IHE Use Cases

Version 2020.3
2021-02-04

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

IHE Use Cases
InterSystems Version 2020.3 2021-02-04

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

List of Tables

About This Book

This book employs use cases to describe how to perform IHE messaging in the following products.

- InterSystems IRIS for Health™
- HealthShare Health Connect

Chapters include:

- "Introduction" summarizes the IHE use cases that InterSystems products can solve.
- "Supported IHE Transactions" lists the IHE transactions that InterSystems supports.
- "Performing PIX and PDQ Queries Against an EMPI"
- "Adding or Updating Demographic Data Held in an EMPI (PIX Add)"
- "Querying an XDS Registry and Retrieving a Document from a Repository"
- "Providing and Registering Documents to an XDS Document Repository"
- "Querying and Retrieving a Document from another Affinity Domain (XCA)"
- "Generating a C-CDA Document from an HL7 Message"
- "Generating a C-CDA Document by Querying an Internal Database"
- "Receiving a C-CDA Document and Converting it to an Internal Format"
- "Using Cross-Enterprise User Assertion (XUA)"
- "Internet User Authorization (IUA) Support"
- "Sharing Imaging (XCA-I and XDS.b-I)"
- "Registry Settings for IHE Communication" describes how to configure the InterSystems registries to enable communication with other systems.
- "Using the IHE Test Utility" describes how to learn to perform IHE messaging with the Test Utility.
- "Auditing" describes how to send auditing info to an external repository.

This book also contains a complete table of contents.

For more information on InterSystems, see the following books:

- *CDA Interoperability with SDA* describes how to work with CDA documents, transforms, and annotations.
- *SDA: InterSystems Clinical Data Format* introduces the SDA clinical data format and how to customize it.
- *Registry Guide for InterSystems IRIS for Health and Health Connect* describes how to set up and manage the necessary service registries.
- *FHIR Support in InterSystems Products* describes FHIR support in InterSystems products.
- *InterSystems IRIS for Health Installation Guide* describes how to install InterSystems IRIS for Health and discusses considerations for mirroring and Docker containers.
- *Health Connect Installation and Migration Guide* describes how to install Health Connect or migrate to Health Connect 2018.1 and later.
- *Health Connect Release Notes* lists new features in Health Connect.

- InterSystems IRIS Supported Platforms lists the platforms on which InterSystems healthcare products are supported.

1

Introduction

InterSystems provides business services, business processes, and business operations that you can use to implement the IHE profiles, actors and transactions supported by InterSystems. This book does not document all business hosts that are available, but rather provides an introduction to a few important use cases. For the complete list of IHE profiles, actors, options, and transactions supported by InterSystems products, see the InterSystems IHE Integration Statements.

Important: Prior to using a FHIR®-based profile, you need to enable the `HS_Services` user. To enable the user, open the Management Portal and navigate to **System Administration** > **Security** > **Users**. Select **HS_Services**, and then select **User Enabled**.

For each use case in this book, messages are traced through the product. A table lists the components and settings employed in each use case, and examples provide instructions on creating messages for input.

Note: Throughout this document, the following terms are used interchangeably:

- XDS.b/XDS
- PIXv3/PIX
- PDQv3/PDQ

The following use cases are detailed:

- PIX/PDQ
  - Perform a deterministic query against an EMPI
  - Perform a probabilistic query against an EMPI
  - Add or update demographic data held in an EMPI
- XDS
  - Query the Affinity Domain's registry and retrieve a document from a repository
  - Provide and register a document (CDA or other) to a repository in the Affinity Domain
- Service Registry
  - Query the registry in another Affinity Domain and retrieve a document from a repository
- C-CDA
  - Generate a C-CDA document from an HL7 message and provide and register it to a repository.

  - Generate a C-CDA document by querying an internal database
  - Receive a C-CDA document (via a registry or point-to-point) and convert it to an internal format

- XUA inbound and outbound requests

2

Supported IHE Transactions

For the complete list of IHE profiles, actors, options, and transactions supported by InterSystems products, see the InterSystems IHE Integration Statements.

3

Performing PIX and PDQ Queries Against an EMPI

InterSystems can perform both probabilistic and deterministic queries against an external EMPI (Enterprise Master Patient Index).

3.1 Perform a Deterministic Query Against an EMPI (PIXv3)

InterSystems products support deterministic queries against an EMPI via the IHE "PIXv3" profile. A PIX query provides an MRN (Medical Record Number) and assigning authority and receives back the name and MPI ID of a single patient.

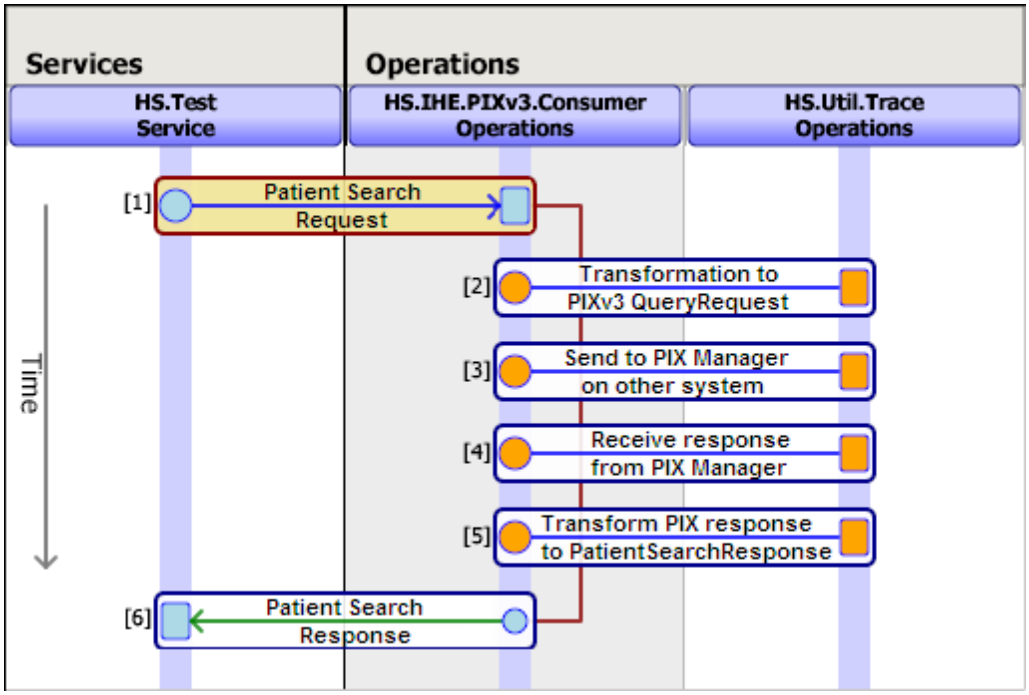
This section includes:

- PIX query message trace
- PIX query procedure
- PIX query components
- PIX query example

3.1.1 PIX Query Message Trace

The following diagram is an annotated PIX query message trace.

The test service shown in the diagram is a simple message router. Trace operations is a utility that makes intermediate processing steps visible. The numbers in the diagram match the steps in the procedure below.



3.1.2 PIX Query Procedure

1. You provide a **Patient Search Request** message containing an MRN and assigning authority to the InterSystems PIX Consumer operation.
2. The InterSystems PIX Consumer operation transforms the message into an IHE "PIXv3_QueryRequest" using the transformation specified in the TransformPatientSearchToPIX setting.
3. The InterSystems PIX Consumer operation then forwards the PIX request to the service named in the ServiceName setting. This setting is typically `PIXv3.Manager`, and it refers to a Service Registry entry pointing to the location of the PIX manager actor endpoint in another system.
4. The PIX manager on the other system returns a PIX response message that contains the name and MPI ID for a single patient if there is a match.
5. The InterSystems PIX Consumer operation transforms the response into a **Patient Search Response** message using the transformation specified in TransformPIXToPatientSearch.
6. The InterSystems PIX Consumer returns the **Patient Search Response** message, which contains the name and MPI ID of the patient, if found. If no patient is found, the response message indicates a `<ResultsCount>` of zero. If there is an error, the PIX Consumer returns null.

3.1.3 PIX Query Components and Settings

Table 3–1: Components and Settings Used in a PIX Query

Component	Setting
Business Hosts	PIX Consumer: **HS.IHE.PIXv3.Consumer.Operations**
Production Settings	TransformPatientSearchToPIX in the PIX Consumer operation
Production Settings	ServiceName in the PIX Consumer operation
Production Settings	TransformPIXToPatientSearch in the PIX Consumer operation
Production Messages	**HS.Message.PatientSearchRequest**
Production Messages	**HS.Message.PatientSearchResponse**
XSL Transformations	IHE/PIX/Version1/PatientSearchToPRPAIN201309UV.xsl
XSL Transformations	IHE/PIX/Version1/PRPAIN201310UVToPatientSearchResponse.xsl
Service Registry Entries	PIXv3.Manager
External IHE Actor Endpoint	PIX manager

3.1.4 PIX Query Example

The method below generates a PIX query:

```
ClassMethod PIXQuery()
 {
    /// Create Patient Search Request message
    Set obj=##class(HS.Message.PatientSearchRequest).%New()

    //Provide the Patient MRN
   Set obj.AssigningAuthority="EXTERNAL" //refers to an Assigning Authority entry in the OID Registry

    Set obj.MRN="1111222"

    // Send to the routing service (or directly to HS.IHE.PIXv3.Consumer.Operations)
    Do ##class(HS.Test.Service).SendSync(obj,.pr)

    quit
 }
```

3.2 Perform a Probabilistic Query Against an EMPI (PDQv3)

InterSystems products support probabilistic queries against an EMPI via the IHE “PDQv3” profile. A PDQ query provides a partial set of patient demographics and receives back full demographics for one or more MPI IDs (patients) that match the provided demographics.

This section includes:

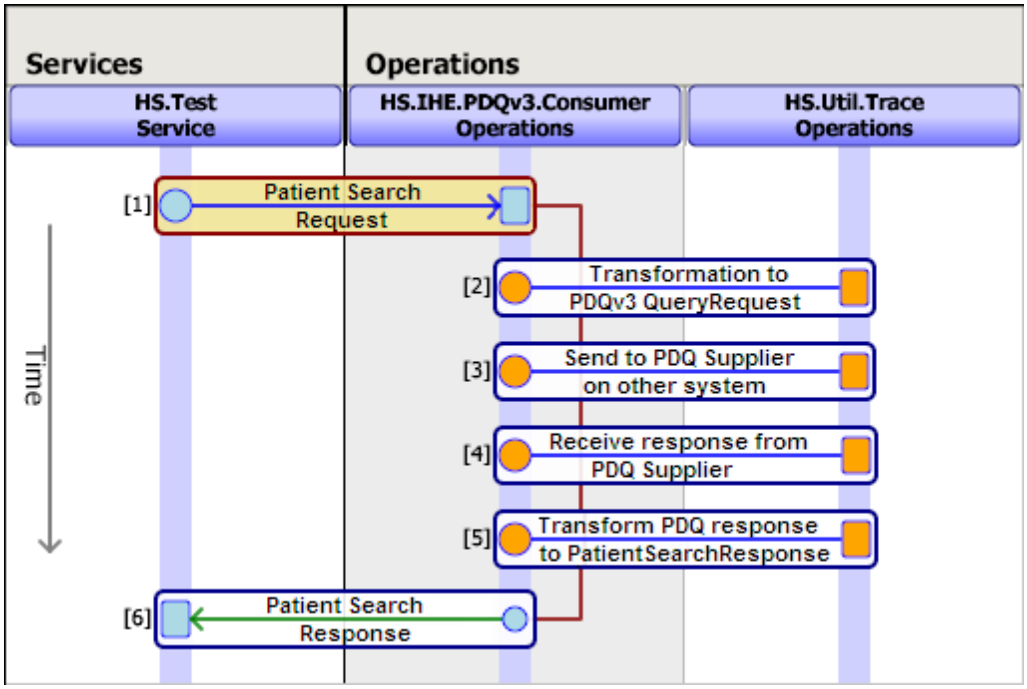
- PDQ query message trace
- PDQ query procedure
- PDQ query components

- PDQ query example

3.2.1 PDQ Query Message Trace

The following diagram is an annotated PDQ query message trace.

The test service shown in the diagram is a simple message router. Trace operations is a utility that makes intermediate processing steps visible in the trace. The numbers in the diagram match the steps in the procedure below.



3.2.2 PDQ Query Procedure

1. You provide a **Patient Search Request** message containing partial demographics to the InterSystems PDQ Consumer.
2. The InterSystems PDQ Consumer transforms the message into an IHE "PDQv3_QueryRequest" using the transformation specified in the TransformPatientSearchToPDQ setting.
3. The InterSystems PDQ Consumer then forwards the PDQ request to the PDQ supplier endpoint on another system that is named in the ServiceName setting.
4. The PDQ supplier on the other system returns a PDQ response message that contains the complete demographics for all patients that match the supplied partial demographics.
5. The InterSystems PDQ Consumer transforms the response into a **Patient Search Response** message using the transformation specified in the TransformPDQToPatientSearch setting.
6. The InterSystems PDQ Consumer returns the **Patient Search Response** message, which contains the full demographics and MPI IDs of the matching patients. If no patient is found, the response message indicates a `<ResultsCount>` of zero. If there is an error, the PDQ Consumer returns null.

3.2.3 PDQ Query Components and Settings

Table 3–2: Components and Settings Used in a PDQ Query

Component	Setting
Business Hosts	PDQ Consumer: **HS.IHE.PDQv3.Consumer.Operations**
Production Settings	TransformPatientSearchToPDQ in the PDQ Consumer
Production Settings	ServiceName in the PDQ Consumer
Production Settings	TransformPDQToPatientSearch in the PDQ Consumer
Production Messages	**HS.Message.PatientSearchRequest**
Production Messages	**HS.Message.PatientSearchResponse**
XSL Transformations	IHE/PDQ/Version1/PatientSearchToPRPAIN201305UV.xsl
XSL Transformations	IHE/PDQ/Version1/PRPAIN201306UVToPatientSearchResponse.xsl
Service Registry Entries	PDQv3.Supplier
External IHE Actor Endpoint	PDQ Supplier

3.2.4 PDQ Query Example

The method below generates a PDQ query:

```
ClassMethod PDQQuery()
 {
    // Create Patient Search Request message
    Set obj=##class(HS.Message.PatientSearchRequest).%New()

    // Provide patient demographics
    Set obj.FirstName="James"
    Set obj.LastName="Smith"

    // Required only for HS.Test.Service to distinguish between PIX/PDQ
    Do obj.AdditionalInfo.SetAt(1,"PDQ")

    // Send to the routing service (or directly to HS.IHE.PDQv3.Consumer.Operations)
    Do ##class(HS.Test.Service).SendSync(obj,.sr)

    quit
 }
```

4

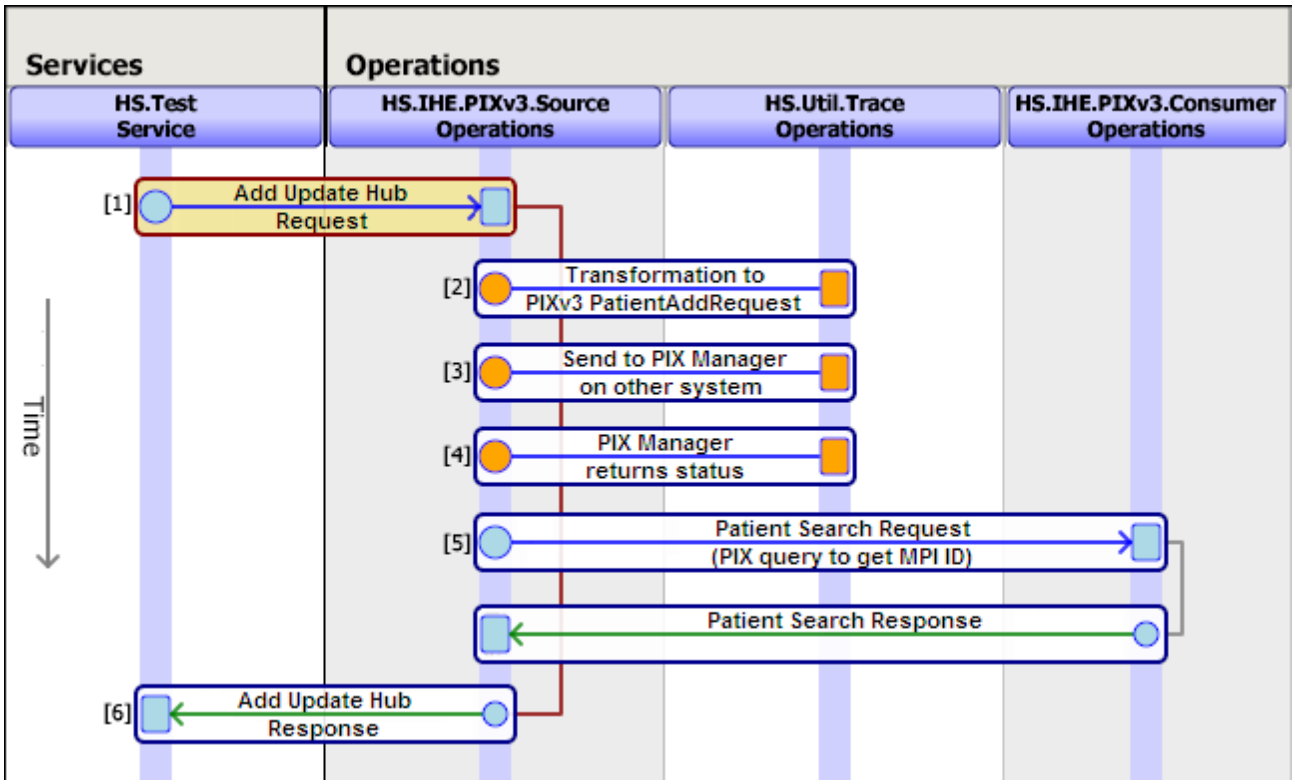
Adding or Updating Demographic Data Held in an EMPI (PIX Add)

InterSystems products can add a patient or update the demographic data held for a patient in an external EMPI via an IHE "PIXv3_PatientAddRequest" transaction.

