InterSystems HEALTHSHARE™

HealthShare Foundation Basics

Version 2012.2
31 August 2012

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

HealthShare Foundation Basics
HealthShare Foundation Version 2012.2 31 August 2012

This book was assembled and formatted in Adobe Page Description Format (PDF) using tools and information from the following sources: Sun Microsystems, RenderX, Inc., Adobe Systems, and the World Wide Web Consortium at www.w3c.org. The primary document development tools were special-purpose XML-processing applications built by InterSystems using Caché and Java.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Customer Support

Tel:	+1 617 621-0700
Fax:	+1 617 374-9391
Email:	support@InterSystems.com

Table of Contents

List of Tables

About This Book

This book describes how to use HealthShare Foundation. It is divided into the following chapters:

- "Introduction to HealthShare Foundation" describes what HealthShare Foundation is and lists the IHE profiles and actors supported by the product.
- "HealthShare Foundation Use Cases" describes the use cases that HealthShare Foundation can solve.
- "Registry Settings for IHE Communication" describes how to configure the HealthShare Foundation registries to enable communication with other systems.
- "SDA Documents" describes the XML format used in HealthShare Foundation to represent patient records.
- "Auditing" describes how to configure auditing in HealthShare Foundation.

This book also contains a complete table of contents.

For more information on HealthShare Foundation, see the following books:

- *InterSystems Supported Platforms* lists the platforms on which HealthShare Foundation is supported.
- *HealthShare Foundation Administration Guide* describes how to administer HealthShare.
- *HealthShare Foundation Release Notes* lists new features in the product.
- *HealthShare Foundation Installation and Migration Guide* describes how to install or upgrade HealthShare Foundation. (Not yet completed.)

For general information, see *Using InterSystems Documentation.*

1

Introduction to HealthShare Foundation

Integrating the Healthcare Enterprise (IHE) is an emerging international standard for communication between healthcare entities. IHE provides a common standard for healthcare applications to interoperate. HealthShare Foundation provides Ensemble-based tools that perform IHE transactions with other systems. HealthShare Foundation supports a variety of IHE profiles and actors.

IHE transactions are based on Integration Profiles that use a complex set of XML messages whose structure is laid out in Technical Framework documents that run hundreds of pages long. HealthShare foundation allows you to perform IHE queries and send and receive IHE documents simply by constructing Ensemble messages, without requiring in-depth knowledge of the underlying IHE technology in use.

HealthShare Foundation includes:

- a set of Ensemble business hosts that you can add to a production to perform IHE transactions
- a trace business operation that can help you diagnose problems by reporting the status of a transaction at each stage of the transaction
- a set of Ensemble messages for initiating transactions and receiving results
- a set of XSL transformations that
  - translate Ensemble messages into IHE messages
  - translate IHE messages into Ensemble messages
  - construct IHE messages from other IHE messages
  - construct CCD documents from your data (via SDA, a documented, internal XML representation of patient data)
  - construct CCD documents from HL7
  - parse CCD documents into SDA, so you can integrate received CCD data into your database
- a utility (Caché class) that can set up a sample production for you
- a Test service, which is a simple Ensemble message router
- a set of registries to store:
  - assigning authorities
  - OIDs (of assigning authorities, home communities, coding systems, repositories, devices, facilities, etc.)
  - configuration options
  - internet addresses of services
  - SAML credentials

1.1 IHE Support in HealthShare Foundation

HealthShare Foundation supports the following IHE profiles and actors:

Profile	Description	Supported Actors	Supported Transactions	ITI #
ATNA	Audit Trail and Node Authentication	Secure Application	Record Audit Event	ITI-20
XCA	Cross-Community Access	Initiating Gateway	Retrieve Document Set	ITI-43
			Cross Gateway Retrieve	ITI-39
XCPD	Cross-Community Patient Discovery	Initiating Gateway	Patient Discovery Request	ITI-55
XDS.b	Cross-Enterprise Document Sharing, version b	Document Consumer	Registry Stored Query	ITI-18
			Retrieve Document Set	ITI-43
		Document Source	Provide and Register Document Set-b	ITI-41
PDQv3	Patient Demographics Query for HL7 version 3	Consumer	Patient Demographics Query	ITI-47
PIXv3	Patient Identifier Cross Reference for HL7 version 3	Consumer	PIX Query	ITI-45
		Patient Identity Source	Patient Identity Feed	ITI-44

InterSystems' official IHE integration statement for the full line of HealthShare products can be found online at www.intersystems.com/healthshare/ihe.

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

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

2

IHE Use Cases in HealthShare Foundation

This chapter details how HealthShare Foundation handles several IHE transaction scenarios:

- 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
- XCA
  - Query the registry in another Affinity Domain and retrieve a document from a repository
- CCD
  - Generate a CCD document from an HL7 message and provide and register it to a repository.
  - Generate a CCD document by querying an internal database
  - Receive a CCD document (via a registry or point-to-point) and convert it to an internal format

Each use case is described and the 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.

2.1 Performing PIX and PDQ Queries Against an EMPI

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

- Deterministic queries use the PIXv3 profile.
- Probabilistic queries use the PDQv3 profile.

2.1.1 Perform a Deterministic Query Against an EMPI (PIXv3)

HealthShare Foundation supports 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.

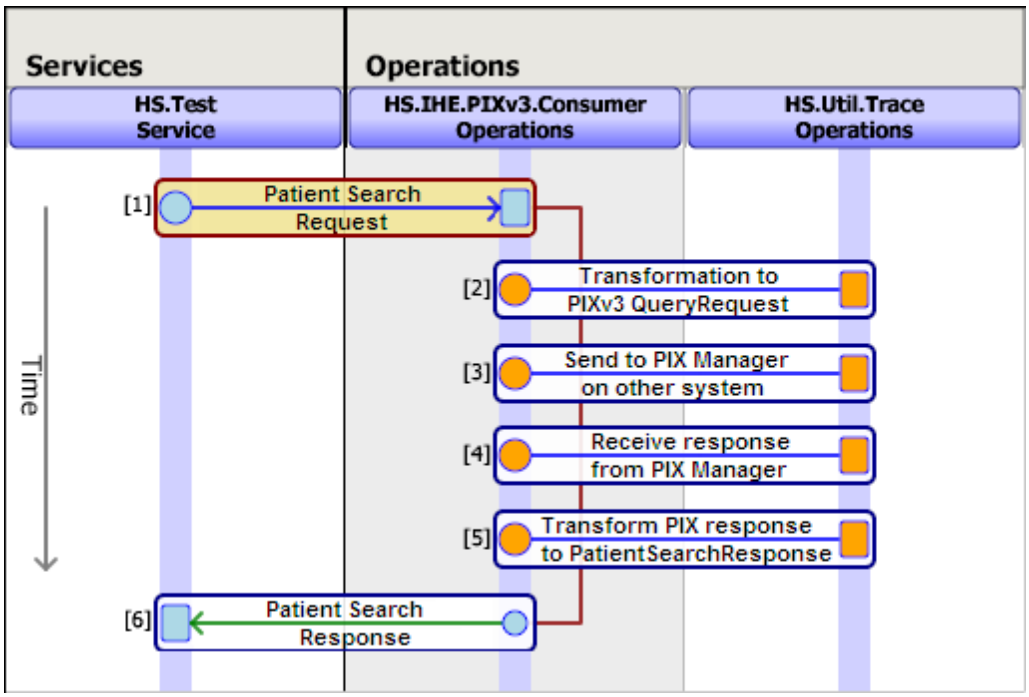
This section includes:

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

2.1.1.1 PIX Query Message Trace

Below 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.



