C:/.hip/xdb-repository/hip.xdbrepo.config.xsd"/>
</by-metadata>
```

Schema Locator by Custom Class

The **custom** element defines the configuration to locate the schema by custom class.

Example:

```
<custom class="a.fully.qualified.classname">
   ...any configuration specific to the custom schema locator
</custom>
```

The class referenced by the class attribute must exist in the class path and must implement the **com.emc.healthcare.repository.extensions.api.CustomSchemaLocator** interface located in the `hip-repository-extensions-api-1.9.0.jar` file.

Custom Validation

The **custom** element defines a custom validation, wherein the XML Document is validated against the configurations specific to the custom validator.

The class referenced by the class attribute must exist in the class path and must implement the **com.emc.healthcare.repository.extensions.api.CustomValidator** interface located in the `hip-repository-extensions-api-1.9.0.jar` file

Example:

```
<custom class="a.fully.qualified.classname">
   ...any configuraiton specific to the custom validator
</custom>
```

Configuring the Metadata Storage

The **metadata** element consists of a list of input metatdata that must be stored along with the content in the xDB XML store. The **to** attribute defines the name of the input content metatdata that must be read and stored in the xDB with the name defined in the **from** attriubute.

Example:

```
<metadata>
  <map from="uniqueId" to="uniqueId"/>
  <map from="mimetype" to="mimetype"/>
</metadata>
```

Configuring the Transformations

The **transformations** element consists of elements that define the transformations that are performed on an XML document before it is stored in xDB.

Example:

```
<transformations>
    ....
</transformations>
```

Identity Transformation

The **identity** element defines a transformation that returns the value passed to it.

```
<identity/>
```

XSLT Transformation

The **xslt** element consists of elements that define the XSLT transformation that is performed on an XML document before it is stored in xDB.

```
<xslt>
   <stylesheet>some.file.xsl</stylesheet>
```

```
   <output-properties>…</output-properties>
   <parameters>...</parameters>
</xslt>
```

Custom Transformation

The **custom** element defines a custom transformation that is performed on an XML document before it is stored in xDB.

The class referenced by the class attribute must exist in the class path and must implement the **com.emc.healthcare.repository.extensions.api.CustomTransformation** interface located in the `hip-repository-extensions-api-1.9.0.jar` file

Example:

```
<custom class="a.fully.qualified.classname">
   ...any configuraiton specific to the custom transformation
</custom>
```

Switch Transformation

The **switch** element define the configuration to dynamically choose a transform based on a condition.

```
<switch>
   <value>
     <rootnamespace/>
      ...or...
     <metadata field="nameoffield"/>
   </value>
   <case value="">...a transformation goes here...</case>
   <default>  if no case applies, the default is used....
              ...If absent, the identity transform is used.</default>
</switch>
```

Chapter 4

Customizing XDS Repository Connector for Documentum

This chapter describes the steps to customize the XDS Repository Connector for Documentum.

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 is 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 Properties, page 37 provides more 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 Document Repository, and to request Document Repository to store these documents and then register them with Document Registry.

A PnR transaction carries the following:

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

Whenever an `XDSDocumentEntry` object is added to XDS Repository through the PnR transaction, a document is created in XDS Repository. The Document Creation Extensions feature allows you to alter the way the documents are created in 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 are 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;
    }
}
```

Implementing PnR Custom Extension Support

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

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 must also instruct 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 customers. The latest 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 an entry key configured as **`defaultRoute`** and value configured to any custom route that can process unknown messages.

Implementing Custom HL7 Message Processing

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

By default, HIP Repository rejects messages 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 Process HL7 MDM T02 Inbound Messages

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 follows:

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>
```

Chapter 5

Configuration and Customization Examples

EMC XDS Repository Connector for Documentum SDK provides examples for Documentum configurations and code samples for HIP customization.

The SDK document provides user stories for different scenarios along with sample configurations and code samples for those scenarios. The sample configurations and code samples are intended to enable EMC partners and integrators to fully utilize the new capabilities of HIP with minimum dependency on the Engineering team.

Appendix A

Sample Configuration Files

This appendix provides sample content of the XDS Repository Connector property files, XDS registration XML file, and XDS registration schema.

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 username 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/services/xds-iti42
# REQUIRED, NO DEFAULT
# registry.registerDocumentSetUrl=

#The URL for performing asynchronous RegisterDocumentSet transactions
# with the registry.
# Example:
# registry.registerDocumentSetAsyncUrl=http://localhost/registry/services/xds-iti42as
# DEFAULT:  The synchronous URL will be used in Async routes
# if the asynchronous routes are not defined
# registry.registerDocumentSetAsyncUrl=

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

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


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

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

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

#The URL for performing asynchronous update transactions with the registry.
# Example:
# registry.updateDocumentSetAsyncUrl=http://localhost/registry/services/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 likelihood 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=

#------------------ Unified EndPoint Related Properties----------------#
 unifiedEndpoint.routeBuilderScriptSource=classpath:com/emc/healthcare/xds/repository/commons/UnifiedEndpointRouteBuilder.groovy
```

```
unifiedEndpoint.enabled = true
unifiedEndpoint.xdsServer.port=0
unifiedEndpoint.xdsServer.contextPath=/cs-repository
unifiedEndpoint.xdsServer.servicePath=/services/
```

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 onReceiveDuplicateMessage
# is set to reject then Repository will reject duplicate message and send
# AE( Application Error ) response code to the Sender. If onReceiveDuplicateMessage
# is set to ignore then Repository will ignore the message and will send
# a positive acknowledgement the sender.
# default = reject
#-------------------------------------------------------------------------
hl7.inbound.onReceiveDuplicateMessage=reject


#-------------------------------------------------------------------------
# HL7 Inbound Message response when message processing has failed or an
# exception has occurred while message processing.If onExceptionBehaviour
# is set NAK, then repository will send AE( Application Error )
# response code to the Sender. If onExceptionBehaviour is set to ACK,
# then repository will send a positive acknowledgement to the sender.
# default
#-------------------------------------------------------------------------
hl7.inbound.onExceptionBehaviour=NAK
#-------------------------------------------------------------------------
# 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

#-------------------------------------------------------------------------------
# Repository opens connection with external HL7 Systems for outbound hl7
# processing, connection remains open for entire life time of repository or
# connection is forcibly closed by external HL7 System.
# To close connection , set the property to false
# DEFAULT = true
#-------------------------------------------------------------------------------
hl7.outbound.mllpConnection.keepAlive=true
#-------------------------------------------------------------------------------
# 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=
```

```
#-----------------------------------------------------------------------------
# HL7 MLLP ENCODING
# DEFAULT = UTF-8
#-----------------------------------------------------------------------------

#hl7.inbound.message.encoding=UTF-8
#-----------------------------------------------------------------------------
#
# 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>
```

xdb-repository.properties

```
# User settable properties are listed in this file.
# The order of property evaluation is as follows:
# 1: ${com.emc.healthcare.home}/xdb-repository/xdb-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=XDB Repository Server
#--------------------------------
# The following set of 5 parameters are used to define the primary READ XDB node.
# These settings are mandatory.
#--------------------------------
xdb.readNode.bootstrapFileName=xhive://localhost:1235
xdb.readNode.databaseName=
xdb.readNode.userName=
xdb.readNode.password=
# xdb.readNode.keystore property is required if HIP encrypted password is
# configured

# DEFAULT=${com.emc.healthcare.home}/xdb-repository/hip.repository
# xdb.readNode.keystore=${com.emc.healthcare.home}/xdb-repository/hip.keystore
#
# The same applies to xdb.writeNode.keystore, xdb.backup.readNode.keystore
#  and xdb.backup.writeNode.keystore.
#--------------------------------
```

```
# The following (optional) set of 5 parameters is used to define
# the primary WRITE XDB node.
# For Single node deployment, set these 4 values to blank
#--------------------------------
xdb.writeNode.bootstrapFileName=
xdb.writeNode.databaseName=
xdb.writeNode.userName=
xdb.writeNode.password=
#xdb.writeNode.keystore=   is required if HIP encrypted password is configured
#--------------------------------
# The following (optional) parameters are comma separated list for
# Backup READ XDB nodes setup.
#--------------------------------
xdb.backup.readNode.bootstrapFileName=
xdb.backup.readNode.databaseName=
xdb.backup.readNode.userName=
xdb.backup.readNode.password=
#xdb.backup.readNode.keystore=  is required if HIP encrypted password is configured