4.1 PIX Add Message Trace

The following diagram is an annotated PIX add message trace that returns the MPIID.

The test service shown in the diagram is a simple message router. Trace operations is a utility that makes intermediate processing steps visible in the trace. The numbers in the diagram match the steps in the procedure below.



4.2 PIX Add Procedure

1. You provide an **Add Update Hub Request** message that contains the necessary demographic data to the InterSystems PIX Source.
2. The InterSystems PIX Source transforms the message into an IHE "PIXv3_PatientAddRequest" using the transformation specified in the TransformAddUpdateHubToPIX setting.
3. The InterSystems PIX Source then forwards the PIX request to the PIX manager endpoint on another system that is named in the ServiceName setting.
4. The PIX manager on the other system returns an acknowledgement or error.
5. If the OperationToLocateMPIID setting in the InterSystems PIX Source contains a value (typically **HS.IHE.PIXv3.Consumer.Operations**), then it sends a patient search request to the named operation, and that operation performs a PIX query to get the MPIID of the patient.
6. The InterSystems PIX Source returns an **Add Update Hub Response** message. Depending on the setting in step 5, this response message may contain the MPIID. If there is an error, the InterSystems PIX Source returns null.

4.3 PIX Add Components and Settings

Table 4–1: Components and Settings Used in a PIX Add

Component	Setting
Business Hosts	PIX Source: **HS.IHE.PIXv3.Source.Operations**
Business Hosts	PIX Consumer: **HS.IHE.PIXv3.Consumer.Operations** • if returning the MPI ID
Production Settings	TransformAddUpdateHubToPIX in PIX Source
Production Settings	ServiceName in PIX Source
Production Settings	OperationToLocateMPIID in PIX Source
Production Messages	**HS.Message.AddUpdateHubRequest**
Production Messages	**HS.Message.AddUpdateHubResponse**
Production Messages	**HS.Message.PatientSearchRequest** (if PIX)
Production Messages	**HS.Message.PatientSearchResponse** (if PIX)
XSL Transformations	IHE/PIX/Version1/AddUpdateHubRequestToPRPAIN201301UV.xsl
Service Registry Entries	PIXv3.Manager
External IHE Actor Endpoint	PIX manager

4.4 PIX Add Example

The method below generates a PIX add:

```
ClassMethod PIXADD()
 {
   // Create AddUpdateHub Message
   Set obj=##class(HS.Message.AddUpdateHubRequest).%New()

   // Name, sex, DOB
   Set obj.FirstName="James"
   Set obj.LastName="Smith"
   Set obj.Sex="M"
   Set obj.DOB=obj.DOBDisplayToLogical("2000-09-30")

   // Inserts full birth name information for the patient
   Set tName = ##class(HS.Types.PersonName).%New()
   Set tName.Prefix = "Mr."
   Set tName.Given = "James"
   Set tName.Middle = "Henry"
   Set tName.Family = "Smith"
   Set tName.Suffix = "IV"
   Set tName.Type="Birth"
   Do obj.Names.Insert(tName)

   // Inserts name of patient's spouse
   Set tName = ##class(HS.Types.PersonName).%New()
   Set tName.Prefix = "Mx."
   Set tName.Given = "Pat"
   Set tName.Middle = "A."
   Set tName.Family = "Henderson"
   Set tName.Suffix = ""
   Set obj.SpousesName=tName

   // Patient ID
   Set obj.MRN="1111222"
  Set obj.AssigningAuthority="EXTERNAL"  // refers to an Assigning Authority entry in the OID Registry

   Set obj.Facility="EXTERNAL"    // refers to a Facility entry in the OID Registry

   // Address 1
   Set addr=##class(HS.Types.Address).%New()
   Set addr.City="Somewhere"
   Set addr.State="SW"
   Set addr.StreetLine="123 Money Street"
   Set addr.Use="HP"   // Primary Home address
   Do obj.Addresses.Insert(addr)

   // Address 2
   Set addr=##class(HS.Types.Address).%New()
   Set addr.City="Anywhere"
   Set addr.StreetLine="456 Any Street"
   Set addr.Use="WP"   // Work Place address
   Do obj.Addresses.Insert(addr)

   //Telephone
   Set tel=##class(HS.Types.Telecom).%New()
   Set tel.PhoneCountryCode="1"
   Set tel.PhoneAreaCode=705
   Set tel.PhoneNumber=5551212
   Set tel.Use="HP"   // Primary Home phone
   Set tel.Type="L"   // Landline
   Do obj.Telecoms.Insert(tel)

   // Alternate ID
   Set tIdent=##class(HS.Types.Identifier).%New()
   Set tIdent.Root="Other.AA"   // refers to an Assigning Authority entry in the OID Registry
   Set tIdent.Extension="98754321"
   Do obj.Identifiers.Insert(tIdent)

   // Send to the routing service (or directly to HS.IHE.PIXv3.Source.Operations)
   Do ##class(HS.Test.Service).SendSync(obj,.r)
   Quit
 }
```

5

Querying an XDS Registry and Retrieving a Document from a Repository

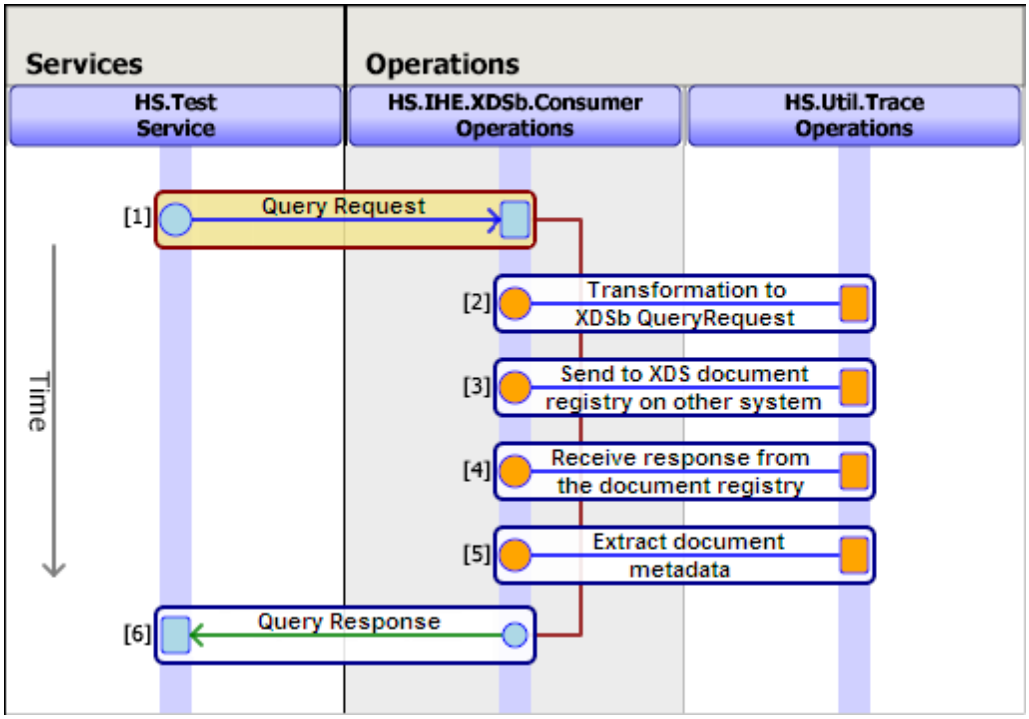
InterSystems products can query an XDS document registry requesting a list of documents for a patient via the IHE “XDS.b Registry Stored Query” transaction. It can then retrieve one or more stored documents from an XDS repository via the IHE “XDS.b Retrieve Document Set” transaction.

IHE supports the idea of an Affinity Domain of healthcare systems that share a common IHE infrastructure. An Affinity Domain might be a RHIO, an IDN, or some other organization. Each Affinity Domain has a single document registry, and may have multiple document repositories. Use the XDS.b profile to query and retrieve documents only within your Affinity Domain. To query and retrieve documents outside of an Affinity Domain as well, use the XCA profile instead.

5.1 XDS Query Message Trace

The following diagram is an annotated XDS query message trace.

The test service shown in the diagram is a simple message router. Trace operations is a utility that makes intermediate processing steps visible in the trace. The numbers in the diagram match the steps in the procedure below.



5.2 XDS Query Procedure

1. You provide an **XDS.b Query Request** message that contains an MPI ID and assigning authority to the InterSystems Document Consumer. You can obtain the MPI ID through a PIX or PDQ query (described above). The query request also specifies the document type and status (for example "approved"), and may also include a list of filters.
2. The InterSystems Document Consumer transforms the message into an IHE "XDSb_QueryRequest" message using an internally-specified transform.
3. The InterSystems Document Consumer then forwards the query to the XDS document registry endpoint on another system that is named in the XDSbRegistryServiceName setting.
4. The XDS document registry on the other system returns a list of available documents along with their metadata and locations.
5. The InterSystems Document Consumer extracts the document metadata from the response using the transformation specified in the TransformToMetadata setting. It then constructs an **XDS.b Query Response** message.
6. The InterSystems Document Consumer returns the **XDS.b Query Response** message, which contains both the original XDS document registry response and the extracted metadata.

5.3 XDS Query Components and Settings

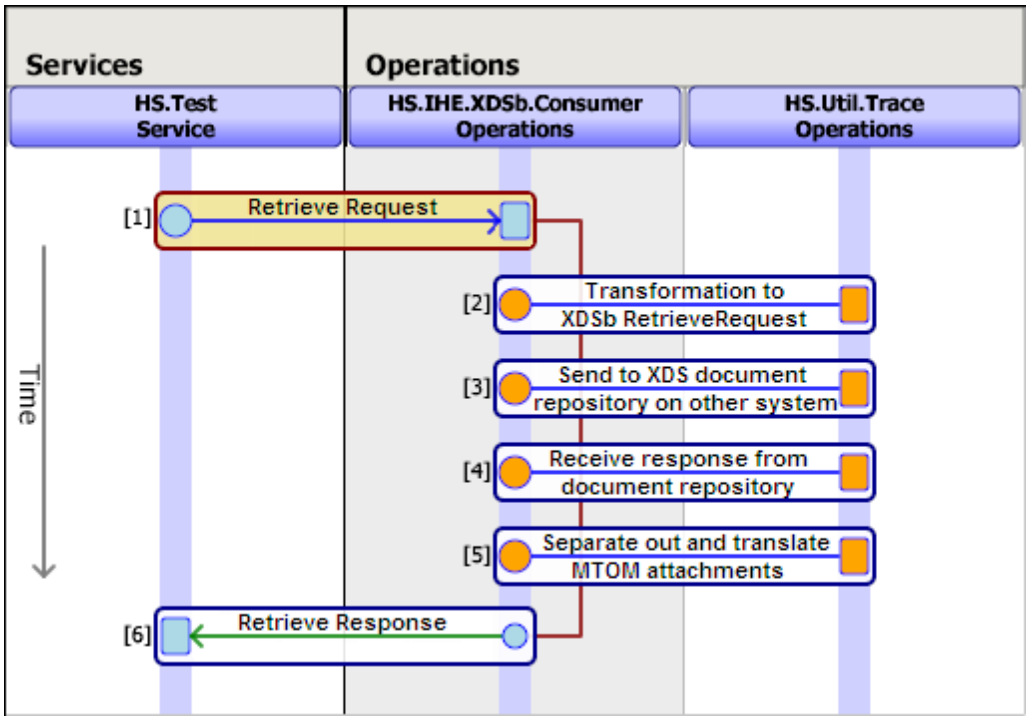
Table 5–1: Components and Settings Used in XDS Query

Component	Setting
Business Hosts	Document Consumer: **HS.HC.IHE.XDSb.Consumer.Operations**
Production Settings	XDSbRegistryServiceName in the Document Consumer
Production Settings	TransformToMetadata in the Document Consumer
Production Messages	**HS.Message.IHE.XDSb.QueryRequest**
Production Messages	**HS.Message.IHE.XDSb.QueryResponse**
XSL Transformations	QueryRequestToXDSbQuery.xsl
XSL Transformations	MessageToMetadata.xsl
Service Registry Entries	XDSb.Registry
External IHE Actor Endpoint	XDS Document Registry

5.4 XDS Retrieve Message Trace

The following diagram is an annotated XDS retrieve message trace.

The test service shown in the diagram is a simple message router. Trace operations is a utility that makes intermediate processing steps visible in the trace. The numbers in the diagram match the steps in the procedure below.



5.5 XDS Retrieve Document Set Procedure

1. Take the query response (above) and construct an **HS.Message.IHE.XDSb.RetrieveRequest** message that contains the document unique ID and repository unique ID (OID) for one or more of the documents listed in the query. Each retrieve request can go to only one repository, so if the query response refers to multiple repositories, you must split these out into separate retrieve requests, one for each repository.

   Send the XDS.b Retrieve Request to the InterSystems Document Consumer.

2. The InterSystems Document Consumer transforms the message into an IHE "XDSb_RetrieveRequest" message using an internally-specified transform.

3. The InterSystems Document Consumer then forwards the request to the XDS document repository endpoint on another system that is indicated by the `<RepositoryUniqueID>` value in the message. This value is an OID. In order to translate the OID into a URL, the following setup is required:

   - An OID registry entry of type "Repository" for the OID.
   - A Service Registry entry that provides the URL of the repository:
     - The Service Registry entry must include the OID registry code in the **Repository** field. This ties the OID and Service Registry entries together.

   Note: If there is a value in the XDSbRepositoryServiceName setting in the InterSystems Document Consumer, then the message will go to the repository specified there rather than to the repository specified in the message. This is useful for testing purposes, but this setting should be blank in a production application.

4. The XDS document repository on the other system responds by providing the requested document(s) as MIME encoded MTOM attachments.

5. The InterSystems Document Consumer separates out the attachments and translates them.

6. The InterSystems Document Consumer returns the MTOM attachments in an XML message of the "RetrieveDocumentSetResponse" variety.