2.1.1.2 PIX Query Procedure

1. You provide a **Patient Search Request** message containing an MRN and assigning authority to the HealthShare PIX Consumer.
2. The HealthShare PIX Consumer transforms the message into an IHE "PIXv3_QueryRequest" using the transformation specified in the TransformPatientSearchToPIX setting.
3. The HealthShare PIX Consumer 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 HealthShare PIX Consumer transforms the response into a **Patient Search Response** message using the transformation specified in the TransformPIXToPatientSearch.
6. The HealthShare 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.

2.1.1.3 PIX Query Components and Settings

Table 2–1: Components and Settings Used in a PIX Query

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

2.1.1.4 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
 }
```

2.1.2 Perform a Probabilistic Query Against an EMPI (PDQv3)

HealthShare Foundation supports 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:

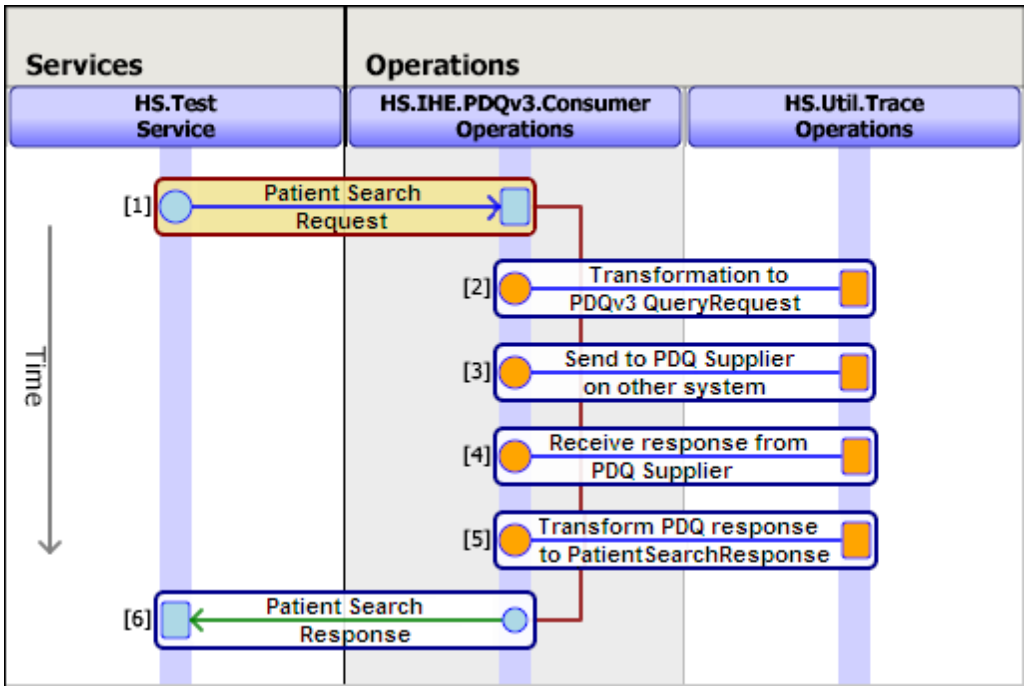
- PDQ query message trace
- PDQ query procedure
- PDQ query components

- PDQ query example

2.1.2.1 PDQ Query Message Trace

Below is an annotated PDQ 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.



2.1.2.2 PDQ Query Procedure

1. You provide a **Patient Search Request** message containing partial demographics to the HealthShare PDQ Consumer.
2. The HealthShare PDQ Consumer transforms the message into an IHE "PDQv3_QueryRequest" using the transformation specified in the TransformPatientSearchToPDQ setting.
3. The HealthShare 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 HealthShare PDQ Consumer transforms the response into a **Patient Search Response** message using the transformation specified in the TransformPDQToPatientSearch setting.
6. The HealthShare 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.

2.1.2.3 PDQ Query Components and Settings

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

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

2.1.2.4 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
 }
```

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

HealthShare Foundation can add a patient or update the demographic data held for a patient in an external EMPI via an IHE “PIXv3_PatientAddRequest” transaction.

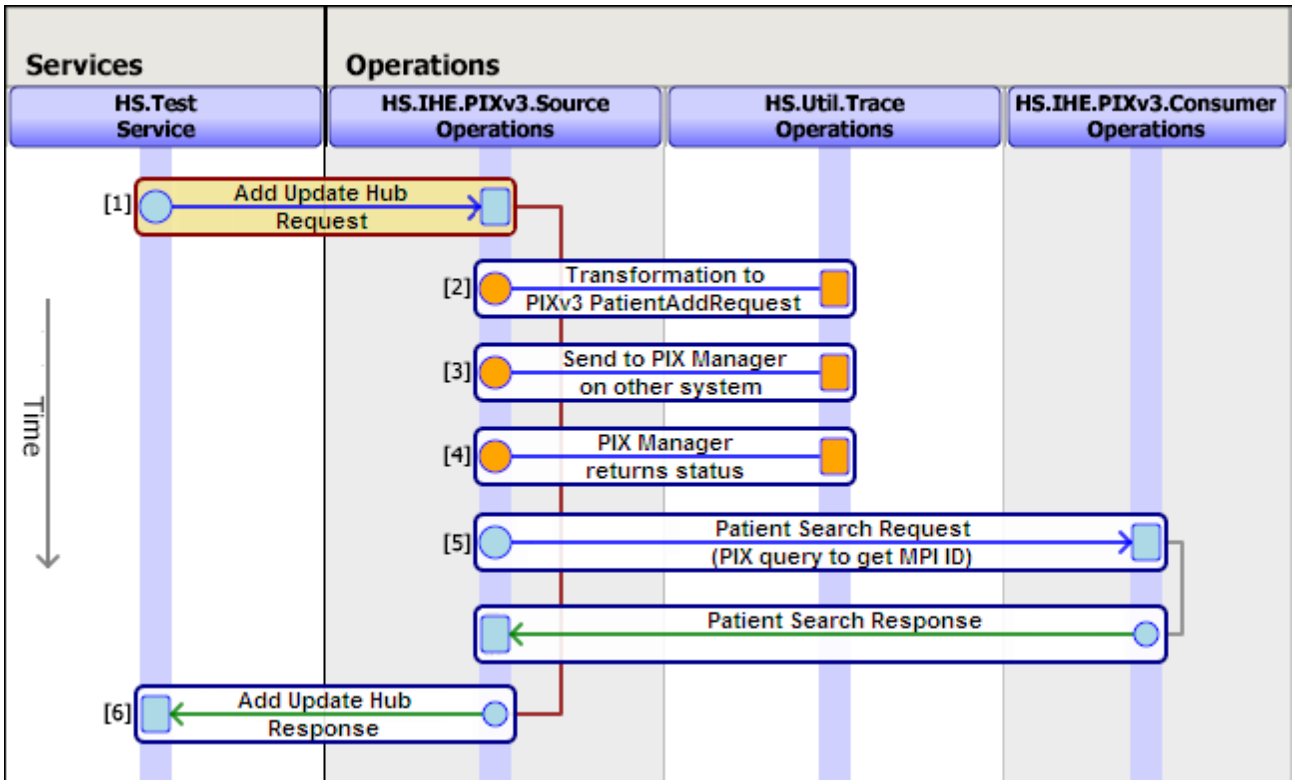
This section includes:

- PIX add message trace
- PIX add procedure
- PIX add components
- PIX add example

2.2.1 PIX Add Message Trace

Below 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.