#--------------------------------
# The following (optional) parameters are comma separated list for
# Backup WRITE XDB nodes setup.
#--------------------------------
xdb.backup.writeNode.bootstrapFileName=
xdb.backup.writeNode.databaseName=
xdb.backup.writeNode.userName=
xdb.backup.writeNode.password=
#xdb.backup.writeNode.keystore=   is required if HIP encrypted
# password is configured

#-------------------------------------------------------
#        BLOB STORE CONFIGURATION
#-------------------------------------------------------

# BLOB Storage Type
# DEFAULT value is xdb
# blobstorage.type=xdb

# BLOB Storage partition strategy
# Supported partition strategy is default, hash
# Default value is default
# set the property value to default to put all the objects in the
# library specified by blobstorage.xdb.partition.default.library
# set the property value to hash if container segmentation is required

# blobstorage.xdb.partition.strategy=default

# Blob storage hash library folder
# blobstorage.xdb.partition.hash.library=/BLOBS

# Blob storage default library folder
# blobstorage.xdb.partition.default.library=/BLOBS

# Blob Container hash partition range, used when hash partition strategy
# is configured
# DEFAULT value is 10
# blobstorage.xdb.partition.hash.range=10
#-------------------------------------------------------
#               XML STORE CONFIGURATION
#-------------------------------------------------------

# XML storage type
# DEFAULT value is xdb
# xmlstorage.type=xdb
```

```
# Configuration file to validate input xml
# DEFAULT value is ${com.emc.healthcare.home}/xdb-repository/hip.xmlstorage.xdb.config.xml
# xmlstorage.xdb.config.file=${com.emc.healthcare.home}/xdb-repository/hip.xmlstorage.xdb.config.xml

# XML Storage partition strategy
# Supported partition strategy is default, hash
# Default value is default
# set the property value to default to put all the objects in the library
# specified by xmlstorage.xdb.partition.default.library
# set the property value to hash if container segmentation is required

# xmlstorage.xdb.partition.strategy=default


# XML storage hash library folder
# xmlstorage.xdb.partition.hash.library=/XML

# XML storage default library folder
# xmlstorage.xdb.partition.default.library=/XML

# XML Storage Container hash partition range, used when hash partition
# strategy is configured
# DEFAULT value is 10
# xmlstorage.xdb.partition.hash.range=10

#-------------------------------------------------
# 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=

# 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=

# ---------------------------- HTTPS Related Properties -------------------- #
#
#   The following properites must be configured for secure communication
#   These are used for keystore and truststore configuration.
#
#   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}

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

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

# ------------------------- 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=
# ----------------------- ECS Blobstore Properties -------------------------#
# ECS container segmentation strategy
# Supported segmentation strategy are default, hash
# Default value is default
# set the property value to default to put all the objects in the container
# specified by ecs.container.name property
# set the property value to hash if container segmentation is required
# ecs.container.strategy=default

# ECS Container segmentation range, used when ecs.container.strategy property is
# set to hash
# DEFAULT value is 10
# ecs.container.segment.range=10
```

```
# ECS Container prefix, used when ecs.container.strategy property is set to hash
# DEFAULT value is hip
# ecs.container.prefix=hip

# ECS Container suffix, used when ecs.container.strategy property is set to hash
# DEFAULT value is hip
# ecs.container.suffix=hip

# Name of container in ecs  under which the content will be stored.
# default container name is hip
# ecs.container.name=hip
# flag to decide whether the application should create the contianer.
# If set to false the container should already exist in the ECS.
# Default value is true.
# ecs.autocreate.container=true
# ECS Swift username
# No default.
# ecs.swift.username=
# ECS Swift password
# No default.
# ecs.swift.password=
#Keystore filePath is required if "ecs.swift.password" is encrypted
# using HIP EncryptPassword
# No default.
#ecs.swift.keystore=${com.emc.healthcare.home}/xdb-repository/hip.keystore
# ECS Swift endpoint URL.
# No default.
# ecs.swift.endpoint=
```

Index

O

P

R

S

T

U

W

X