5.6 XDS Retrieve Components and Settings

Table 5–2: Components and Settings Used in XDS Retrieve

Component	Setting
Business Hosts	Document Consumer: **HS.HC.IHE.XDSb.Consumer.Operations**
Production Settings	XDSbRepositoryServiceName in the Document Consumer • For testing purposes only
Production Messages	**HS.Message.IHE.XDSb.RetrieveRequest**
Production Messages	**HS.Message.XMLMessage**: • RetrieveDocumentSetResponse
XSL Transformations	RetrieveRequestToXDSbRetrieve.xsl
Service Registry Entries	XDSb.Repository (or the repository specified in the message)
External IHE Actor Endpoint	XDS Document Repository

5.7 XDS Query and Retrieve Example

Below is a sample XDS query and retrieve:

```
ClassMethod XDSbQueryRetrieve()
 {
     // Create the XDSb Query Request message
     Set MyQuery=##class(HS.Message.IHE.XDSb.QueryRequest).%New()

     // Add the MPI ID, document status and type
     Do MyQuery.AddPatientId("100000002^^^&1.3.6.1.4.1.21367.2010.1.2.300&ISO")
     Do MyQuery.AddStatusValues("Approved")
     Do MyQuery.AddDocumentType(3)

     // Optionally insert other query parameters
     // Do MyQuery.Parameters.Insert(##class(HS.Message.IHE.XDSb.QueryItem).CodedValue(
     // "$XDSDocumentEntryFormatCode",code,scheme))

     // Send the message to the test service (or directly to HS.IHE.XDSb.Consumer.Operations or
     // HS.HC.IHE.XDSb.Consumer.Operations)
     Write ##class(HS.Test.Service).SendSync(MyQuery,.pr)

     Break


     /// XDSbRetrieve ///

     // Assumes you are using the response from the previous query.
     // Currently this is limited to retrieval from a single repository,
     // so if the query response contains multiple repositories, they should
     // be split out into separate retrieve requests.

     // Create the XDSb Retrieve Request message
     Set obj=##class(HS.Message.IHE.XDSb.RetrieveRequest).%New()

     // Use the results of the query to populate the message
     Set obj.Documents=pr.Documents

     // Send the message to the test service (or directly to HS.IHE.XDSb.Consumer.Operations or
     // HS.HC.IHE.XDSb.Consumer.Operations)
     Write ##class(HS.Test.Service).SendSync(obj,.rr)
```

```
    Quit
}
```

6

Providing and Registering Documents to an XDS Document Repository

InterSystems products can provide and register a document to an XDS repository via the IHE “XDS.b Provide and Register Document Set” transaction. If the document uses the Clinical Document Architecture (CDA), then InterSystems can extract the document metadata directly from the document. For all other kinds of documents, for example PDF files, you must provide the document metadata in the message.

6.1 Provide and Register a CDA Document

InterSystems products can provide and register a CDA document to a document repository within the Affinity Domain via the IHE “XDS.b Provide and Register Document Set” transaction (PnR). InterSystems products can extract the metadata from the document itself.

If an MPI ID is not included in the Provide and Register, then a PIX Query is automatically performed. The PIX query uses the extracted MRN to get the MPI ID of the patient.

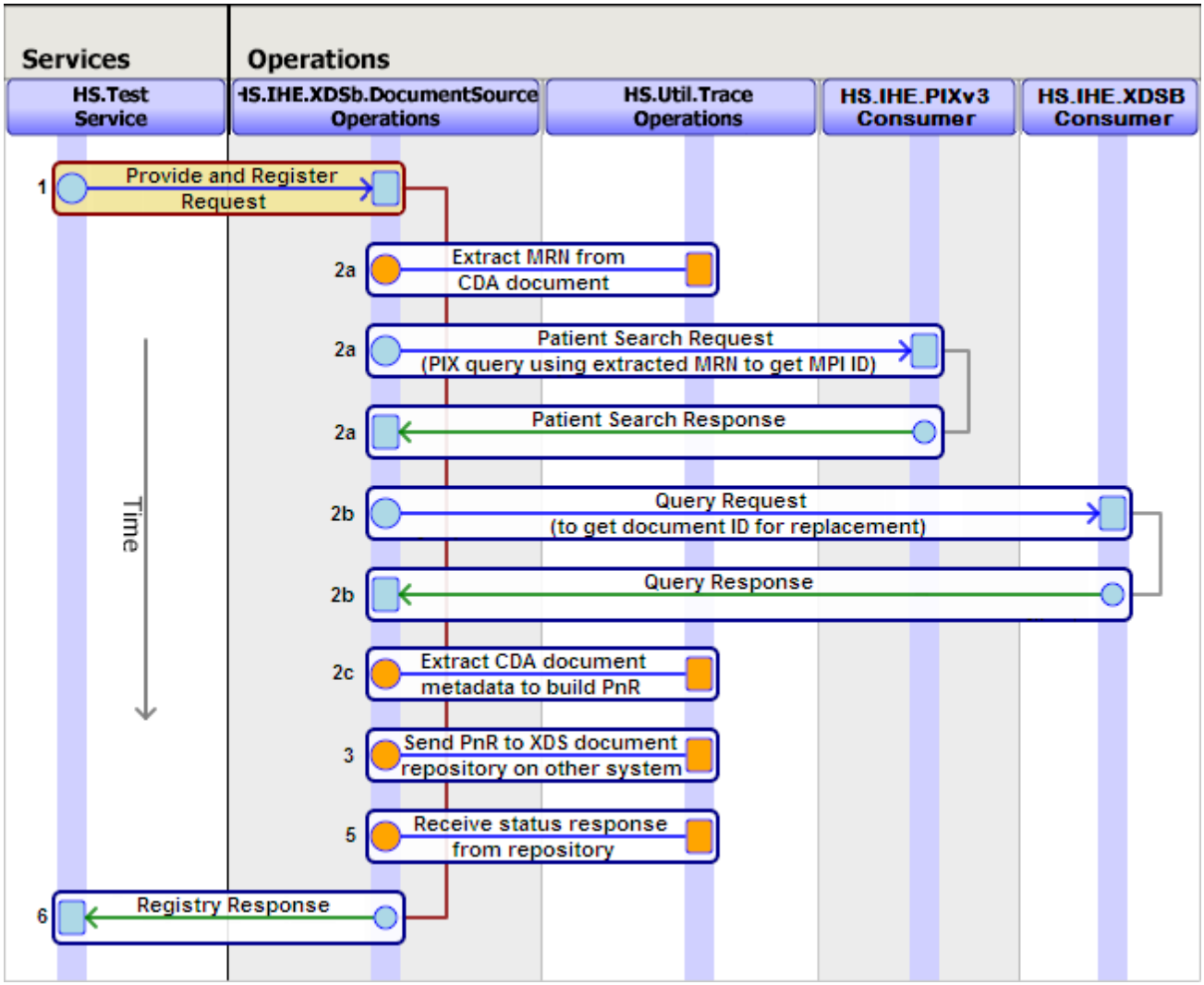
If the CDA is replacing another document (`<ReplacementContext>` is specified in the Provide and Register), then an XDS query is performed to get the unique ID of the document to be replaced.

This section includes:

6.1.1 CDA Provide and Register Message Trace

The following diagram is an annotated XDS provide and register message trace for a CDA document, showing both the optional search request and replacement query.

The test service shown in the diagram is a simple message router. Trace operations is a utility that makes intermediate processing steps visible in the trace. The numbers in the diagram match the steps in the procedure below.



6.1.2 CDA Provide and Register Procedure

1. You provide a **Provide and Register Request** with minimal metadata to the InterSystems Document Source.

   The minimum required document metadata includes the `<MIMEType>`, `<FormatCode>`, and the file contents in `<BodyCharacter>`. When you open the file to include in `<BodyCharacter>`, be sure to open it as the correct file type, for example, open a CDA document in XML format as a text file.

   When a new Provide and Register Request is created, it automatically generates a document unique ID.

2. The InterSystems Document Source extracts the document metadata from the CDA and uses it to build a complete IHE "ProvideAndRegisterDocumentSetRequest":

   a. If an MPI ID is not included in the Provide and Register request, the InterSystems Document Source extracts the MRN and assigning authority from the CDA and performs a PIX query using the procedure described earlier.

   b. If the Provide and Register indicates that this document should replace an existing document, the Document Source performs an XDS query to get the ID of the document to be replaced in the repository. The Provide and Register may include a variety of conditions for the replacement context, but the MPI ID and default entry status of "approved" are included by default if not specified.

   c. Using the XSL transformation specified in the DocumentTransform setting, the Document Source extracts additional metadata from the CDA and uses it to populate the Provide and Register request.

3. The InterSystems Document Source forwards the Provide and Register request to the XDS document repository endpoint on another system that is named in the XDSbRepositoryServiceName setting. The Provide and Register contains both the extracted document metadata and the document itself (the contents of <BodyCharacter>) as an MTOM attachment.

   Note: Because the InterSystems Document Source uses the XDSbRepositoryServiceName setting to determine where to send documents, if you post documents to more than one repository, then you must have a separate document source operation for each repository. Use a message router to determine which document source operation to send the document to, based on whatever criteria you use to determine the appropriate repository for a particular document.

4. The document repository on the other system stores the document and updates the Affinity Domain's document registry with the document metadata provided in the request.
5. The document repository on the other system responds with a success (or failure) message.
6. The InterSystems Document Source returns an XML message of the "RegistryResponse" variety, indicating success or failure. If the transaction fails before the document is sent to the registry, the InterSystems Document Source returns null.

6.1.3 CDA Provide and Register Components and Settings

Table 6–1: Components and Settings Used in Provide and Register of a CDA Document

Component	Setting
Business Hosts	Document Source: **HS.IHE.XDSb.DocumentSource.Operations**
Business Hosts	PIX Consumer: **HS.IHE.PIXv3.Consumer.Operations** • if needed to get MPI ID
Business Hosts	Document Consumer: **HS.HC.IHE.XDSb.Consumer.Operations** • if <Replacement Context> is specified in the message
Production Settings	DocumentTransform in the Document Source
Production Settings	XDSbRepositoryServiceName in the Document Source
Production Messages	**HS.Message.IHE.XDSb.ProvideAndRegisterRequest**
Production Messages	**HS.Message.XMLMessage**: • RegistryResponse
Production Messages	**HS.Message.PatientSearchRequest** (if PIX)
Production Messages	**HS.Message.PatientSearchResponse** (if PIX)
Production Messages	**HS.Message.IHE.XDSb.QueryRequest** (if replacement)
Production Messages	**HS.Message.IHE.XDSb.QueryResponse** (if replacement)

Component	Setting
Minimum Document Metadata when Sending to InterSystems products. (See IHE Metadata Requirements for Third Party Systems for details on minimum IHE requirements.)	Open the file as the correct type, for example, open an XML as a text file. Metadata: • <ContentTypeCode> — indicating the content type of the document • <SourceId> — the OID of the facility that is providing the document • <MIMEType> — usually text/xml • <FormatCode> — indicating the document type • <BodyCharacter> — containing the file contents • <HealthCareFacilityTypeCode> — indicating the facility type • <PracticeSettingCode> — indicating the practice type
XSL Transformations	IHE/XDSb/Version1/DocumentToProvideAndRegister.xsl
Service Registry Entries	XDSb.Repository
Service Registry Entries	PIXv3.Manager (if PIX)
Service Registry Entries	XDSb.Registry (if replacement)
External IHE Actor Endpoint	XDS Document Repository
External IHE Actor Endpoint	PIX Manager (if PIX)
External IHE Actor Endpoint	XDS Document Registry (if replacement)

6.1.4 CDA Provide and Register Example

The method below opens a CDA file and provides the minimum metadata required to provide and register the document. To use the method, go to **HS.IHE.XDSb.DocumentSource.Operations** in your Foundation production and change the value on the **XDSbConsumerOperations** setting from **HS.IHE.XDSb.Consumer.Operations** to **HS.HC.IHE.XDSb.Consumer.Operations**.

```
ClassMethod CDAPnR()
{
    ///Provide and Register a CDA document

    //Create the Provide and Register message, which automatically assigns a document unique ID
    Set tMessage=##class(HS.Message.IHE.XDSb.ProvideAndRegisterRequest).%New()

    // Identify the message source
    Set tMessage.SourceId="1.3.6.1.4.1.21367.1"

    // Create a document instance to hold the document metadata
    Set tDocument = ##class(HS.Message.IHE.XDSb.Document).%New()

    // Open the document file and insert it into <BodyCharacter> (for non-CDA, this is <Body>).
    // The document will become an MTOM attachment in the outbound message.
    // In this case, we are providing a Consolidated CDA (C-CDA) in XML,
    // so open the file as a text file.
    Set tFile = ##class(%File).%New()
    Set tFile.Name="C:\wtemp\testccda.xml"
    Do tFile.Open("R")
    Do tFile.Rewind()
    Do tDocument.BodyCharacter.CopyFrom(tFile)
    Kill tFile

    // Set the required minimum document metadata. For CDA, the <MIMEType> is "text/xml"
    Set tDocument.MimeType="text/xml"
    Set tDocument.FormatCode=##class(HS.IHE.XDSb.Types.CodedValue).Create(
          "urn:hl7-org:sdwg:ccda-structuredBody:1.1","1.3.6.1.4.1.19376.1.2.3","Consolidated CDA R1.1
 Structured Body Document")
    Set tDocument.HealthcareFacilityTypeCode=##class(HS.IHE.XDSb.Types.CodedValue).Create(
```

```
          "22232009”,”2.16.840.1.113883.6.96","Hospital")
    Set tDocument.PracticeSettingCode=##class(HS.IHE.XDSb.Types.CodedValue).Create(
          "394802001","2.16.840.1.113883.6.96","General Medicine")

    // This is optional and controls replacement. If you specify a ReplacementContext,
    // then an XDS.B query is performed to obtain the document unique ID
    // of documents that match the context.
    //
    // In the case below, it is looking for documents that match the specified format code.
    // You may include other context items as well.
    // InterSystems automatically adds the Patient ID and the Status of "approved"
    // to the context.
    Set tContext = ##class(HS.Message.IHE.XDSb.QueryItem).CodedValue(

"$XDSDocumentEntryFormatCode","urn:hl7-org:sdwg:ccda-structuredBody:1.1","1.3.6.1.4.1.19376.1.2.3")

    Do tDocument.ReplacementContext.Insert(tContext)

    /// Insert the document metadata into the message
    Do tMessage.Documents.Insert(tDocument)

    /// Send to the routing service (or directly to HS.IHE.XDSb.DocumentSource.Operations)
    Write ##class(HS.Test.Service).SendSync(tMessage,.rr)

    quit
}
```

6.2 Provide and Register a Non-CDA Document

InterSystems products can provide and register any kind of document to a document repository within the Affinity Domain via the IHE “XDS.b Provide and Register Document Set” transaction (PnR). You must provide your InterSystems product with the document metadata.

If the CDA is replacing another document (`<ReplacementContext>` is specified in the Provide and Register), then an XDS query is performed to get the unique ID of the document to be replaced.

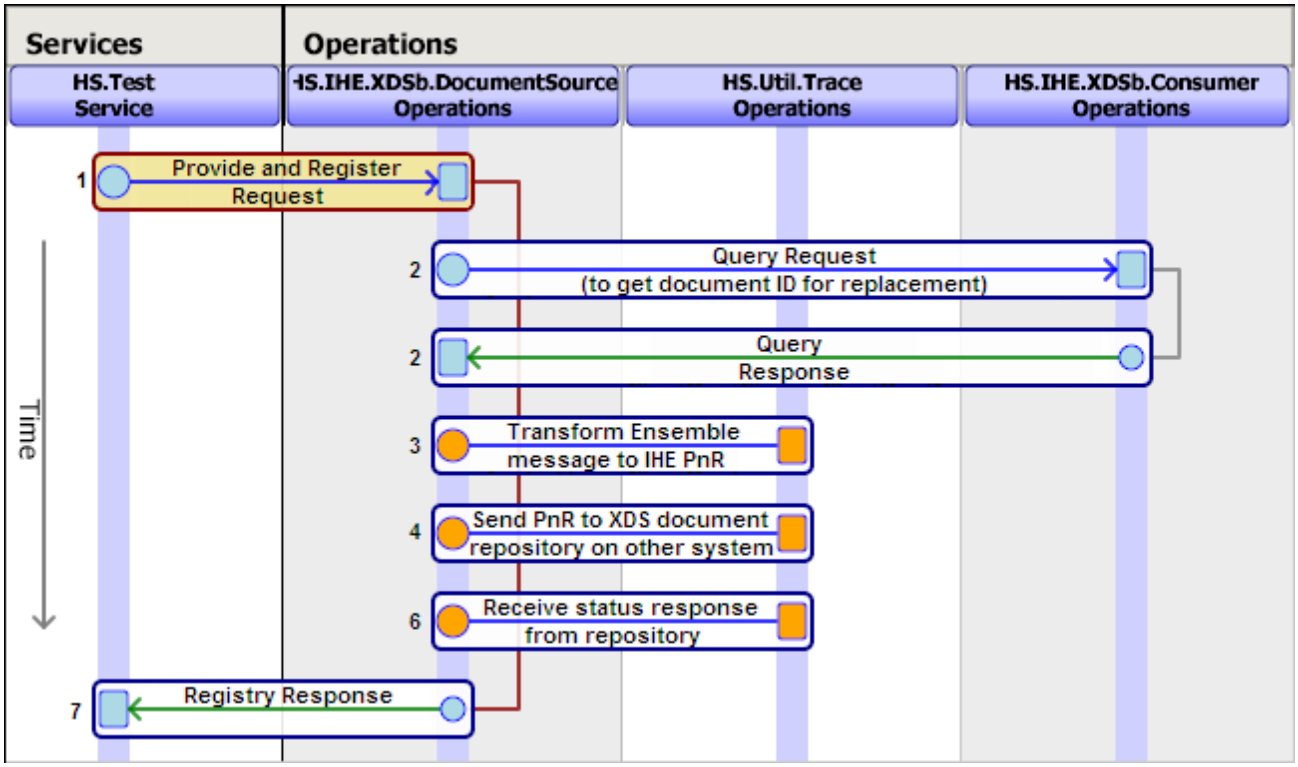
This section includes:

- Non-CDA provide and register message trace
- Non-CDA provide and register procedure
- Non-CDA provide and register components
- Non-CDA provide and register example

6.2.1 Non-CDA Provide and Register Message Trace

The following diagram is an annotated XDS provide and register message trace for a non-CDA document, that includes the query request for replacement context.