2.2.2 PIX Add Procedure

1. You provide an **Add Update Hub Request** message that contains the necessary demographic data to the HealthShare PIX Source.
2. The HealthShare PIX Source transforms the message into an IHE "PIXv3_PatientAddRequest" using the transformation specified in the TransformAddUpdateHubToPIX setting.
3. The HealthShare 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 HealthShare 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 HealthShare 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 HealthShare PIX Source returns null.

2.2.3 PIX Add Components and Settings

Table 2–3: Components and Settings Used in a PIX Add

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

2.2.4 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")

   // 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
  }
```

2.3 Querying an XDS Registry and Retrieving a Document from a Repository

HealthShare Foundation 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.

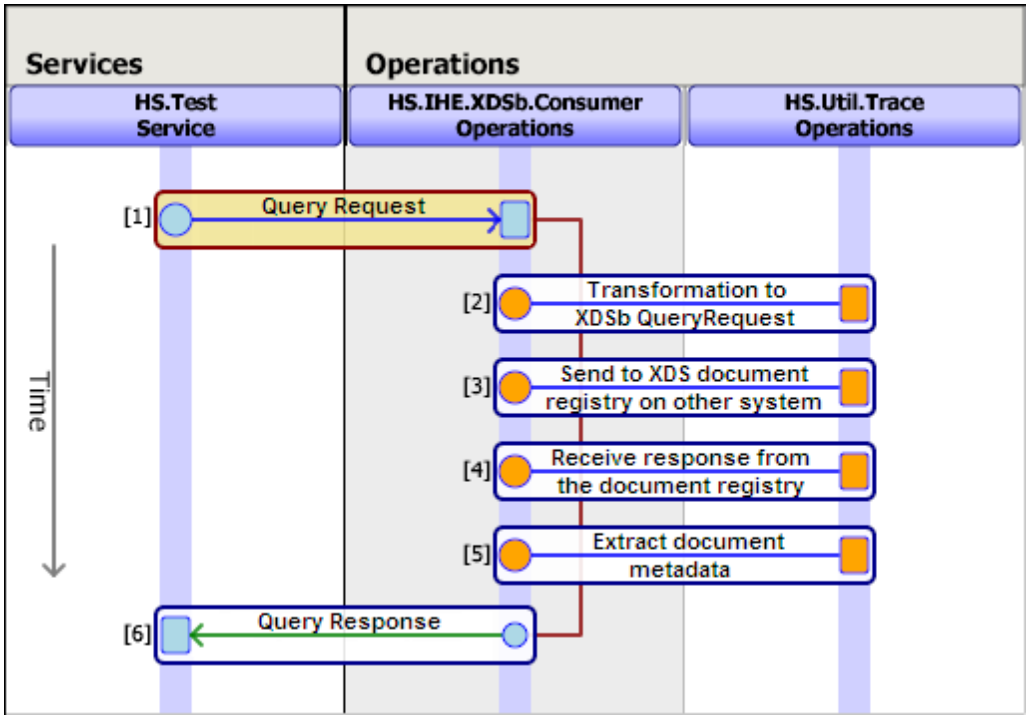
This section includes:

- XDS Query:
  - XDS query message trace
  - XDS query procedure
  - XDS query components
- XDS Retrieve:
  - XDS retrieve message trace
  - XDS retrieve procedure
  - XDS retrieve components
- Example XDS query and retrieve

2.3.1 XDS Query Message Trace

Below 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.



2.3.2 XDS Query Procedure

1. You provide an **XDS.b Query Request** message that contains an MPI ID and assigning authority to the HealthShare 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 HealthShare Document Consumer transforms the Ensemble message into an IHE "XDSb_QueryRequest" message using an internally-specified transform.
3. The HealthShare 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 HealthShare 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 HealthShare Document Consumer returns the **XDS.b Query Response** message, which contains both the original XDS document registry response and the extracted metadata.

2.3.3 XDS Query Components and Settings

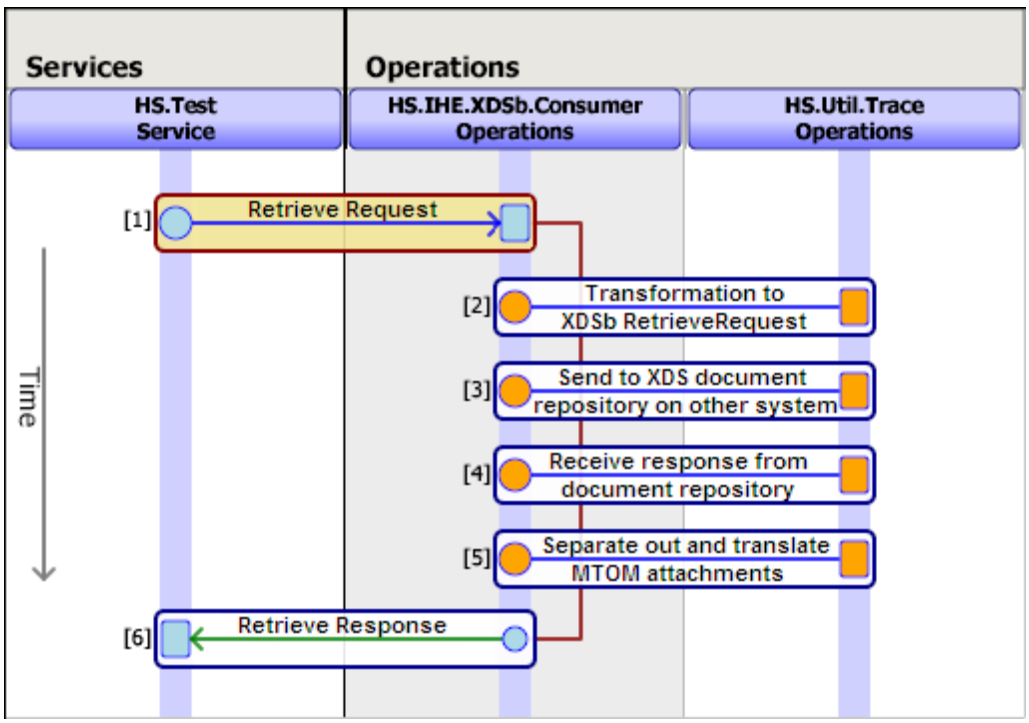
Table 2–4: Components and Settings Used in XDS Query

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

2.3.4 XDS Retrieve Message Trace

Below 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.



2.3.5 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 HealthShare Document Consumer.

2. The HealthShare Document Consumer transforms the Ensemble message into an IHE "XDSb_RetrieveRequest" message using an internally-specified transform.
3. The HealthShare 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 HealthShare 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 HealthShare Document Consumer separates out the attachments and translates them.
6. The HealthShare Document Consumer returns the MTOM attachments in an XML message of the "RetrieveDocumentSetResponse" variety.

2.3.6 XDS Retrieve Components and Settings

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

Business Hosts	Document Consumer: **HS.IHE.XDSb.Consumer.Operations**
Production Settings	XDSbRepositoryServiceName in the Document Consumer • For testing purposes only
Ensemble Messages	**HS.Message.IHE.XDSb.RetrieveRequest**
	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

2.3.7 XDS Query and Retrieve Example

Below is a sample XDS query and retrieve:

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

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

     // Optionally insert other query parameters
     // Do QueryRequest.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)
     Write ##class(HS.Test.Service).SendSync(o,.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)
     Write ##class(HS.Test.Service).SendSync(obj,.rr)

   Quit
 }
```

2.4 Providing and Registering Documents to an XDS Document Repository

HealthShare Foundation 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 HealthShare Foundation 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 Ensemble message.