The test service shown in the diagram is a simple message router. Trace operations is a utility that makes intermediate processing steps visible in the trace. The numbers in the diagram match the steps in the procedure below.



6.2.2 Provide and Register a Non-CDA Document Procedure

InterSystems products can provide and register any clinical document (for example a PDF) to a document repository within the Affinity Domain via the IHE "XDS.b Provide and Register Document Set" transaction (PnR).

1. You provide a **Provide and Register Request** with complete document metadata to the InterSystems Document Source. You must include the MPI ID of the patient in `<SourcePatientId>` in the message. If you only have the MRN, perform a PIX query to get the MPI ID before creating the message.

   The minimum required document metadata includes the `<MIMEType>`, `<FormatCode>`, `<Body>`, `<SourcePatientInfo>`, and author information. When you open the file to include in `<Body>`, be sure to open it as the correct file type, for example open a PDF file as a binary.

   When a new Provide and Register Request is created, it automatically generates a document unique ID.

2. If the Provide and Register indicates that this document should replace an existing document, the Document Source performs an XDS Query to get the ID of the document to be replaced in the repository. The Provide and Register may include a variety of conditions for the replacement context, but the MPI ID and default entry status of "approved" are included by default if not specified.

3. The InterSystems Document Source transforms the message into an IHE "ProvideAndRegisterDocumentSetRequest" using the XSL transformation specified in the DocumentTransform setting.

4. The InterSystems Document Source forwards the Provide and Register request to the XDS document repository endpoint on another system that is named in the XDSbRepositoryServiceName setting. The Provide and Register contains both the document metadata and the document itself (the contents of `<Body>`) as an MTOM attachment.

5. The XDS document repository on the other system stores the document and updates the Affinity Domain's XDS document registry with the document metadata provided in the request.

6. The XDS document repository on the other system responds with a success (or failure) message.

7. The InterSystems Document Source returns an XML message of the "RegistryResponse" variety, indicating success or failure. If the transaction fails before the document is sent to the registry, the InterSystems Document Source returns null.

6.2.3 Non-CDA Provide and Register Components and Settings

Table 6–2: Components and Settings Used in Provide and Register of a non-CDA Document

Component	Setting
Business Hosts	Document Source: **HS.IHE.XDSb.DocumentSource.Operations**
Business Hosts	Document Consumer: **HS.HC.IHE.XDSb.Consumer.Operations** • if `<Replacement Context>` is specified in the message
Production Settings	XDSbRepositoryServiceName in the Document Source
Production Messages	**HS.Message.IHE.XDSb.ProvideAndRegisterRequest**
Production Messages	**HS.Message.XMLMessage**: • RegistryResponse
Production Messages	**HS.Message.IHE.XDSb.QueryRequest** (if replacement)
Production Messages	**HS.Message.IHE.XDSb.QueryResponse** (if replacement)
Minimum Document Metadata when Sending to InterSystems products • See IHE Metadata Requirements for Third Party Systems for details on minimum IHE requirements • See the class reference for the **Provide and Register** request message for complete document metadata details.	Open the file as the correct type, for example, open a PDF as a binary. Metadata: • `<MIMEType>` — for example, `application/pdf`. See iana.org for a complete list of MIME types. • `<FormatCode>` — indicating document type • `<Body>` — containing the file contents • `<SourcePatientInfo>` — patient demographics as HL7 segments • `<HealthCareFacilityTypeCode>` — indicating the facility type • `<PracticeSettingCode>` — indicating the practice type • `<Author...>` (...Institution, ...Role, ...Person, ...Specialty)
XSL Transformations	IHE/XDSb/Version1/DocumentToProvideAndRegister.xsl
Service Registry Entries	XDSb.Repository
Service Registry Entries	XDSb.Registry (if replacement)
External IHE Actor Endpoints	XDS Document Repository
External IHE Actor Endpoints	XDS Document Registry (if replacement)

6.2.4 Provide and Register a Non-CDA Document Example

The method below generates an XDS provide and register for a PDF document. To use the method, go to **HS.IHE.XDSb.DocumentSource.Operations** in your Foundation production and change the value on the **XDSbConsumerOperations** setting from **HS.IHE.XDSb.Consumer.Operations** to **HS.HC.IHE.XDSb.Consumer.Operations**.

```
ClassMethod CDAPnR2()
{
 // Create the message, which automatically assigns a document unique ID
 Set tMessage=##class(HS.Message.IHE.XDSb.ProvideAndRegisterRequest).%New()

 // Provide the SubmissionSet metadata: Patient ID (MPIID & Home Community OID),
 // document source, and content code:
 //
 // - Use a PIX query to obtain the Patient ID if you have only the MRN.
 //
 // - The SourceId should be the OID of a facility associated with
 //   your repository.
 //
 // - Select a ContentTypeCode from the Coded Entry Registry (HS.IHE.CodedEntry)
 //   The "Scheme" OID in the example below "2.16.840.1.113883.6.1" means LOINC
 //
 Set tMessage.PatientId="100000001^^^&1.3.6.1.4.1.21367.2010.1.2.300&ISO"
 Set tMessage.SourceId="1.3.6.1.4.1.21367.2010.1.2.300.2.0"
 Set tMessage.ContentTypeCode = ##class(HS.IHE.XDSb.Types.CodedValue).Create(
     "11488-4","2.16.840.1.113883.6.1","Consultation Note")

 // Create a document instance to hold the document metadata
 //
 Set tDocument = ##class(HS.Message.IHE.XDSb.Document).%New()

 // Open the document file and insert it into "Body" (for CDA, this is "CharacterBody").
 // The document will become an MTOM attachment in the outbound message.
 // In this case, we are providing a PDF, so open the file as a binary.
 //
 Set tFile = ##class(%Stream.FileBinary).%New()
 Set tFile.Filename="C:\wtemp\testdoc.pdf"
 Do tFile.Rewind()
 Do tDocument.Body.CopyFrom(tFile)
 Kill tFile

 // Set the "MIMEType", for a PDF, this is "application/pdf".
 // See http://www.iana.org/assignments/media-types/index.html for a list of valid types.
 //
 Set tDocument.MimeType="application/pdf"

 // Enter the document metadata
 // Note that document creation time uses YYYYMMDDHHMMSS-offset format according to the IHE
 // specification. The following will produce the current time in the correct format:
 //   $REPLACE($TR($ZDT($H,3,6),":T"),"-","",,2)
 //
 Set tDocument.CreationTime="20180821102615-0400"
 Set tDocument.LanguageCode="en-CA"

 Set tDocument.ClassCode=##class(HS.IHE.XDSb.Types.CodedValue).Create(
     "26435-8","2.16.840.1.113883.6.1","MOLECULAR PATHOLOGY STUDIES")
 Set tDocument.FormatCode=##class(HS.IHE.XDSb.Types.CodedValue).Create(
     "urn:ihe:iti:xds-sd:pdf:2008","1.3.6.1.4.1.19376.1.2.3","Scanned Documents PDF")
 Set tDocument.HealthcareFacilityTypeCode=##class(HS.IHE.XDSb.Types.CodedValue).Create(
     "OF","2.16.840.1.113883.5.11","Outpatient facility")
 Set tDocument.PracticeSettingCode=##class(HS.IHE.XDSb.Types.CodedValue).Create(
     "394802001","2.16.840.1.113883.6.96","General Medicine")
 Set tDocument.TypeCode=##class(HS.IHE.XDSb.Types.CodedValue).Create(
     "18768-2","2.16.840.1.113883.6.1","CELL COUNTS+DIFFERENTIAL STUDIES")

 Do tDocument.ConfidentialityCode.Insert(##class(HS.IHE.XDSb.Types.CodedValue).Create(
     "N","2.16.840.1.113883.5.25","Normal"))
 Do tDocument.EventCodeList.Insert(##class(HS.IHE.XDSb.Types.CodedValue).Create(
     "1.2.840.10065.1.12.1.13","1.2.840.10065.1.12","Review Signature"))

 // Continued...
 // Patient demographics
 Set tDocument.SourcePatientId="1111222^^^&1.3.6.1.4.1.21367.2010.1.2.310&ISO"
 Do tDocument.SourcePatientInfo.Insert(##class(HS.IHE.XDSb.Types.SlotValue).%New(
     "PID-3|1111222^^^&1.3.6.1.4.1.21367.2010.1.2.310&ISO"))
 Do tDocument.SourcePatientInfo.Insert(##class(HS.IHE.XDSb.Types.SlotValue).%New(
     "PID-5|Smith^James^"))
 Do tDocument.SourcePatientInfo.Insert(##class(HS.IHE.XDSb.Types.SlotValue).%New(
     "PID-7|20000930"))
```

```
 Do tDocument.SourcePatientInfo.Insert(##class(HS.IHE.XDSb.Types.SlotValue).%New("PID-8|M"))
 Do tDocument.SourcePatientInfo.Insert(##class(HS.IHE.XDSb.Types.SlotValue).%New(
     "PID-11|123 Money Street^^Somewhere^SW^"))

 // Document author
 Set tAuthor= ##class(HS.IHE.XDSb.Types.Author).%New()
 Set tAuthor.AuthorPerson="John Smith"
 Do tAuthor.AuthorInstitution.Insert(##class(HS.IHE.XDSb.Types.SlotValue).%New("Johns Hopkins"))
 Do tAuthor.AuthorRole.Insert(##class(HS.IHE.XDSb.Types.SlotValue).%New("Role"))
 Do tAuthor.AuthorSpecialty.Insert(##class(HS.IHE.XDSb.Types.SlotValue).%New("Specialty"))
 Do tDocument.Author.Insert(tAuthor)

 // This is optional and controls replacement. If you specify a ReplacementContext, then an XDS.B
 // query is performed to obtain the document unique ID of documents that match the context.
 //
 // In the case below, it is looking for documents that match the specified eventcode.
 // You may include other context items as well.
 // InterSystems automatically adds the Patient ID and the Status of
 // "approved" to the context.
 Set tContext = ##class(HS.Message.IHE.XDSb.QueryItem).CodedValue("$XDSDocumentEntryEventCodeList",
    "1.2.840.10065.1.12.1.13","1.2.840.10065.1.12")
 Do tDocument.ReplacementContext.Insert(tContext)

 // Insert the document metadata into the message
 Do tMessage.Documents.Insert(tDocument)

 // Send the message to the test service (or directly to HS.IHE.XDSb.DocumentSource.Operations)
 Write ##class(HS.Test.Service).SendSync(tMessage,.rr)

 // To trap errors when using the Test Service, use the following instead
 //
 // Set tSC = ##class(HS.Test.Service).SendSync(tMessage,.rr)
 // if $$$ISOK(tSC) {
 //   set statusSuccess=rr.ContentStream.FindAt(
 //      1,"urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Success")
 //   set statusFail=rr.ContentStream.FindAt(
 //      1,"urn:oasis:names:tc:ebxml-regrep:ResponseStatusType:Failure")
 //   if statusFail0 { w !, "Response Status was a failure!"
 //      Do rr.ContentStream.Rewind() w !, rr.ContentStream.Read(231), " ... " }
 //   if statusSuccess>0 { w !, "Response Status was Success"  }
 // } else { Do $System.Status.DisplayError(tSC) }

 Quit
}
```

6.3 IHE Metadata Requirements for Third Party Systems

The tables below indicate what metadata attributes are required by the IHE specification. When sending a document to a third party system, they may require all or a subset of the required metadata attributes. For example, when providing a CDA document to InterSystems products, it does not require the source patient ID to appear in the metadata as it can extract this information from the document. Other systems may not support that behavior.

Table 6–3: DocumentEntry Metadata Attribute Requirements (Required* means "required if known")

Metadata Attribute	XDS.b Provide and Register	XDS.b Register	XDM Distribute	XDR Provide and Register	XDR (Meta-data-limited) Provide and Register
author	**Required**	*Required**	*Required**	*Required**	*Required**
availabilityStatus	Optional	Optional	Optional	Optional	Optional
classCode	**Required**	**Required**	*Required**	**Required**	*Required**
comments	Optional	Optional	Optional	Optional	Optional

Metadata Attribute	XDS.b Provide and Register	XDS.b Register	XDM Distribute	XDR Provide and Register	XDR (Meta-data-limited) Provide and Register
confidentialityCode	**Required**	**Required**	*Required**	**Required**	*Required**
creationTime	**Required**	**Required**	*Required**	**Required**	*Required**
entryUUID	**Required**	**Required**	**Required**	**Required**	**Required**
eventCodeList	Optional	Optional	Optional	Optional	Optional
formatCode	**Required**	**Required**	*Required**	**Required**	*Required**
hash	Optional	**Required**	**Required**	Optional	Optional
healthcareFacility TypeCode	**Required**	**Required**	*Required**	**Required**	*Required**
homeCommunityId	Optional	Optional	Optional	Optional	Optional
languageCode	**Required**	**Required**	*Required**	**Required**	*Required**
legalAuthenticator	Optional	Optional	Optional	Optional	Optional
mimeType	**Required**	**Required**	**Required**	**Required**	**Required**
patientId	**Required**	**Required**	*Required**	**Required**	*Required**
practiceSettingCode	**Required**	**Required**	*Required**	**Required**	*Required**
repositoryUniqueId	Optional	Required	Optional	Optional	Optional
serviceStartTime	*Required**	*Required**	*Required**	*Required**	*Required**
serviceStopTime	*Required**	*Required**	*Required**	*Required**	*Required**
size	Optional	**Required**	**Required**	Optional	Optional
sourcePatientId	**Required**	**Required**	*Required**	**Required**	*Required**
sourcePatientInfo	Optional	Optional	*Required**	Optional	*Required**
title	Optional	Optional	Optional	Optional	Optional
typeCode	**Required**	**Required**	*Required**	**Required**	*Required**
uniqueId	**Required**	**Required**	**Required**	**Required**	**Required**
URI	Optional	Optional	**Required**	Optional	Optional

Table 6–4: SubmissionSet Metadata Attribute Requirements (Required means "required if known")*

Metadata Attribute	XDS.b Provide and Register	XDS.b Register	XDM Distribute	XDR Provide and Register	XDR (Meta-data-limited) Provide and Register
author	*Required**	*Required**	*Required**	*Required**	*Required**
availabilityStatus	Optional	Optional	Optional	Optional	Optional
comments	Optional	Optional	Optional	Optional	Optional
contentTypeCode	**Required**	**Required**	*Required**	**Required**	*Required**
entryUUID	**Required**	**Required**	**Required**	**Required**	**Required**
homeCommunityId	Optional	Optional	Optional	Optional	Optional
intendedRecipient	Optional	Optional	*Required**	*Required**	*Required**
patientId	**Required**	**Required**	*Required**	**Required**	*Required**
sourceId	**Required**	**Required**	**Required**	**Required**	**Required**
submissionTime	**Required**	**Required**	**Required**	**Required**	**Required**

7

Querying and Retrieving a Document from another Affinity Domain (XCA)

IHE supports the idea of an Affinity Domain of healthcare systems that share a common IHE infrastructure. An Affinity Domain might be a RHIO, an IDN, or some other organization. Each Affinity Domain has a single document registry, and may have multiple document repositories.

InterSystems products can look up patient identifiers in other Affinity Domains via the XCPD *Cross-Gateway Patient Discovery* transaction. With the patient identifier, you may then query document registries and request a list of documents via the XCA *Cross-Gateway Query* transaction. It can then retrieve one or more stored documents from the specified repositories via the XCA *Retrieve Document Set* transaction.

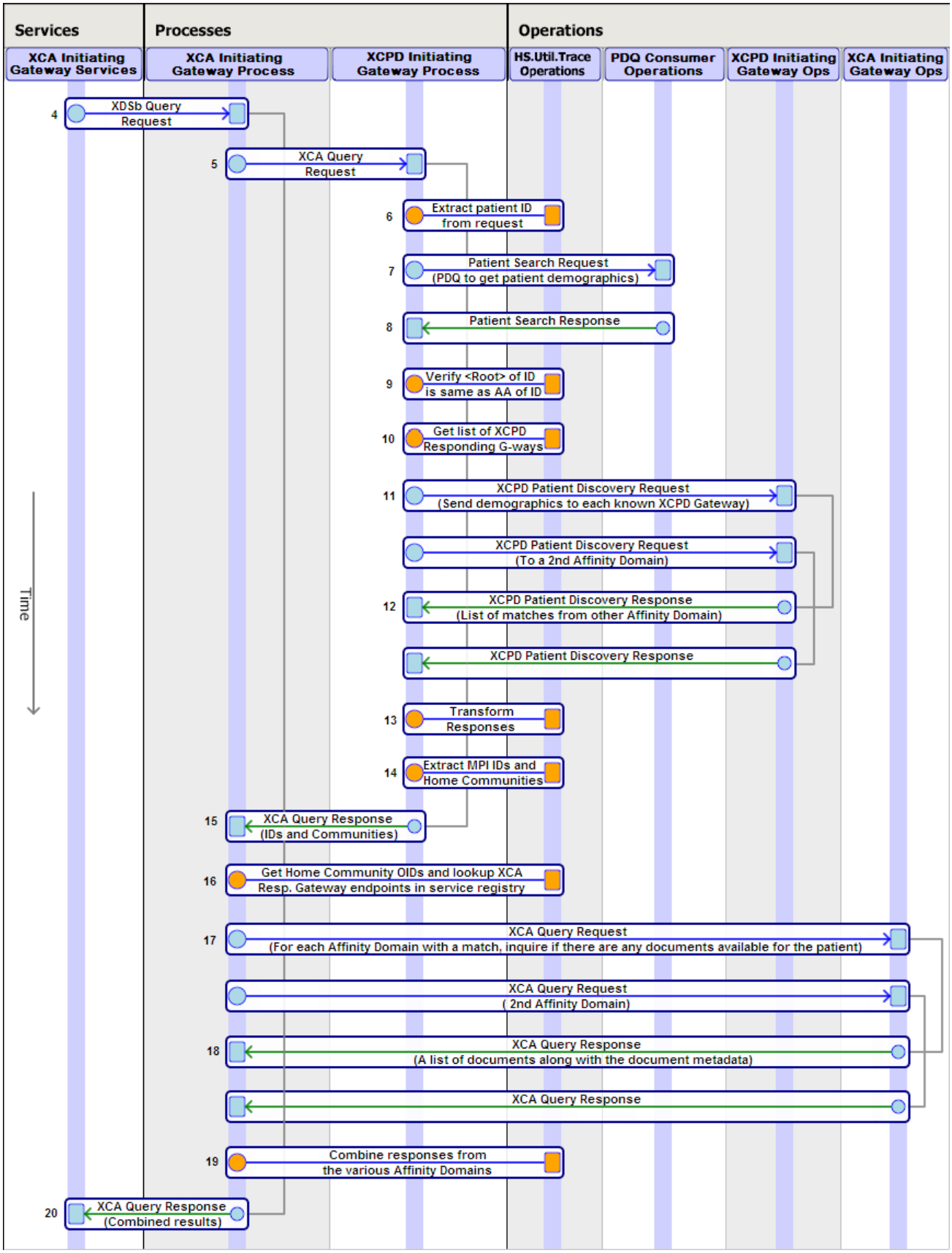
From the outside, XCA is essentially the same as XDS, except for the patient discovery. It uses the same messaging infrastructure as XDS, and you can set up InterSystems products so that document queries search both the local repository and other Affinity Domains.

This chapter includes:

7.1 XCA Query Message Trace

The following diagram is an annotated XCA Query message trace, illustrating the patient discovery (XCPD) and the document query.

The trace operations shown in the diagram is a utility that makes intermediate processing steps visible in the trace. The numbers in the diagram match the steps in the procedure below. The first 3 steps and last 3 steps in the procedure appear in a different session, and are basically identical to the steps in the XDS.b query transaction.


Services
Processes
Operations
XCA Initiating Gateway Services
XCA Initiating Gateway Process
XCPD Initiating Gateway Process
HS.Util.Trace Operations
PDQ Consumer Operations
XCPD Initiating Gateway Ops
XCA Initiating Gateway Ops
Time
4 XDSb Query Request
5 XCA Query Request
6 Extract patient ID from request
7 Patient Search Request (PDQ to get patient demographics)
8 Patient Search Response
9 Verify <Root> of ID is same as AA of ID
10 Get list of XCPD Responding G-ways
11 XCPD Patient Discovery Request (Send demographics to each known XCPD Gateway)
XCPD Patient Discovery Request (To a 2nd Affinity Domain)
12 XCPD Patient Discovery Response (List of matches from other Affinity Domain)
XCPD Patient Discovery Response
13 Transform Responses
14 Extract MPI IDs and Home Communities
15 XCA Query Response (IDs and Communities)
16 Get Home Community OIDs and lookup XCA Resp. Gateway endpoints in service registry
17 XCA Query Request (For each Affinity Domain with a match, inquire if there are any documents available for the patient)
XCA Query Request (2nd Affinity Domain)
18 XCA Query Response (A list of documents along with the document metadata)
XCA Query Response
19 Combine responses from the various Affinity Domains
20 XCA Query Response (Combined results)

7.2 XCA Query Procedure

An XCA query transaction in InterSystems products begins with an XDS query request message. Rather than pointing the XDS Document Consumer to a particular repository, the consumer points instead to the XCA Initiating Gateway, which knows about both the local repository and the repositories in other Affinity Domains. The easiest way to set this up is to create a Service Registry entry that contains the URL of the Initiating Gateway service and call it, for example, "XCA.IG." Then set the XDSbRegistryServiceName setting in your XCA Document Consumer to `XCA.IG`.

The full XCA query procedure is documented below. The first 3 steps and the last 3 steps occur in a separate session, and are basically identical to an XDS.b query transaction. They are not shown in the diagram.

1. You provide an **XDSb.QueryRequest** message that contains an MPI ID and assigning authority to the XDS.b Document Consumer. The query request also specifies the document type and status (for example "approved"), and may also include a list of filters.

   You can obtain the MPI ID through a PIX or PDQ query (described above).

2. The document consumer transforms the message into an IHE "XDSb_QueryRequest" message using the TransformRetrieveToXDSb setting.

3. The document consumer then forwards the query to the XCA Initiating Gateway service that is indicated in the XDSbRegistryServiceName setting. The rest of the procedure takes place in a new session until the document consumer receives its response.

4. The XCA Initiating Gateway service forwards the message to the XCPD Initiating Gateway process named in the XCPDInitiatingGatewayProcess setting.

5. The XCA Initiating Gateway process transforms the message into an IHE "XCA_QueryRequest" message using an internally-specified transform, and forwards it to the XCPD Initiating Gateway process to begin the patient discovery.

6. The XCPD Initiating Gateway process extracts the MPI ID from the request and constructs a Patient Search Request message.

7. The XCPD Initiating Gateway process gets the patient's demographics by sending the Patient Search Request to the PDQ consumer named in the PDQv3Consumer setting.

8. The PDQ consumer returns the demographics.

9. The XCPD Initiating Gateway process confirms that the `<Root>` element of the `<LivingSubjectID>` returned from the PDQ is the same as the assigning authority of the provided MPI ID.

10. The XCPD Initiating Gateway process checks the XCPDQueryServiceNames setting. This setting should contain a comma-separated list of Service Registry entries that point to XCPD Responding Gateway endpoints in other Affinity Domains.

11. The XCPD Initiating Gateway process sends one XCPD Patient Discovery Request to the XCPD Initiating Gateway operation for each known XCPD Responding Gateway.

12. The XCPD Initiating Gateway operation forwards the discovery requests to the XCPD Responding Gateways in the other Affinity Domains. It then returns XCPD Patient Discovery Responses from the XCPD Responding Gateways to the XCPD Initiating Gateway process. Each response contains the MPI IDs and demographics of zero, one, or several possible patients that match the provided demographics. These are exactly like PDQ responses.

13. The XCPD Initiating Gateway process transforms the responses and replaces the Home Community OIDs with Home Community IDs from the OID registry.

14. The XCPD Initiating Gateway process extracts the MPI IDs and Home Communities (assigning authorities) for the unique matches, discarding any that match multiple patients within a Home Community.

15. The XCPD Initiating Gateway process returns an IHE "XCA_QueryResponse" message to the XCA Initiating Gateway process that contains a list of MPI IDs and Home Community IDs, with at most a single entry for each Home Community.
16. The XCA Initiating Gateway process uses the Home Community IDs to get the Home Community OIDs. In order to translate each OID into a URL, the following setup is required:
    - An OID registry entry of type "HomeCommunity" for each Home Community OID.
    - A Service Registry entry for each Home Community that provides the URL of the XCA Responding Gateway actor in that community:
        - Each Service Registry entry must include the OID registry code in the **HomeCommunity** field. This ties the OID and Service Registry entries together.
        - Each Service Registry entry must include `XCA.Query` in the **Device Function** field. This indicates that it is the "XCA query device" for this community.
17. For each match returned by the XCPD Initiating Gateway process, the XCA Initiating Gateway process sends an XCA_QueryRequest message to the XCA Initiating Gateway operation named in the XCAInitiatingGatewayOperation setting.

    If there is a value in the XDSbQueryServiceName field, which should point to the local XDS registry, then the XCA Initiating Gateway process also sends an XDSb_QueryRequest to the XCA Initiating Gateway operation. This will trigger a document search in the local XDS registry.
18. The XCA Initiating Gateway operation forwards the query requests to the XCA Responding Gateways (and possibly the local registry). It then returns the query responses from the XCA Responding Gateways (and local registry) to the XCA Initiating Gateway Process. Each response indicates the document metadata and location for each document available for the patient.
19. The XCA Initiating Gateway Process combines the various responses.
20. The XCA Initiating Gateway Process returns an XML message of the "XCA_QueryResponse" variety to the XCA Initiating Gateway service.
21. The XCA Initiating Gateway service returns the combined responses to the XDS.b Document Consumer in the original session in an XML message of the "XDSb_QueryResponse" variety.
22. The document consumer extracts the document metadata from the response using the transformation specified in the TransformToMetadata setting. It then constructs an **HS.Message.IHE.XDSb.QueryResponse** message.
23. The document consumer returns the **XDS.b Query Response** message, which contains both the original XCA and XDS.b responses and the extracted metadata. This message can be used to initiate an XCA retrieve, as described in the XCA Retrieve procedure.

7.3 XCA Query Components and Settings

Table 7–1: Components and Settings Used in XCA Query

Component	Setting
Business Hosts	XCA Document Consumer: **HS.HC.IHE.XDSb.Consumer.Operations**
Business Hosts	XCA Initiating Gateway Service: **HS.IHE.XCA.InitiatingGateway.Services**
Business Hosts	XCA Initiating Gateway Process: **HS.IHE.XCA.InitiatingGateway.Process**

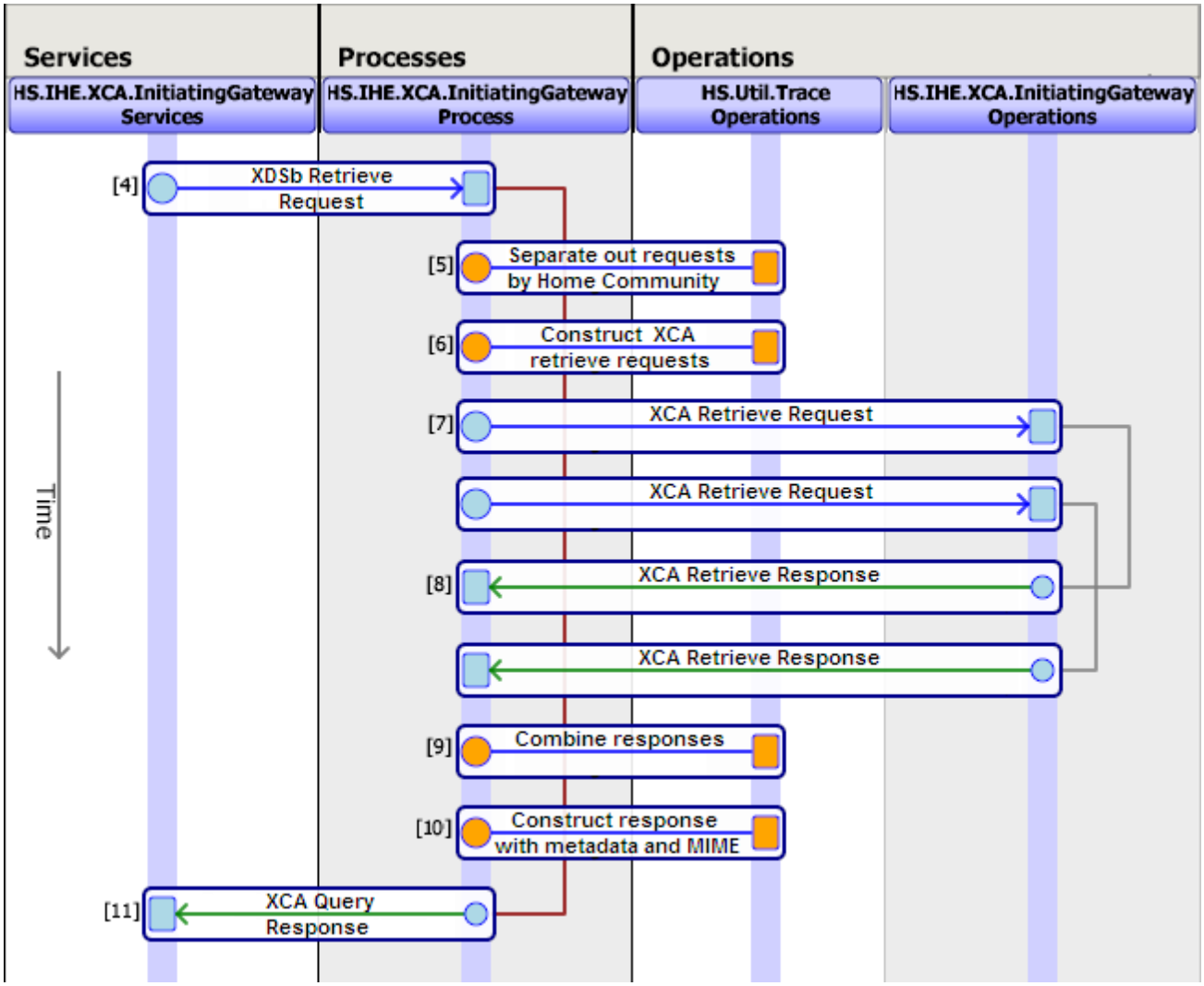
Component	Setting
Business Hosts	XCA Initiating Gateway Operation: **HS.IHE.XCA.InitiatingGateway.Operations**
Business Hosts	XCPD Initiating Gateway Process: **HS.IHE.XCPD.InitiatingGateway.Process**
Business Hosts	XCPD Initiating Gateway Operation: **HS.IHE.XCPD.InitiatingGateway.Operation**
Business Hosts	PDQ Consumer: **HS.IHE.PDQv3.Consumer.Operations**
Production Settings	XDSbRegistryServiceName in the XDS.b Document Consumer: • Set to location of the XCA Initiating Gateway service.
Production Settings	XCPDInitiatingGatewayProcess in the XCA Initiating Gateway service
Production Settings	PDQv3Consumer in the XCPD Initiating Gateway process
Production Settings	Service Name in the PDQ Consumer operation
Production Settings	XCPDQueryServiceNames in the XCPD Initiating Gateway process: • A comma-separated list of Service Registry entries pointing to XCPD Responding Gateway endpoints on other systems.
Production Settings	XCAInitiatingGatewayOperation in the XCA Initiating Gateway process
Production Settings	XDSbQueryServiceName in the XCA Initiating Gateway process: • A Service Registry entry with the URL of your local XDS.b registry. If you enter a value here, then the XCA Initiating Gateway can be used for both XCA and XDS.b queries.
Production Settings	TransformToMetadata in the XDS.b Document Consumer
Production Messages	**HS.Message.IHE.XDSb.QueryRequest**
Production Messages	**HS.Message.IHE.XDSb.QueryResponse**
Production Messages	**HS.Message.XMLMessage**: • XDSb_QueryRequest • XDSb_QueryResponse • XCA_QueryRequest • XCA_QueryResponse • XCPD_PatientDiscoveryRequest • XCPD_PatientDiscoveryResponse
Production Messages	**HS.Message.PatientSearchRequest** (for PDQ query)
Production Messages	**HS.Message.PatientSearchResponse** (for PDQ query)
XSL Transformations	QueryRequestToXDSbQuery.xsl in the XDS.b Document Consumer
XSL Transformations	IHE/PDQ/Version1/PatientSearchToPRPAIN201305UV.xsl in PDQ
XSL Transformations	IHE/PDQ/Version1/PRPAIN201306UVToPatientSearchResponse.xsl in PDQ

Component	Setting
XSL Transformations	Message-To-Metadata.xsl in the XDS.b Document Consumer
Service Registry Entries	XCA.IG (or similar) • points to the XCA Initiating Gateway service
Service Registry Entries	PDQv3.Supplier (for PDQ query)
Service Registry Entries	XCPD.RespondingGateway.1 XCPD.RespondingGateway.2 ...etc: • URLS of the various XCPD Responding Gateways in the Home Communities
Service Registry Entries	An XCA.Query device entry for each Home Community that points to the URL of the XCA Responding Gateway endpoint in that community. The OID registry code for the community should appear in the **HomeCommunity** field, and `XCA.Query` should appear in the **Device Function** field.
OID Registry Entries	A HomeCommunity OID for each Home Community
External IHE Actor Endpoints	PDQ Supplier
External IHE Actor Endpoints	XCPD Responding Gateway(s)
External IHE Actor Endpoints	XCA Responding Gateway(s)

7.4 XCA Retrieve Message Trace

The following diagram is an annotated XCA document retrieve message trace.