This section describes how to:

- Provide and register a CDA document to an XDS repository
- Provide and register a non-CDA document to an XDS repository

2.4.1 Provide and Register a CDA Document

HealthShare Foundation 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). HealthShare foundation 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.

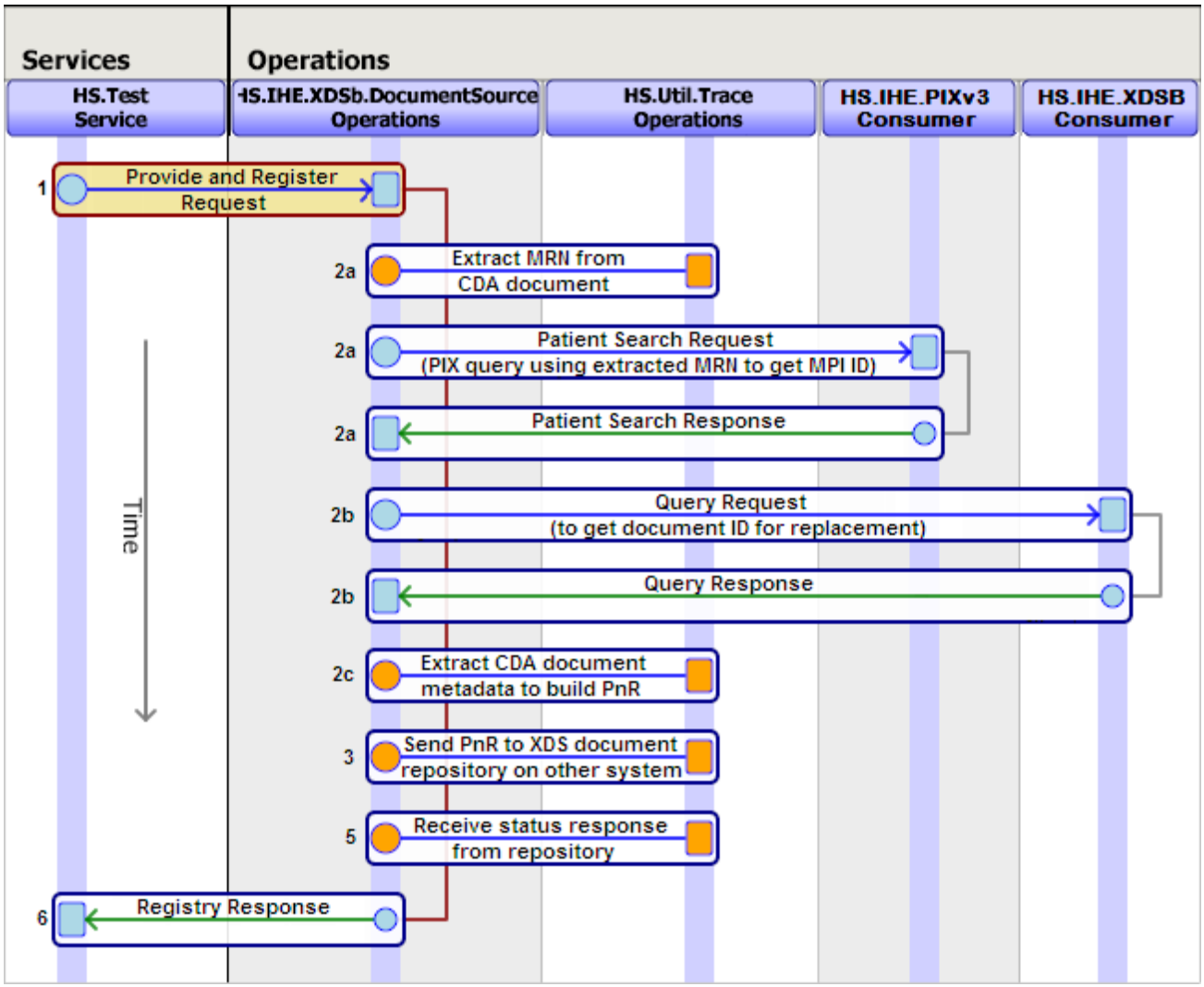
This section includes:

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

2.4.1.1 CDA Provide and Register Message Trace

Below 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.



2.4.1.2 CDA Provide and Register Procedure

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

2.4.1.3 CDA Provide and Register Components and Settings

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

Business Hosts	Document Source: **HS.IHE.XDSb.DocumentSource.Operations**
	PIX Consumer: **HS.IHE.PIXv3.Consumer.Operations** • if needed to get MPI ID
	Document Consumer: **HS.IHE.XDSb.Consumer.Operations** • if `<Replacement Context>` is specified in the message
Production Settings	DocumentTransform in the Document Source
	XDSbRepositoryServiceName in the Document Source

Ensemble Messages	**HS.Message.IHE.XDSb.ProvideAndRegisterRequest**
	HS.Message.XMLMessage: • RegistryResponse
	HS.Message.PatientSearchRequest (if PIX)
	HS.Message.PatientSearchResponse (if PIX)
	HS.Message.IHE.XDSb.QueryRequest (if replacement)
	HS.Message.IHE.XDSb.QueryResponse (if replacement)
Minimum Document Metadata	Open the file as the correct type, for example, open an XML as a text file.
	`<MIMEType>` — usually `text/xml`
	`<FormatCode>` — indicating the document type
	`<BodyCharacter>` — containing the file contents
XSL Transformations	IHE/XDSb/Version1/DocumentToProvideAndRegister.xsl
Service Registry Entries	XDSb.Repository
	PIXv3.Manager (if PIX)
	XDSb.Registry (if replacement)
External IHE Actor Endpoint	XDS Document Repository
	PIX Manager (if PIX)
	XDS Document Registry (if replacement)

2.4.1.4 CDA Provide and Register Example

The method below opens a CDA file and provides the minimum metadata required to provide and register the document:

```
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.SourceOID="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 CCD in XML, so open the file as a text file.
   Set tFile = ##class(%File).%New()
   Set tFile.Name="C:\wtemp\testccd.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).%New(
         "2.16.840.1.113883.10.20.1","HITSP","HL7 CCD Document")

   // 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.
    // HealthShare Foundation automatically adds the Patient ID and the Status of "approved"
    // to the context.
    Set tContext = ##class(HS.Message.IHE.XDSb.QueryItem).CodedValue(
           "$XDSDocumentEntryFormatCode","2.16.840.1.113883.10.20.1","HITSP")

    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
}
```

2.4.2 Provide and Register a Non-CDA Document

HealthShare Foundation can provide and register any kind of document to a document repository within the Affinity Domain via the IHE "XDS.bProvide and Register Document Set" transaction (PnR). You must provide HealthShare Foundation 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.