The trace operations shown in the diagram is a utility that makes intermediate processing steps visible in the trace. The numbers in the diagram match the steps in the procedure below. The first 3 steps and last 3 steps in the procedure appear in a different session, and are basically identical to the steps in the XDS.b retrieve transaction.



7.5 XCA Retrieve Procedure

The XCA retrieve procedure is documented below. The first 3 steps and the last 3 steps occur in a separate session, and are basically identical to an XDS.b retrieve transaction. They are not shown in the diagram.

1. Take the XDS.b query response (above) and construct an **HS.Message.IHE.XDSb.RetrieveRequest** message that contains the document unique ID and repository unique ID (OID) for one or more of the documents listed in the query response.

   Send the XDS.b Retrieve Request to the XDS.b Document Consumer.

2. The document consumer transforms the message into an IHE "XDSb_RetrieveRequest" message using an internally-specified transform.

3. The document consumer then forwards the request to the XCA Initiating Gateway service. This service is specified in the XDSbRepositoryServiceName field. For XCA, this field should contain a Service Registry entry containing the URL of the XCA Initiating Gateway service in your production. If you are using only XDS.b, this field is normally left empty.

   The rest of the procedure takes place in a new session until the document consumer receives its response.

4. The XCA Initiating Gateway service forwards the request to the XCA Initiating Gateway process.
5. The request may contain documents from different Home Communities, possibly including the local community. The XCA Initiating Gateway process separates them out. The `<HomeCommunityId>` in the message is specified as an OID. In order to translate the OID into a URL, the following setup is required:
   - An OID registry entry of type "HomeCommunity" for each Home Community OID.
   - A Service Registry entry for each non-local Home Community that provides the URL of the XCA Responding Gateway actor in that community:
     - Each Service Registry entry must include the OID registry code in the **Home Community** field. This ties the OID and Service Registry entries together.
     - Each Service Registry entry must include `XCA.Retrieve` in the **Device Function** field. This indicates that it is the "XCA retrieve device" for the Home Community.
   - A Service Registry entry for the local Home Community that provides the URL of the XDS repository:
     - This Service Registry entry must include the OID registry code in the **Repository** field. This ties the OID and Service Registry entries together.
     - This Service Registry entry must include `XDSb.Retrieve` in the **Device Function** field. This indicates that it is the "XDS.b retrieve device" for the local Home Community.
6. The XCA Initiating Gateway process constructs an XCA_RetrieveRequest for each community.
7. The XCA Initiating Gateway process sends the requests to the XCA Initiating Gateway operation.
8. The XCA Initiating Gateway operation forwards the requests to the appropriate XCA Responding Gateway endpoints and returns their responses to the XCA Initiating Gateway process.
9. The XCA Initiating Gateway process combines the responses from the various Home Communities.
10. The XCA Initiating Gateway process constructs an XCA_RetrieveResponse message that contains a list of documents in the message body, and a set of MIME-encoded MTOM attachments, one for each document.
11. The XCA Initiating Gateway process returns the query response to the XCA Initiating Gateway service.
12. The XCA Initiating Gateway service returns the combined responses to the XDS.b Document Consumer in the original session in an XML message of the "XDSb_RetrieveResponse" variety.
13. The document consumer separates out the attachments and translates them.
14. The document consumer returns the list of documents and the MIME-encoded MTOM attachments in an XML message of the "XDSb_RetrieveResponse" variety with a <DocType> of "RetrieveDocumentSetResponse".

7.6 XCA Retrieve Components and Settings

Table 7–2: Components and Settings Used in XCA Retrieve

Component	Setting
Business Hosts	XCA Document Consumer: **HS.HC.IHE.XDSb.Consumer.Operations**
Business Hosts	XCA Initiating Gateway Service: **HS.IHE.XCA.InitiatingGateway.Services**
Business Hosts	XCA Initiating Gateway Process: **HS.IHE.XCA.InitiatingGateway.Process**

Component	Setting
Business Hosts	XCA Initiating Gateway Operation: **HS.IHE.XCA.InitiatingGateway.Operations**
Production Settings	XDSbRepositoryServiceName in the XCA Document Consumer • Points to the local XCA Initiating Gateway service.
Production Settings	XCAInitiatingGatewayOperation in the XCA Initiating Gateway process
Production Messages	**HS.Message.IHE.XDSb.RetrieveRequest**
Production Messages	**HS.Message.IHE.XDSb.QueryResponse**
Production Messages	**HS.Message.XMLMessage**: • XCA_RetrieveRequest • XCA_RetrieveResponse • XCA_QueryResponse • XDSb_RetrieveResponse • XDSb_RetrieveResponse (RetrieveDocumentSetResponse)
XSL Transformations	RetrieveRequestToXDSbRetrieve.xsl in XCA Document Consumer
XSL Transformations	Message-To-Metadata.xsl in XCA Document Consumer
Service Registry Entries	XCA.IG (or similar) • points to the local XCA Initiating Gateway service
Service Registry Entries	An XCA.Retrieve device entry for each foreign Home Community that points to the URL of the XCA Responding Gateway endpoint in that community. The OID registry code for the community should appear in the **HomeCommunity** field, and `XCA.Retrieve` should appear in the **Device Function** field.
Service Registry Entries	A Service Registry entry for the local Home Community that provides the URL of the XDS repository. This Service Registry entry must include the OID registry code in the **Repository** field. This ties the OID and Service Registry entries together. This Service Registry entry must include `XDSb.Retrieve` in the **Device Function** field. This indicates that it is the "XDS.b retrieve device" for the local Home Community.
OID Registry Entries	A HomeCommunity OID for the local and each foreign Home Community
OID Registry Entries	A repository OID for the local XDS repository
External IHE Actor Endpoints	XCA Responding Gateway(s) XDS Repository (in the local Home Community)

7.7 XCA Query and Retrieve Example

The method below sends a message that queries the document registry in each listed foreign Affinity Domain and then retrieves whatever documents are found. The user must enter `G` after the break command to continue.

```
/// XCA Query and retrieve///
ClassMethod XCAQuery()
{

  // Create an XDSb Query Request message
  s o=##class(HS.Message.IHE.XDSb.QueryRequest).%New()

  // Add the MPI ID, document status, type and creation date
  Do o.AddPatientId("100000001^^^&1.3.6.1.4.1.21367.2010.1.2.300&ISO")  //This is the format required
 by IHE

  Do o.AddStatusValues("Approved")

  Do o.AddCreationFrom("20110510102615-0400")

  Do o.AddDocumentType(1) // 1 is stable, 2 is on-demand, 3 is both

  Do o.AdditionalInfo.SetAt(1,"XCA")

  // Send the message to the test service (or directly to HS.IHE.XDSb.Consumer.Operations or
  // HS.HC.IHE.XDSb.Consumer.Operations)
  w ##class(HS.Test.Service).SendSync(o,.pr)

  Break

  /// XCA Retrieve ///

  // Assumes you are using the response from the previous query.

  // Create the XDSb Retrieve Request message
  Set obj=##class(HS.Message.IHE.XDSb.RetrieveRequest).%New()

  // Use the results of the query to populate the message
  Set obj.Documents=pr.Documents

  // Required only for HS.Test.Service to distinguish between XCA and XDS.b
  Do obj.AdditionalInfo.SetAt(1,"XCA")

  // Send the message to the test service (or directly to XCA Document Consumer)
  Write ##class(HS.Test.Service).SendSync(obj,.rr)

  Quit
}
```

8

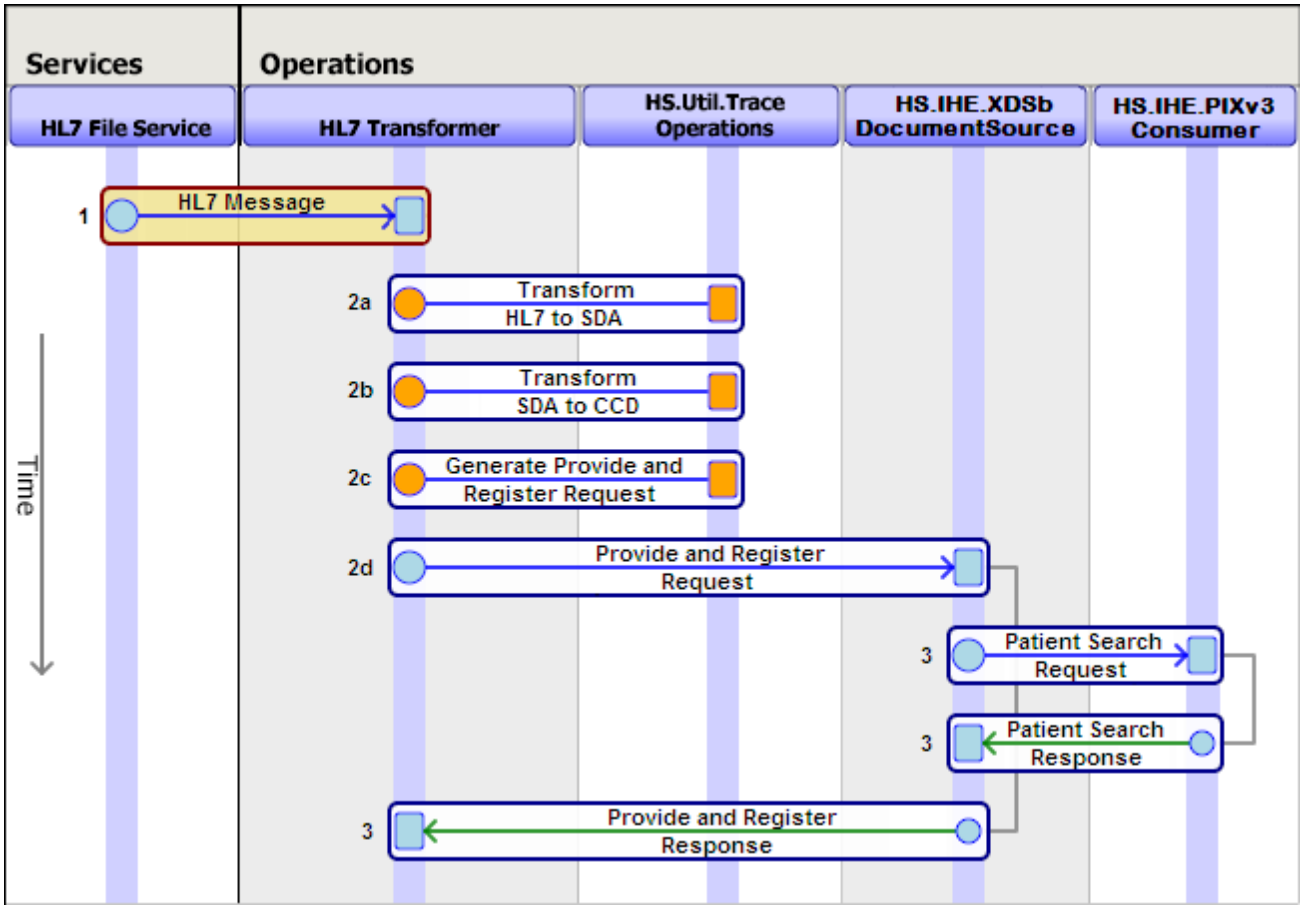
Generating a C-CDA Document from an HL7 Message

InterSystems products includes transformations that can accept an HL7 message, transform it into SDA (an internal representation), and then transform it into a C-CDA document. You can then provide and register the C-CDA to an XDS document repository.

8.1 HL7 to C-CDA Provide and Register Message Trace

The following diagram is an annotated message trace for an HL7 message transformed into a C-CDA followed by an XDS provide and register. The provide and register includes a PIX search to get the MPI ID from the MRN extracted from the C-CDA.

The test service shown in the diagram is a simple message router. Trace operations is a utility that makes intermediate processing steps visible in the trace. The numbers in the diagram match the steps in the procedure below.



8.2 HL7 to C-CDA Provide and Register Procedure

To set up InterSystems products to convert HL7 into C-CDA:

1. Create an HL7 input service (for example, an HL7 file service). Set up the HL7 input service to forward HL7 messages to some operation named in the TargetConfigNames setting.
2. Create the operation named in the previous step, and set it up to:
   a. transform the HL7 message into SDA3.
   b. transform the SDA into a C-CDA 1.1 or 2.1 document
   c. package the C-CDA into a Provide and Register request.
   d. send the Provide and Register request to the InterSystems Document Source.
3. The Document Source then follows the path laid out in the section "Provide and Register a CDA document".

8.3 HL7 to C-CDA Provide and Register Components and Settings

Table 8–1: Components and Settings Used in Transformation of HL7 to C-CDA followed by a Provide and Register

Component	Setting
Business Hosts	HL7 Service: EnsLib.HL7.Service.FileService (or similar)
Business Hosts	HL7 to C-CDA Transformer: Custom Business Operation
Business Hosts	Document Source: **HS.IHE.XDSb.DocumentSource.Operations**
Business Hosts	PIX Consumer: **HS.IHE.PIXv3.Consumer.Operations**
Production Settings	TargetConfigNames in the HL7 Service
Production Messages	**HS.Message.IHE.XDSb.ProvideAndRegisterRequest**
Production Messages	**HS.Message.XMLMessage**: • RegistryResponse
Production Messages	**HS.Message.PatientSearchRequest**
Production Messages	**HS.Message.PatientSearchResponse**
XSL Transformations	SDA-to-CCDAv21-CCD.xsl or other SDA to C-CDA transformation
Service Registry Entries	XDSb.Repository
Service Registry Entries	PIXv3.Manager
External IHE Actor Endpoints	XDS Document Repository
External IHE Actor Endpoints	PIX Manager

8.4 Example Transformer Business Operation to Generate a C-CDA Document from an HL7 Message

Below is a sample business operation that accepts an HL7 message and transforms it into a C-CDA document:

```
 Class Test.HL7Transformer Extends (Ens.BusinessOperation)
       [ Inheritance = right, ProcedureBlock ]
{
  Parameter INVOCATION = "Queue";

  /// XDSb source operations component
  Property XDSbSourceOperations As Ens.DataType.ConfigName
     [ InitialExpression = "HS.IHE.XDSb.DocumentSource.Operations"];
  Parameter SETTINGS As %String = "XDSbSourceOperations";

  XData MessageMap
  {
    <MapItems>
       <MapItem MessageType="EnsLib.HL7.Message">
         <Method>ProcessHL7Message</Method>
       </MapItem>
    </MapItems>
```

```
  }
  /// Process an inbound HL7 v2.5.1 message
  Method ProcessHL7Message(pRequest As EnsLib.HL7.Message,
                           Output pResponse As EnsLib.HL7.Message) As %Status
  {
    Try {
          // Convert the HL7 message to SDA3
          Set tSC = ##class(HS.Gateway.HL7.HL7ToSDA3).GetSDA(pRequest,.tSDA3Stream)
          Quit:$$$ISERR(tSC)

          // Transform the SDA3 to a C-CDA
          Set tTransformer = ##class(HS.Util.XSLTTransformer).%New()
          Set tSC= tTransformer.Transform(tSDA3Stream,"SDA3/SDA-to-CCDA-CCD.xsl",.tCDAStream)
          Quit:$$$ISERR(tSC)

          // Create a Provide and Register
          Set tRequest = ##class(HS.Message.IHE.XDSb.ProvideAndRegisterRequest).%New()
          Set tDocument = ##class(HS.Message.IHE.XDSb.Document).%New()
          Set tDocument.FormatCode=##class(HS.IHE.XDSb.Types.CodedValue).Create(
            "urn:hl7-org:sdwg:ccda-structuredBody:1.1","1.3.6.1.4.1.19376.1.2.3","Consolidated CDA R1.1
 Structured Body Document")
          Set tDocument.MimeType="text/xml"

          // Copy the C-CDA into the message
          Do tDocument.BodyCharacter.CopyFrom(tCDAStream)
          Do tRequest.Documents.Insert(tDocument)

          // Call the document source operation
          Set tSC = ..SendRequestSync(..XDSbSourceOperations,tRequest,.tResponse)
          Quit:$$$ISERR(tSC)
        } Catch ex {
                    Set tSC= ex.AsStatus()
                   }
          Quit tSC
  }
}
```

9

Generating a C-CDA Document by Querying an Internal Database

You can generate a C-CDA document using XSL transformations provided by InterSystems that transform SDA into C-CDA, for example, *installDir*\CSP\xslt\SDA3\SDA-to-CCDAv21-CCD.xsl. You must write code to transform the query result from your internal database into SDA.

See the chapter "SDA Documents" in *SDA: InterSystems Clinical Data Format* for more details on the structure of the SDA.

10

Receiving a C-CDA Document and Converting it to an Internal Format

If you receive a C-CDA document and wish to parse it and store its contents in an internal database, apply the provided transformation from C-CDA to SDA, for example, *HSF-home*\CSP\xslt\SDA3\CDA-to-SDA.xsl, and then write code to transform the SDA into your internal database format.

See the chapter "SDA Documents" in *SDA: InterSystems Clinical Data Format* for more details on the structure of the SDA.

11

Using Cross-Enterprise User Assertion (XUA)

Cross-Enterprise User Assertion (XUA) is supported in InterSystems products.

- To accept an inbound message, you must extend HS.HC.IHE.XUA.Processor and implement the **ProcessRequest()** method, where inbound user data is authenticated.
- To send a message, you must extend HS.HC.IHE.XUA.Creator and implement the following methods:
  - **FetchUserOrganization()** provides organization lookup
  - **FetchUser()** provides lookup of individuals

12

Internet User Authorization (IUA) Support

InterSystems products have built-in support for OAuth 2.0 that satisfies the requirements of the IHE Internet User Authorization (IUA) profile, which provides an authorization profile for HTTP RESTful transactions. For an introduction to the IUA profile, see the IHE Wiki.

All of the IUA actors and transactions are supported by InterSystems products. Descriptions of how to implement these actors and transactions are found in sections of the *Using OAuth 2.0 and OpenID Connect* guide. For specific instructions related to an actor or transaction, see the links below. Note that the *Using OAuth 2.0 and OpenID Connect* guide uses the term *access token*, while the IHE profile uses the term *authorization token*.

12.1 IUA Actors

InterSystems products can act as all three IUA actors:

- Authorization Client — Obtains an authorization token from the Authorization Server and attaches that token to a request to the Resource Server to prove that it is authorized to complete the transaction. For implementation details, see "Using an InterSystems IRIS Web Application as an OAuth 2.0 Client" in the *Using OAuth 2.0 and OpenID Connect* guide.
- Authorization Server — Confirms the Authorization Client's credentials and other information before issuing the client an authorization token it will use to complete a transaction with the Resource Server. For implementation details, see "Using InterSystems IRIS as an OAuth 2.0 Authorization Server" in the *Using OAuth 2.0 and OpenID Connect* guide.
- Resource Server — Accepts a HTTP RESTful transaction request as long as the request includes a valid authorization token. For implementation details, see "Using an InterSystems IRIS Web Application as an OAuth 2.0 Resource Server" in the *Using OAuth 2.0 and OpenID Connect* guide.

12.2 IUA Transactions

InterSystems products can successfully complete both IUA transactions: Get Authorization Token and Incorporate Authorization Token.

12.2.1 Get Authorization Token (ITI-71)

The Get Authorization Token transaction involves the Authorization Client and Authorization Server, both of which are supported by InterSystems products. The setup and process by which the Authorization Client requests and obtains an authorization token is described in the "Configuration Requirements" and "Obtaining Tokens" sections of "Using an InterSystems IRIS Web Application as an OAuth 2.0 Client". The setup and basic logic for handling a client's request for an authorization token and the granting of that token is described in "Using InterSystems IRIS as an OAuth 2.0 Authorization Server" in the *Using OAuth 2.0 and OpenID Connect* guide.

12.2.2 Incorporate Authorization Token (ITI-72)

Once an Authorization Client has successfully obtained an authorization token, the client must incorporate that token into the RESTful transaction request sent to the Resource Server. For details on how InterSystems products incorporate a token into a RESTful request, see "Adding an Access Token to an HTTP Request" in the *Using OAuth 2.0 and OpenID Connect* guide. You can also use InterSystems products as the Resource Server that is accepting the HTTP request. For guidance on how to build a Resource Server that extracts an authorization token from an incoming request and examines it, see "Code Requirements" in the *Using OAuth 2.0 and OpenID Connect* guide.

12.3 IUA Actor Options

According to the IUA profile, all actors are required to support the JSON Web Token format (JWT) for the authorization token. These actors can also support the SAML or OAuth Bearer Token formats. You can build your InterSystems Authorization Client and Resource Server to use any of the token formats, however InterSystems has not tested generating and handling a SAML token.

By default, when an InterSystems product is used as an Authorization Server, it generates OAuth Bearer Tokens. To switch to generating a JWT token:

1. Open the Management Portal and navigate to **System Administration** > **Security** > **OAuth 2.0** > **Server**.
2. Select the **Customization** tab.
3. In the **Generate token class** field, enter: `%OAuth2.Server.JWT`
4. Complete all other requirements for using JWTs as described in *Using OAuth 2.0 and OpenID Connect*.

13

Cross-Community Access for Imaging (XCA-I)

The Cross-Community Access for Imaging (XCA-I) IHE profile makes it possible for communities to share imaging. Once a community uses an XCA actor to retrieve an Imaging Manifest with a ITI-38 Cross Gateway Query, it can use the XCA-I profile to retrieve the image referenced by that Manifest from a XDS-I.b Imaging Document Source. The community requesting access to an image uses an *Initiating Imaging Gateway* actor to send the XCA-I request to the *Responding Imaging Gateway* actor of the community that includes the Imaging Document Source. Specifically, the Initiating Imaging Gateway uses the Cross Gateway Retrieve Imaging Document Set transaction (RAD-75) to request the image from the Responding Imaging Gateway.

InterSystems products provide built-in components to create both XCA-I actors (Initiating Imaging Gateway and Responding Imaging Gateway) and to register an existing Imaging Document Source so these XCA-I actors can find it. At this time, InterSystems products do not provide pre-built components for creating an XDS-I.b Imaging Document Source actor.

13.1 Initiating Imaging Gateway

When a community's Imaging Document Consumer actor needs to retrieve imaging from an XDS-I.b source, it sends a request to its local XCA-I Initiating Imaging Gateway actor using a Retrieve Imaging Document Set transaction (RAD-69). If the XDS-I.b source is located in a different community, the Initiating Imaging Gateway sends an XCA-I request message (RAD-75) to the Responding Imaging Gateway of the community that contains the imaging. If the XDS-I.b source is located in the same community as the Initiating Imaging Gateway, it retrieves the imaging directly without using a Responding Imaging Gateway.

Setting up the Initiating Imaging Gateway consists of creating an interoperability production that contains the following business hosts:

- `HS.IHE.XCAI.InitiatingGateway.Services`
- `HS.IHE.XCAI.InitiatingGateway.Process`
- `HS.IHE.XCAI.InitiatingGateway.Operations`

In addition, you need to define an Service Registry entry for the Responding Imaging Gateway of every external community that contains an XDS-I.b source with imaging that the Initiating Imaging Gateway might need. This Service Registry entry includes the OID of the responding community. For each responding community with an XDS-I.b source:

1. Open the Management Portal.

2. Navigate to **Health** > **IHE Configuration** > **OID Registry** > **Add/Edit OIDs**, and select **Add OID** to add the OID of the responding community. When defining the fields of the OID, be sure that you select **HomeCommunity** in the **Type** field.
3. Navigate to **Health** > **Service Registry** to create a new Service Name for the Responding Imaging Gateway. When defining the required fields of the Service Registry entry, be sure to:
   - Select the correct Code of the responding community in the **HomeCommunity** drop-down list. This is the Code for the community's OID that you defined in the previous step.
   - Specify **XCAI.Retrieve** in the **Device Function** field.

When the Initiating Imaging Gateway receives a message from a Imaging Document Consumer actor, it examines the `HomeCommunityId` and uses the following logic to find the correct XDS-I.b source:

- If the value of `HomeCommunityId` matches the home community of the Initiating Imaging Gateway, the local XDS-I.b source is identified by searching the Service Registry for the endpoint with **Device Function** = **XDSI.Retrieve** and a **Repository** that matches the message's `RepositoryUniqueId` OID.
- If the value of `HomeCommunityId` is not the same as the Initiating Imaging Gateway's community, then the Responding Imaging Gateway where the XCA-I message will be sent is identified by searching the Service Registry for an endpoint with **Device Function** = **XCAI.Retrieve** and a **HomeCommunity** that matches the `HomeCommunityId` OID.

13.2 Responding Imaging Gateway

Using an InterSystems product to configure a responding community to accept XCA-I requests and return imaging from an XDS-I.b source consists of creating an interoperability production for the Responding Imaging Gateway and defining an OID and Service Registry entry for the XDS-I.b source.

The interoperability production for the Responding Imaging Gateway must include the following business hosts:

- `HS.IHE.XCAI.RespondingGateway.Services`
- `HS.IHE.XCAI.RespondingGateway.Process`
- `HS.IHE.XCAI.RespondingGateway.Operations`

For details on defining the OID and Service Registry entry for the responding community's XDS-I.b source, see Registering XDS-I.b Sources.

13.3 Registering XDS-I.b Sources

Though InterSystems products do not provide built-in components designed to create an XDS-I.b source, they do provide the ability to register existing sources from which images can be retrieved by XCA-I actors. The XDS-I.b source can be in the same community as the Initiating Imaging Gateway or it can be part of the Responding Imaging Gateway's community, in which case the Responding Imaging Gateway will retrieve the image from the source when it receives an XCA-I request from another community.

Registering an XDS-I.b Imaging Document Source consists of defining an OID and Service Name entry as described in the following procedure:

1. Open the Management Portal.

2. Navigate to **Health** > **IHE Configuration** > **OID Registry** > **Add/Edit OIDs**, and select **Add OID** to add the OID of the Imaging Document Source. When defining the fields of the OID, be sure that you select **Repository** in the **Type** field.
3. Navigate to **Health** > **Service Registry** to create a new Service Name for the XDS-I.b source. When defining the required fields of the Service Registry entry, be sure to:
   - Select the correct Code of the XDS-I.b source in the **Repository** drop-down list. This is the Code for the XDS-I.b source that you created in the previous step.
   - Select **XDSI.Retrieve** from the **Device Function** drop-down list.

14

Registry Settings for IHE Communication

In addition to configuring your productions for IHE, you must create certain registry entries to facilitate IHE communication. Entries must be made in the:

- Service Registry
- OID Registry
- Configuration Registry

See the *Registry Guide for InterSystems IRIS for Health and Health Connect* for details on how to create registry entries.

14.1 Sample Service Registry Entries for IHE

PIXv3.Consumer

Web address of the PIX consumer operation on your system, where PIX queries should be routed.

`http://`*Your_Server*`:`*Port*`/csp/healthshare/`*Namespace*`/HS.IHE.PIXv3.Consumer.Operations.cls`

PIXv3.Manager

Web address of the PIX manager endpoint on another system that you communicate with, where PIX queries should be sent.

`http://`*Other_Server*`:`*Port*`/csp/healthshare/`*Namespace*`/HS.IHE.PIXv3.Manager.Services.cls`

PDQv3.Consumer

Web address of the PDQ consumer operation on your system, where PDQ queries should be routed.

`http://`*Your_Server*`:`*Port*`/csp/healthshare/`*Namespace*`/HS.IHE.PIXv3.Consumer.Operations.cls`

PDQv3.Supplier

Web address of the PDQ supplier endpoint on another system that you communicate with, where PDQ queries should be sent.

`http://`*Other_Server*`:`*Port*`/csp/healthshare/`*Namespace*`/HS.IHE.PDQv3.Supplier.Services.cls`

XDSb.Registry

Web address of the registry endpoint for your Affinity Domain, where XDS queries should be sent. There is only a single registry within a given Affinity Domain.

`http://`*Other_Server*`:`*Port*`/csp/healthshare/`*Namespace*`/HS.IHE.XDSb.Registry.Services.cls`

XDSb.Repository

Web address of the repository endpoint on another system that you communicate with, where Provide and Register requests should be sent. There may be more than one repository within an Affinity Domain, so you may have multiple repository entries in your Service Registry.

`http://`*Other_Server*`:`*Port*`/csp/healthshare/`*Namespace*`/HS.IHE.XDSb.Repository.Services.cls`

XCA.Query

Web address of the XCA responding gateway service on your system, where inbound XCA queries should be routed.

`http://`*Your_Server*`:`*Port*`/csp/healthshare/`*Namespace*`/HS.IHE.XCA.RespondingGateway.Services.cls`

XCA.Retrieve

Web address of the XCA responding gateway service on your system, where inbound XCA retrieve requests should be routed.

`http://`*Your_Server*`:`*Port*`/csp/healthshare/`*Namespace*`/HS.IHE.XCA.RespondingGateway.Services.cls`

XCA.InitiatingGateway

Web address of the XCA initiating gateway service on your system, where outbound XCA queries should be routed.

`http://`*Your_Server*`:`*Port*`/csp/healthshare/`*Namespace*`/HS.IHE.XCA.InitiatingGateway.Services.cls`

OtherSystem.XCPD.RespondingGateway

Web address of the XCPD responding gateway in a foreign system. There is a single responding gateway for each foreign Affinity Domain. A separate entry is required for each Affinity Domain that may hold XCA documents.

OtherSystem.XCA.Query

(Optional) Web address of the XCA responding gateway in a foreign system where XCA queries should be forwarded. Can be used in the XCAQueryServiceName setting in **HS.IHE.XCA.RespondingGateway.Operations** if XCA queries are handled by a different system.

OtherSystem.XCA.Retrieve

(Optional) Web address of the XCA responding gateway in a foreign system where XCA retrieves should be forwarded. Can be used in the XCARetrieveServiceName setting in **HS.IHE.XCA.RespondingGateway.Operations** if XCA retrieves are handled by a different system.

14.2 OID Registry Entries for IHE

The following is the minimum set of OIDs required for IHE communication:

- A Facility OID for each facility.

- An AssigningAuthority OID for the assigning authority associated with each facility (may be the same as the facility OID).
- A HomeCommunity OID for the home community.
- An AssigningAuthority OID for the home community (may be the same as the home community OID).
- HomeCommunity OIDs for other communities.
- A Repository OID for each XDS repository.
- A Device OID for the PIXv3Manager device.
- A Device OID for the PIXv3Consumer device.
- Facility and AssigningAuthority OIDs for external facilities and assigning authorities.

14.3 Configuration Registry Entries for IHE

A minimum configuration for IHE communication requires the `AffinityDomain` and `HomeCommunity` entries to be set in the Configuration Registry. These values should be concise, but descriptive so that your organization can be distinguished from other IHE participant organizations. *Do not* supply generic values under any circumstances.

15

Using the IHE Test Utility

The IHE test utility is a tool that assists in configuring and testing IHE scenarios. The IHE test utility is available from the **Other Management** menu in a registry production, or from the production menu in a foundation production. It can generate and send test data for PIX, PDQ, and XDS.b transactions, taking the place of an external testing utility like SoapUI.

15.1 Configuring your Productions to Use the Test Utility

In order to use the test utility, your Foundation production must have the **HS.Test.Service** enabled. This is the business service to which the utility will send SOAP messages. It includes a series of settings that point to the relevant business actors that perform the IHE transactions.

If you have installed one or more FHIR IHE profiles, your list of additional settings will include the relevant FHIR IHE business hosts.

To test XDS.b, your Foundation production must also have **HS.IHE.XDSb.DocumentSource.Operations** enabled.

To test metadata update, your Foundation production must also have **HS.IHE.XDSb.Administrator.Operations** enabled.

To enhance your ability to track down any testing issues, you may wish to enable TraceOperations in your PIX, PDQ, and XDS.b business operations. Just remember to turn off trace operations before going into a production environment.

15.2 IHE Test Utility Main Menu

The IHE test utility uses a tabbed menu with the following sections:

- Main
- PIX and PDQ
- XDSB
- DSUB
- XCPD
- Any FHIR IHE profiles that you installed

Within each tab are menu items that bring up a particular page, for example the **Configuration** page.