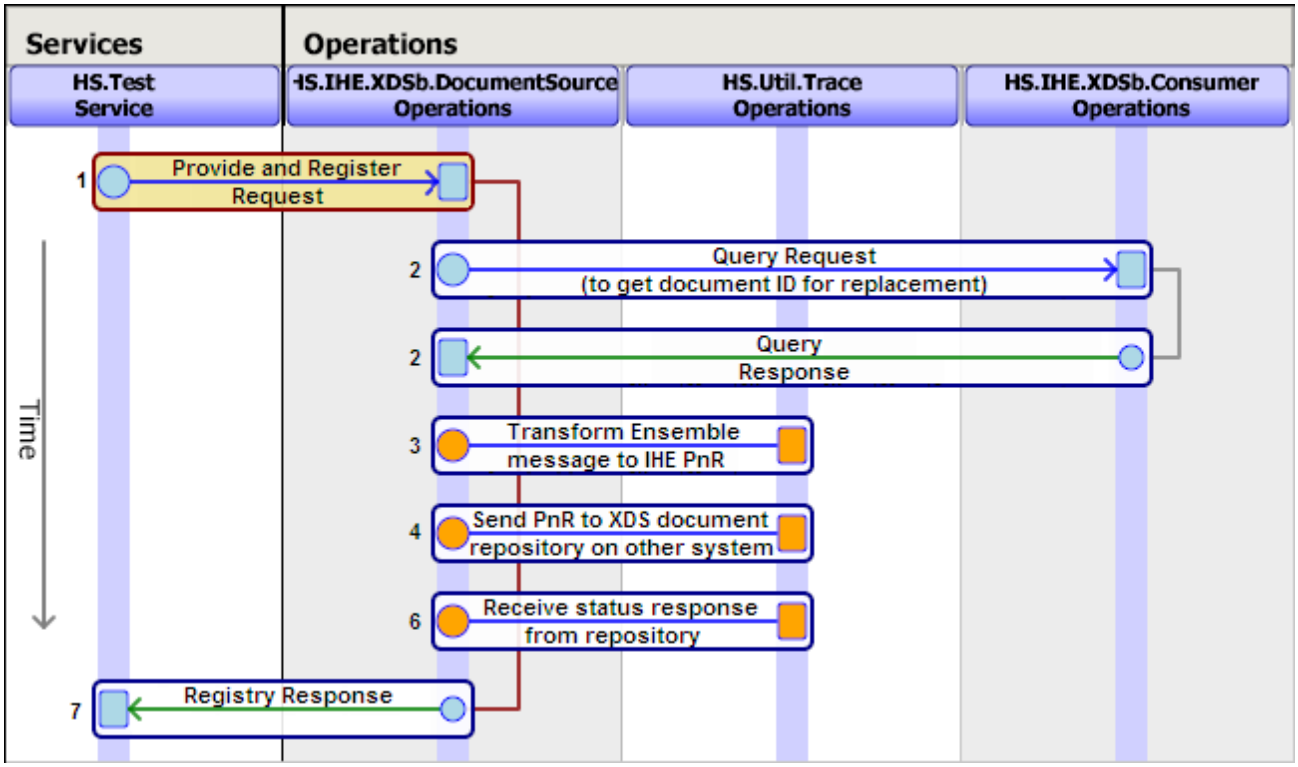
This section includes:

- 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

2.4.2.1 Non-CDA Provide and Register Message Trace

Below 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.



2.4.2.2 Provide and Register a Non-CDA Document Procedure

HealthShare Foundation 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 HealthShare 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 HealthShare Document Source transforms the Ensemble message into an IHE "ProvideAndRegisterDocumentSetRequest" using the XSL transformation specified in the DocumentTransform setting.

4. The HealthShare Document Source forwards the Provide and Register request to 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 HealthShare 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 HealthShare Document Source returns null.

2.4.2.3 Non-CDA Provide and Register Components and Settings

Table 2–7: Components and Settings Used in Provide and Register of a non-CDA Document

Business Hosts	Document Source: **HS.IHE.XDSb.DocumentSource.Operations**
	Document Consumer: **HS.IHE.XDSb.Consumer.Operations** • if `<Replacement Context>` is specified in the message
Production Settings	XDSbRepositoryServiceName in the Document Source
Ensemble Messages	**HS.Message.IHE.XDSb.ProvideAndRegisterRequest**
	HS.Message.XMLMessage: • RegistryResponse
	HS.Message.IHE.XDSb.QueryRequest (if replacement)
	HS.Message.IHE.XDSb.QueryResponse (if replacement)
Minimum Document Metadata **Note:** 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.
	`<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
	`<Author...>` (...Institution, ...Role, ...Person, ...Specialty)
XSL Transformations	IHE/XDSb/Version1/DocumentToProvideAndRegister.xsl
Service Registry Entries	XDSb.Repository
	XDSb.Registry (if replacement)
External IHE Actor Endpoints	XDS Document Repository
	XDS Document Registry (if replacement)

2.4.2.4 Provide and Register a non-CDA Document Example

The COS code below (which can be incorporated into a method) generates an XDS provide and register for a PDF document (in two boxes):

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

// Provide the Patient ID and document source. Use a PIX query to obtain the
// Patient ID if you have only the MRN.
Set tMessage.PatientId="100000001^^^&1.3.6.1.4.1.21367.2010.1.2.300&ISO"
Set tMessage.SourceOID="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 "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
Set tDocument.CreationTime="20110510102615-0400"
Set tDocument.LanguageCode="en-CA"

Set tDocument.ClassCode=##class(HS.IHE.XDSb.Types.CodedValue).%New(
    "Pathology Procedure","Connect-a-thon classCodes","Pathology Procedure")
Set tDocument.FormatCode=##class(HS.IHE.XDSb.Types.CodedValue).%New(
    "PDF","Connect-a-thon formatCodes","PDF")
Set tDocument.HealthcareFacilityTypeCode=##class(HS.IHE.XDSb.Types.CodedValue).%New(
    "Outpatient","Connect-a-thon healthcareFacilityTypeCodes","Outpatient")
Set tDocument.PracticeSettingCode=##class(HS.IHE.XDSb.Types.CodedValue).%New(
    "General Medicine","Connect-a-thon practiceSettingCodes","General Medicine")
Set tDocument.TypeCode=##class(HS.IHE.XDSb.Types.CodedValue).%New(
    "18768-2","2.16.840.1.113883.6.1","CELL COUNTS+DIFFERENTIAL STUDIES")
Do tDocument.EventCodeList.Insert(##class(HS.IHE.XDSb.Types.CodedValue).Create(
    "1.3.6.1.4.1.21367.2006.7.106","Connect-a-thon eventCodeList","OPT-OUT"))
Do tDocument.EventCodeList.Insert(##class(HS.IHE.XDSb.Types.CodedValue).Create(
    "1.3.6.1.4.1.21367.2006.7.108","Connect-a-thon eventCodeList","OPT-IN"))


// 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.
// HealthShare Foundation automatically adds the Patient ID and the Status of
// "approved" to the context.
Set tContext = ##class(HS.Message.IHE.XDSb.QueryItem).CodedValue("$XDSDocumentEntryEventCodeList",
    "1.3.6.1.4.1.21367.2006.7.106","Connect-a-thon eventCodeList")
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)