15.2.1 Configuring the IHE Test Utility

For a single user on an instance, no configuration is required other than enabling the **HS.Test.Service** business service in your Foundation production. Your production must be running. If either of these conditions is not met, the utility issues a warning when you open it.

The utility uses three unique values during its testing: **Trace configuration name**, **Trace configuration port**, and **SAML username**. The utility automatically assigns default values for these fields; you do not need to change these default values unless multiple users are accessing the test utility. In this case, each user must enter unique values and click **Save** before using the utility. If multiple users are using the test utility, keep the following in mind when choosing unique values:

- **Trace configuration name** is any unique string. Default is `IHE.Test`*username*.
- **Trace configuration port** is a numeric value that does not conflict with other ports in use. The utility does not check whether the port number is already being used. Default is 54321.
- In many cases, **SAML username** can be any unique string. However, if the utility is sending messages to parts of HealthShare that validate the username against a registry, you must specify a valid SAML username. For example, if HealthShare is using the username for Consent processing, it must be valid.

15.2.1.1 Enabling Logging in the Test Utility

Optionally, you can select **Enable logging** to turn on logging by default for your transactions. You can also enable or disable logging for transactions on an individual basis. When logging is enabled, a log button appears after you run the transaction. It is labelled with an incrementing log number. The log shows the SOAP message sent to the test service, and the response received. You can also view the full set of logs across all of your transactions from the **Trace** option on the **Main** menu tab.

15.2.2 Viewing Endpoints in the IHE Test Utility

Click **EndPoints** on the **Main** menu tab to bring up a view into the Service Registry. Here you can view and edit the endpoints that the IHE test utility will work with.

15.2.3 Using the History Page in the IHE Test Utility

Click **History** on the **Main** menu tab to bring up a history of the transactions that you have previously performed in the test utility. This is useful in several cases:

- when you need to provide an identifier, as you can copy it from a search response in order to paste it into a new transaction.
- when you need to rerun a transaction, for example after tweaking your production settings after an unsuccessful test.
- when you need to view the logging or session information from a previous transaction.

15.2.4 Submitting a SOAP Request with the Test Utility

To submit a SOAP request outside the scope of the provided transactions, or to submit a request to a different location, the test utility provides a generic **Web Service Submit** page. To submit a SOAP request:

1. Select **WS Submit** from the **Main** menu tab.
2. Optionally select an **EndPoint** to populate the **Host**, **Port**, and **URL** fields. The endpoint list is populated from the Service Registry.
3. Alternatively, populate the **Host**, **Port**, and **URL** fields manually.

4. Enter an **SSL Configuration** if appropriate.
5. Select an **Action** from the dropdown. The dropdown is populated with the IHE transactions that InterSystems supports. This populates the second **Action** field with the IHE urn for the transaction.
6. Type or paste the SOAP request in the **Request** field.
7. Click **Send**.
8. To see the response, click on the **Response** tab.
9. To view the request or response as an XML tree, click **Show XML**.

15.3 Testing a PIX Add Transaction

To test adding a patient to your system via the PIX add transaction:

1. Click the **PIX and PDQ** menu tab.
2. Click **Patient Add**.
3. Optionally **Enable logging**.
4. Select whether you wish to use PIX **v2** or PIX **v3**. This preselects the **EndPoint** as PIXv2.Manager or PIXv3.Manager.
5. Enter a set of demographics for the patient. A minimum set of demographics should include:
   - **First Name**
   - **Last Name**
   - **Gender**
   - **Date of Birth**
   - **Assigning Authority** — select an assigning authority from the dropdown which is populated by assigning authority registry entries that have an OID associated with them.
   - **AssigningAuthority OID** — this is populated automatically when you select an assigning authority.
   - **Facility** — select a facility from the dropdown which is populated by the facility registry. When you enter an assigning authority, if there is a facility with the same code, the utility selects this facility by default.
   - **MRN** — the test utility automatically suggests an incremented MRN when you select an assigning authority. Either accept the suggested MRN or enter your own.
6. Optionally change the **EndPoint** for the transaction. The **EndPoint** dropdown is populated from the Service Registry. The **EndPoint** field is preselected by the **v2** and **v3** buttons at the top of the page.
7. Click **Submit Request**.

The test utility returns a string which includes the patient's name and the home community OID.

Click the **Session** button to view a visual trace of the test utility session. Click **Next Session** to see the details of the actual PIX Add. Click **Next Session** three more times to view additional processing that occurs on a PIX Add. You may wish to enable TraceOperations in **HS.IHE.PIXv3.Source.Operations**, **HS.IHE.PIX.Manager.Process**, and **HS.Hub.MPI.Manager** in order enhance your ability to track down any issues that arise during testing.

15.4 Testing a PIX Search

In a PIX search, you provide a local patient identifier (MRN) and assigning authority and you receive the universal identifier (MPIID) in the response.

To test searching for a patient via PIX:

1. Click the **PIX and PDQ** menu tab.
2. Click **Patient Search (PIX)**.
3. Optionally **Enable logging**.
4. Select whether you wish to use PIX **v2** or PIX **v3**. This preselects the **EndPoint** as PIXv2.Manager or PIXv3.Manager.
5. Select an **AssigningAuthority** from the dropdown. The OIDs are shown next to the codes. The dropdown is populated by assigning authority registry entries that have an OID associated with them.
6. Enter the local patient identifier (**MRN**).
7. Optionally change the **EndPoint** for the transaction. The **EndPoint** dropdown is populated from the Service Registry. The **EndPoint** field is preselected by the **v2** and **v3** buttons at the top of the page. The typical choice is PIXv2.Manager or PIXv3.Manager.
8. Click **Generate PIX Query Request**.

The test utility returns a string which includes:

- a timestamp
- the patient's name
- the MPIID
- the home community OID

Click the **Session** button to view a visual trace of the test utility session. Click **Next Session** to see the details of the actual search. You may wish to enable TraceOperations in **HS.IHE.PIXv3.Consumer.Operations** and **HS.IHE.PIX.Manager.Process** in order enhance your ability to track down any issues that arise during testing.

15.5 Testing a PDQ Search

In a PDQ search, you provide demographics and you receive the universal identifier (MPIID) and a list of local MRNs and assigning authorities in the response.

To test searching for a patient via PDQ:

1. Click the **PIX and PDQ** menu tab.
2. Click **Patient Search (PDQ)**.
3. Optionally **Enable logging**.
4. Select whether you wish to use PDQ **v2** or PDQ **v3.** This preselects the **EndPoint** as PDQv2.Supplier or PDQv3.Supplier.
5. Enter whatever demographics that you have for the patient.

6. Optionally change the **EndPoint** for the transaction. The **EndPoint** dropdown is populated from the Service Registry. The **EndPoint** field is preselected by the **v2** and **v3** buttons at the top of the page. The typical choice is PDQv2.Supplier or PDQv3.Supplier.
7. Click **Submit Request**.

The test utility returns a string which includes:

- a timestamp
- the patient's name
- the MPIID
- the home community OID
- a list of local MRNs and assigning authorities

Click the **Session** button to view a visual trace of the test utility session. Click **Next Session** to see the details of the actual search. You may wish to enable TraceOperations in **HS.IHE.PDQv3.Consumer.Operations** and **HS.IHE.PDQ.Supplier.Process** in order enhance your ability to track down any issues that arise during testing.

15.6 Testing a PIX Merge Transaction

In a PIX merge, you provide demographics, an assigning authority and facility, a surviving MRN, and a prior MRN. The system should merge the prior MRN into the surviving MRN and return a success or failure message in response.

To test PIX merge:

1. Click the **PIX and PDQ** menu tab.
2. Click **Patient Merge**.
3. Optionally **Enable logging**.
4. Select whether you wish to use PDQ **v2** or PDQ **v3**. This preselects the **EndPoint** as PIXv2.Manager or PIXv3.Manager.
5. Enter the patient name.
6. Select an **AssigningAuthority** from the dropdown which is populated by assigning authority registry entries that have an OID associated with them. The **AssigningAuthority OID** is populated automatically when you select an assigning authority.
7. Select a **Facility** from the dropdown which is populated by the facility registry. When you enter an assigning authority, if there is a facility with the same code, the utility selects this facility by default.
8. Enter the surviving **MRN**.
9. Enter the **Prior MRN**.
10. Optionally change the **EndPoint** for the transaction. The **EndPoint** dropdown is populated from the Service Registry. The **EndPoint** field is preselected by the **v2** and **v3** buttons at the top of the page. The typical choice is PIXv2.Manager or PIXv3.Manager.
11. Click **Submit PIX Merge Request**.

The test utility returns a success or failure message.

Click the **Session** button to view a visual trace of the test utility session. Click **Next Session** to see the details of the actual PIX Merge. You may wish to enable TraceOperations in **HS.IHE.PIXv3.Source.Operations**, **HS.IHE.PIX.Manager.Process** and **HS.Hub.MPI.Manager** in order enhance your ability to track down any issues that arise during testing.

15.7 Testing an XDS.b Provide and Register Transaction

An XDS.b provide and register (PnR) involves submitting:

- a document to a repository
- the document metadata to a registry

In the test utility, this is a two-part transaction: first identify the document, then submit it.

Note: In addition to the **HS.Test.Service**, your Foundation production must have the **HS.IHE.XDSb.DocumentSource.Operations** enabled in order to test XDS.b provide and register. This component builds the ProvideAndRegister transaction by extracting the relevant data from a CDA stream. It is only used in a testing context.

To test an XDS.b provide and register:

1. First perform a PIX or PDQ search for the patient as described in the previous sections. This will save you a lot of time and effort since the test utility remembers the results and offers to fill in certain fields for you.
2. Click the **XDSb** menu tab.
3. Click **Provide & Register**. This takes you to the document source page:
4. Choose a data source. The options are:
   - **Upload Local CDA file** — Select **Choose File**. Click the **Browse** button and select a C-CDA file on your local disk.
   - **Use sample xdata block CDA** — Select **Use sample xdata block CDA** to generate a CDA document from an xdata block. A sample is provided.

     You can create an alternate CDA document in your own class if you like. Use the following specification to enter the location of your xdata block in the **Use sample xdata block CDA** field: `xdata://`*classname*`:`*xdatablockname*.
   - **Use sample HL7** — Select **Use sample HL7** to generate HL7 from an xdata block. A sample is provided, or you can create your own.

     You can create an alternate HL7 document in your own class if like. Use the following specification to enter the location of your xdata block in the **Use sample xdata block HL7** field: `xdata://`*classname*`:`*xdataBlockName*.
   - **Upload Local (non) CDA file** — Select **Choose File**. Click the **Browse** button and select a file not of CDA format on your local disk.
5. Click **Upload/generate document**. This takes you to the document submit page:
6. Optionally **Enable logging**.
7. If you have previously performed a PIX or PDQ search, you can select a patient from the **Select Patient** dropdown. This will also populate the **Full Patient ID** field with the MPIID of the form *MPIID*`^^^&`*homeCommuniyOID*`&ISO`.
8. If you did not select a patient, enter the MPIID in the **Full Patient ID** field of the form *MRN*`^^^&`*AssigningAuthority*`&ISO`. If this an HL7 transaction from an xdata block, this field should already be populated.
9. If you did not specify a **Source Patient ID** on the previous page, enter it here of the form *MPIID*`^^^&`*homeCommuniyOID*`&ISO`.

10. Optionally select non-default values for the following types of XDS.b metadata:
    - **Format Code**
    - **Confidentiality Code**
    - **HealthcareFacilityType Code**
    - **Practice Setting Code**
    - **Type Code**
    - **Class Code**

    The values in the dropdowns come from the Coded Entry Registry, which is found under the **IHE Configuration** menu in the Registry namespace.
11. Select the endpoint for the generated provide and register transaction, typically your XDS.b repository.
12. Click **Generate XDSb PnR Request**.

This process kicks off the following steps:

1. The test utility sends a SOAP message containing the document and metadata to **HS.Test.Service** at the Foundation production.
2. **HS.Test.Service** forwards the document and metadata to **HS.IHE.XDSb.DocumentSource.Operations**.
3. **HS.IHE.XDSb.DocumentSource.Operations** generates a provide and register request and forwards it to the repository as a SOAP message.
4. The repository stores the document and sends an XDS.b register request to the registry.
5. **HS.Test.Service** returns a success or failure message.

Click the **Session** button to view a visual trace of the test utility session. Click **Next Session** to see the details of the Register request returned from the repository. You may wish to enable TraceOperations in **HS.IHE.XDSb.DocumentSource.Operations** and **HS.HC.IHE.XDSb.Registry.Operations** to enhance your ability to track down any issues that arise during testing. You can view the message trace of the repository side of the provide and register by opening the Interoperability message viewer on your repository production.

15.8 Testing an XDS.b Query

An XDS.b query transaction allows you to search the registry for documents that are related to specific patient. You can filter an XDS.b query by document type and document status. The query returns a list of document unique IDs and the name of the repository where they are stored.

Note: In order to test an XDS.b query, your Foundation production must contain the **HS.IHE.XDSb.Consumer.Operations** or **HS..HC.IHE.XDSb.Consumer.Operations** business host as well as the **HS.Test.Service**.

To test XDS.b query:

1. First perform a PIX or PDQ search for the patient as described in the previous sections. This will save you a lot of time and effort since the test utility remembers the results and offers to fill in certain fields for you.
2. Click the **XDSb** menu tab.
3. Click **Query**.

4. Optionally **Enable logging**.
5. If you already did a search for the patient, select the patient from the **Select Patient** dropdown. This will fill in the **Full Patient ID**.
6. If you did not conduct a search for the patient, enter the **Full Patient ID** (MPIID) of the form *MRN*^^^&*AssigningAuthority*&ISO.
7. If you are looking for a specific document, enter the **Document Unique ID**.
8. In the **Document Status** field, select whether you want only Approved documents, only Deprecated documents or both.
9. Optionally select **Search for Stable documents** or **Search for OnDemand documents**. The default is to check for both types if no checkbox is selected.
10. Select the **EndPoint** for your query, typically your document registry, XDSb.Registry.
11. Click Submit Request.

The query returns a list of document unique IDs and the name of the repository where they are stored.

Click the **Session** button to view a visual trace of the test utility session. Click **Next Session** to see the details of the XDS.b query. You may wish to enable TraceOperations in **HS.IHE.XDSb.Consumer.Operations** (or **HS.HC.IHE.XDSb.Consumer.Operations**) and **HS.HC.IHE.XDSb.Registry.Operations** to enhance your ability to track down any issues that arise during testing.

15.9 Testing an XDS.b Retrieve

You can retrieve a document that you queried for using the IHE test utility. While the IHE XDS.b Retrieve Document Set transaction allows you to retrieve multiple documents in a single transaction, the test utility supports the retrieval of only a single document for testing purposes.

Note: In order to test an XDS.b retrieve, your Foundation production must contain the **HS.IHE.XDSb.Consumer.Operations** or **HS.HC.IHE.XDSb.Consumer.Operations** business host as well as the **HS.Test.Service**.

To retrieve an XDS.b document:

1. Query for a patient's documents as described above.
2. Click the **XDSb** menu tab.
3. Click **Retrieve**.
4. Optionally **Enable Logging**.
5. In the **Select Document for Retrieval** field, select a patient and document unique ID from the list returned by your previous query. This fills in the **Repository Unique Id** and **Document Unique Id** fields and selects the **EndPoint** associated with the Repository Unique Id OID.
6. Enter the **Home Community ID** (optional if it is the same community).
7. Click **Submit Retrieve Request**.
8. To view the document, click the **View Document** button that appears after the transaction succeeds.

Click the **Session** button to view a visual trace of the document retrieve session. You may wish to enable TraceOperations in **HS.IHE.XDSb.Consumer.Operations** or **HS.HC.IHE.XDSb.Consumer.Operations** to enhance your ability to track

down any issues that arise during testing. You can view the message trace of the repository side of the retrieve transaction by opening the message viewer on your repository production.

16

Auditing

You can audit IHE processes and send audit messages to an ATNA repository.

16.1 Basic Auditing

The **HS.HC.Audit.ConsolidationService** is a business service that batches audit events and periodically updates the InterSystems audit logs. To set up auditing, enable this business service in your production.

16.2 ATNA Auditing

To add ATNA auditing:

1. Add one of the following business operations to your production:
   - **HS.IHE.ATNA.SecureApplication.TCP.Operations**
   - **HS.IHE.ATNA.SecureApplication.UDP.Operations**
2. Configure the ATNA operation to communicate with your ATNA repository via TCP or UDP as appropriate.
3. Set AuditAlertOperations in the **HS.HC.Audit.ConsolidationService** to the name of the business operation you installed in step 1.
4. Optionally edit the contents of the ExcludeUsers field in the **HS.HC.Audit.ConsolidationService** to control which events are audited.