Quit
```

2.5 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.

HealthShare Foundation can query document registries in other Affinity Domains and request a list of documents for a patient via the XCPD *Cross-Gateway Patient Discovery* transaction and 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 Ensemble messaging infrastructure as XDS, and you can set up HealthShare Foundation so that document queries search both the local repository and other Affinity Domains.

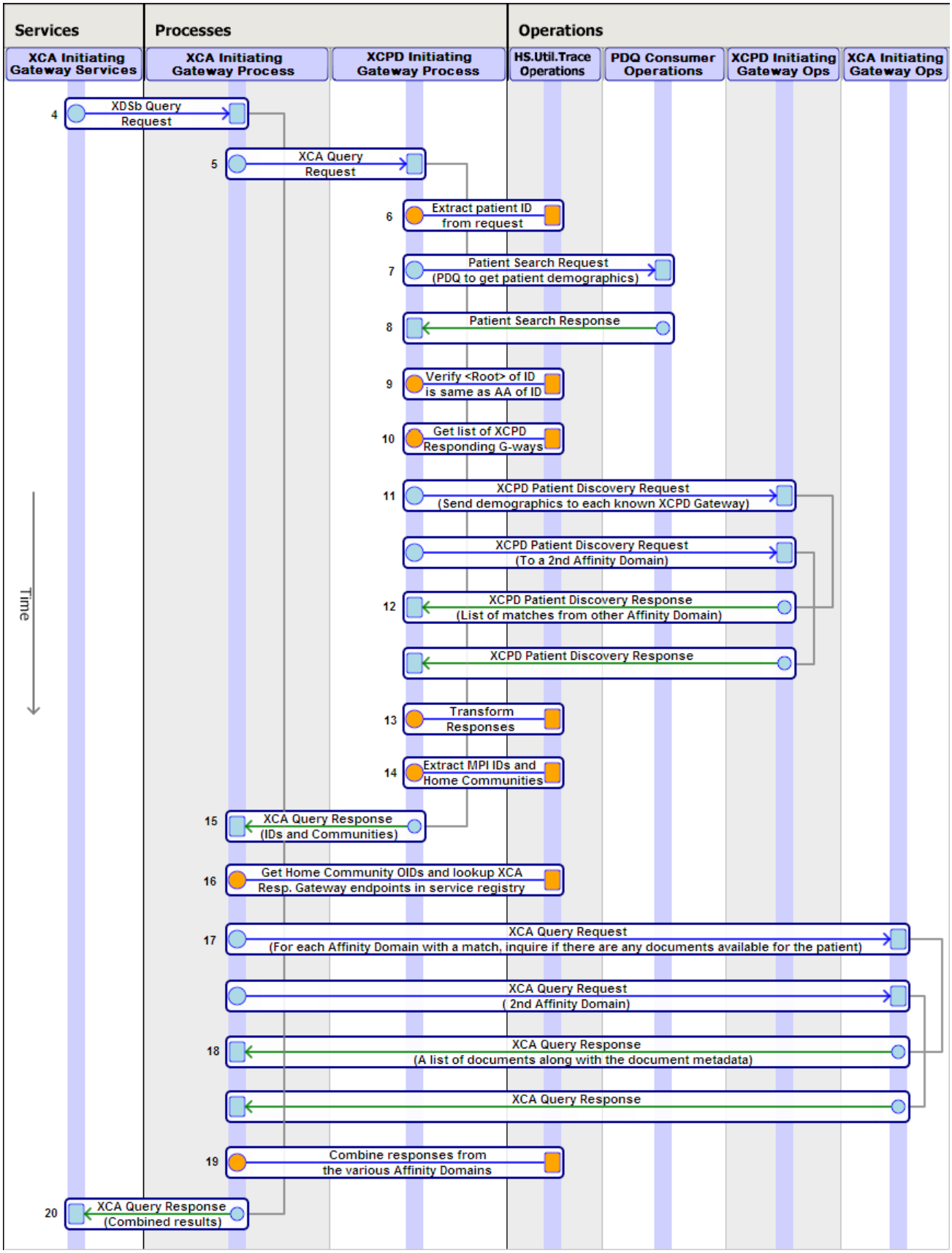
This section includes:

- XCA Query:
  - XCA query message trace
  - XCA query procedure
  - XCA query components
- XCA Retrieve:
  - XCA retrieve message trace
  - XCA retrieve procedure
  - XCA retrieve components
- Example XCA query and retrieve

2.5.1 XCA Query Message Trace

Below 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)

2.5.2 XCA Query Procedure

An XCA query transaction in HealthShare Foundation 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 Ensemble **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 Ensemble message into an IHE "XDSb_QueryRequest" message using an internally-specified transform.

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.

2.5.3 XCA Query Components and Settings

Table 2–8: Components and Settings Used in XCA Query

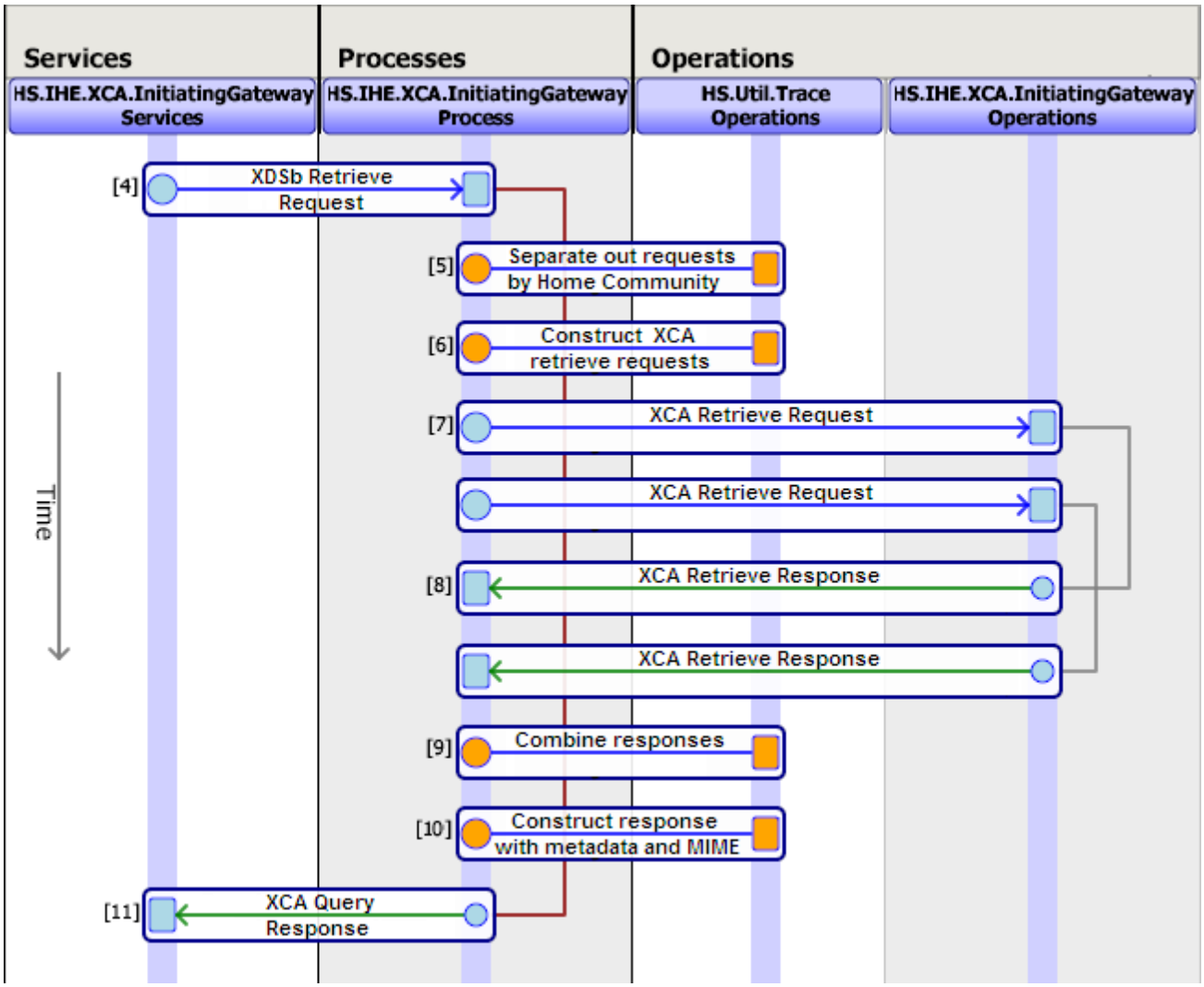
Business Hosts	XCA Document Consumer: **HS.IHE.XDSb.Consumer.Operations**
	XCA Initiating Gateway Service: **HS.IHE.XCA.InitiatingGateway.Services**
	XCA Initiating Gateway Process: **HS.IHE.XCA.InitiatingGateway.Process**
	XCA Initiating Gateway Operation: **HS.IHE.XCA.InitiatingGateway.Operations**
	XCPD Initiating Gateway Process: **HS.IHE.XCPD.InitiatingGateway.Process**
	XCPD Initiating Gateway Operation: **HS.IHE.XCPD.InitiatingGateway.Operation**
	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.
	XCPDInitiatingGatewayProcess in the XCA Initiating Gateway service
	PDQv3Consumer in the XCPD Initiating Gateway process
	Service Name in the PDQ Consumer operation
	XCPDQueryServiceNames in the XCPD Initiating Gateway process: • A comma-separated list of service registry entries pointing to XCPD Responding Gateway endpoints on other systems.
	XCAInitiatingGatewayOperation in the XCA Initiating Gateway process
	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.
	TransformToMetadata in the XDS.b Document Consumer

Ensemble Messages	**HS.Message.IHE.XDSb.QueryRequest**
	HS.Message.IHE.XDSb.QueryResponse
	HS.Message.XMLMessage: • XDSb_QueryRequest • XDSb_QueryResponse • XCA_QueryRequest • XCA_QueryResponse • XCPD_PatientDiscoveryRequest • XCPD_PatientDiscoveryResponse
	HS.Message.PatientSearchRequest (for PDQ query)
	HS.Message.PatientSearchResponse (for PDQ query)
XSL Transformations	QueryRequestToXDSbQuery.xsl in the XDS.b Document Consumer
	IHE/PDQ/Version1/PatientSearchToPRPAIN201305UV.xsl in PDQ
	IHE/PDQ/Version1/PRPAIN201306UVToPatientSearchResponse.xsl in PDQ
	Message-To-Metadata.xsl in the XDS.b Document Consumer
Service Registry Entries	XCA.IG (or similar) • points to the XCA Initiating Gateway service
	PDQv3.Supplier (for PDQ query)
	XCPD.RespondingGateway.1 XCPD.RespondingGateway.2 ...etc: • URLS of the various XCPD Responding Gateways in the Home Communities
	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
	XCPD Responding Gateway(s)
	XCA Responding Gateway(s)

2.5.4 XCA Retrieve Message Trace

Below 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.



2.5.5 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 Ensemble 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_QueryResponse 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".

2.5.6 XCA Retrieve Components and Settings

Table 2–9: Components and Settings Used in XCA Retrieve

Business Hosts	XCA Document Consumer: **HS.IHE.XDSb.Consumer.Operations**
	XCA Initiating Gateway Service: **HS.IHE.XCA.InitiatingGateway.Services**
	XCA Initiating Gateway Process: **HS.IHE.XCA.InitiatingGateway.Process**
	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.
	XCAInitiatingGatewayOperation in the XCA Initiating Gateway process
Ensemble Messages	**HS.Message.IHE.XDSb.RetrieveRequest**
	HS.Message.IHE.XDSb.QueryResponse
	HS.Message.XMLMessage: • XCA_RetrieveRequest • XCA_RetrieveResponse • XCA_QueryResponse • XDSb_RetrieveResponse • XDSb_RetrieveResponse (RetrieveDocumentSetResponse)
XSL Transformations	RetrieveRequestToXDSbRetrieve.xsl in XCA Document Consumer
	Message-To-Metadata.xsl in XCA Document Consumer
Service Registry Entries	XCA.IG (or similar) • points to the local XCA Initiating Gateway service
	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.
	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
	A repository OID for the local XDS repository
External IHE Actor Endpoints	XCA Responding Gateway(s) XDS Repository (in the local Home Community)

2.5.7 XCA Query and Retrieve Example

The method below sends an Ensemble 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)
  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
}
```

2.6 Generating a CCD Document from an HL7 Message

HealthShare Foundation includes transformations that can accept an HL7 message, transform it into SDA (an internal representation), and then transform it into a CCD, for example a C32 document. You can then provide and register the CCD to an XDS document repository.

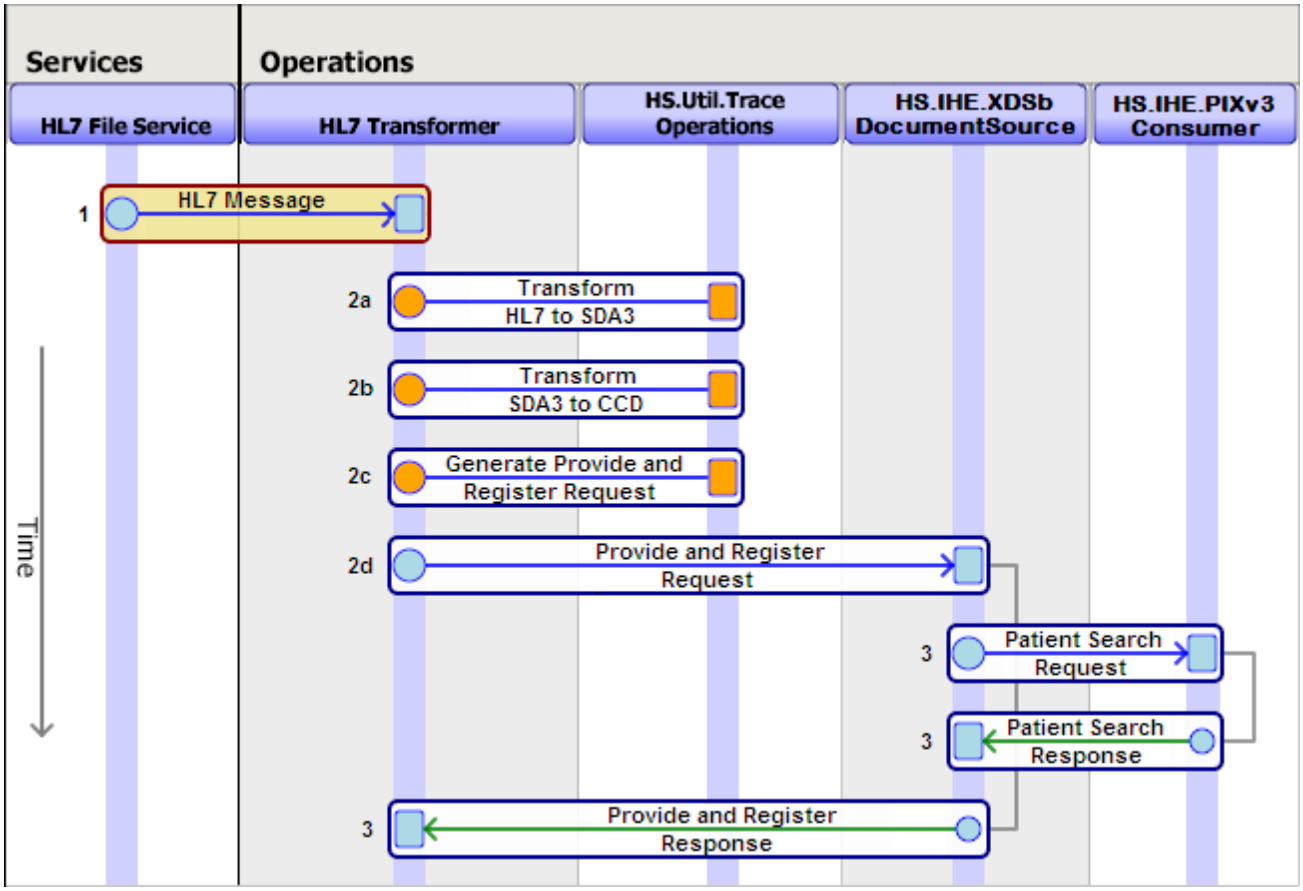
This section includes:

- HL7 to CCD provide and register message trace
- HL7 to CCD provide and register procedure
- HL7 to CCD provide and register components
- HL7 to CCD provide and register example

2.6.1 HL7 to CCD Provide and Register Message Trace

Below is an annotated message trace for an HL7 message transformed into a CCD 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 CCD.

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.



2.6.2 HL7 to CCD Provide and Register Procedure

To set up HealthShare Foundation to convert HL7 into CCD:

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 CCD document, for example, a C32, C48, or other form.
   c. package the CCD into a Provide and Register request.
   d. send the Provide and Register request to the HealthShare Document Source.
3. The Document Source then follows the path laid out in the section "Provide and Register a CDA document".

2.6.3 HL7 to CCD Provide and Register Components and Settings

Table 2–10: Components and Settings Used in Transformation of HL7 to CCD followed by a Provide and Register

Business Hosts	HL7 Service: EnsLib.HL7.Service.FileService (or similar)
	HL7 to CCD Transformer: Custom Business Operation
	Document Source: **HS.IHE.XDSb.DocumentSource.Operations**
	PIX Consumer: **HS.IHE.PIXv3.Consumer.Operations**
Production Settings	TargetConfigNames in the HL7 Service
Ensemble Messages	**HS.Message.IHE.XDSb.ProvideAndRegisterRequest**
	HS.Message.XMLMessage: • RegistryResponse
	HS.Message.PatientSearchRequest
	HS.Message.PatientSearchResponse
XSL Transformations	SDA3/SDA-to-C32v25.xsl
Service Registry Entries	XDSb.Repository
	PIXv3.Manager
External IHE Actor Endpoints	XDS Document Repository
	PIX Manager

2.6.4 Example Transformer Business Operation to Generate a CCD Document from an HL7 Message

Below is a sample business operation that accepts an HL7 message and transforms it into a CCD:

```
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 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,.tSDA3)
        Quit:$$$ISERR(tSC)

        // Transform the SDA3 to a C32
        Set tSC= tTransformer.Transform(tSDA3,"SDA3/SDA-to-C32v25.xsl",.tStream)
```

```
        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).%New(
           "2.16.840.1.113883.10.20.1","HITSP","HL7 CCD Document")
        Set tDocument.MimeType="text/xml"

        // Copy the C32 into the message
        Do tDocument.BodyCharacter.CopyFrom(tStream)
        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
 }
}
```

2.7 Generating a CCD Document by Querying an Internal Database

You can generate a CCD document using XSL transformations provided by HealthShare Foundation that transform SDA into CCD, for example, *HSF-home*\CSP\xslt\SDA3\SDA-to-C37v23.xsl. You must write code to transform the query result from your internal database into SDA.

See the chapter "SDA Documents" for more details on the structure of the SDA.

2.8 Receiving a CCD Document and Converting it to an Internal Format

If you receive a CCD document and wish to parse it and store its contents in an internal database, apply the provided transformation from CCD 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" for more details on the structure of the SDA.

3

Registry Settings for IHE Communication

In addition to configuring your Ensemble 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 *HealthShare Foundation Administration Guide* for details on how the create registry entries.

3.1 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.

3.2 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.

3.3 Configuration Registry Entries for IHE

The following is the minimum set of configuration registry entries required for IHE communication:

Key	Value
\IHE\AffinityDomain	HomeCommunity
\IHE\HomeCommunity	HomeCommunity

4

SDA Documents

SDA (Summary Document Architecture) is an InterSystems format based on CDA (Clinical Document Architecture). This is the format used to represent patient data in HealthShare Foundation.

Information in SDA format is represented by a Caché object that consists of instances of HS.SDA3.Patient, HS.SDA3.Encounter, and other Caché classes. These classes are XML-enabled. You must convert your patient record query result into the SDA XML format in order to use the provided transformations from SDA into the various CCD formats.

Note: This version of HealthShare Foundation is based on SDA3, but previous versions were based on SDA(2). This section describes SDA3, which InterSystems strongly recommends that you use.

4.1 The Basic XML Structure of an SDA Document

The major sections in an SDA document are as follows:

```
<Container>
  <Patient>
  <Encounters>
  <AdvanceDirectives>
  <Alerts>
  <Allergies>
  <Appointments>
  <Problems>
  <Diagnoses>
  <Documents>
  <LabOrders>
  <RadOrders>
  <OtherOrders>
  <Medications>
  <Vaccinations>
  <Observations>
  <PhysicalExams>
  <Procedures>
  <FamilyHistories>
  <IllnessHistories>
  <SocialHistories>
  <CustomObjects>
</Container>
```

The following rules apply to an SDA document:

1. There is a single <Patient>.
2. <Patient> is required and must be the first entry (after properties of the <Container> such as <Action> or <EventCode>).
3. All sections other than <Patient> are optional.
4. All sections other than patient may contain multiple entries, for example:

```
<Procedures>
   <Procedure>
      ...
   </Procedure>
   <Procedure>
      ...
   </Procedure>
</Procedures>
```

5. <Encounters>, if included, must appear directly after <Patient>.
6. All other sections may appear in any order.
7. Any entry may optionally reference an encounter number. In this case, <Encounters> must be included and must contain an <Encounter> with that <EncounterNumber>.

4.2 The Patient in SDA

The HS.SDA3.Patient class represents the patient. This class contains properties that store information like the following:

- Demographic information and other basics, for example: name, a list of addresses, gender, marital status, race, religion, and so on
- Patient numbers

The following table shows the properties of HS.SDA3.Patient. Some of these properties are simple (for example, strings), some are complex objects which contain properties of their own, and some are lists of complex objects:

Table 4–1: Properties in HS.SDA3.Patient

Simple Property	Complex Object	List of Complex Objects
BirthTime	Citizenship	Addresses
DeathLocation	ContactInfo	Aliases
DeathTime	DeathDeclaredBy	PatientNumbers
InactiveMRNs	FamilyDoctor	PriorPatientNumbers
IsDead	Gender	OtherLanguages
MothersMaidenName	MaritalStatus	SupportContacts
MPIID	Name	
	PrimaryLanguage	
	Race	
	Religion	

4.3 Encounters in SDA

An encounter encompasses all of the medical information related to a specific medical incident. The HS.SDA3.Encounter class represents a medical encounter of a patient. Other sections in the SDA, representing orders, procedures, exams and the like can refer to the encounter number, and thus be tied together.

The following table shows the properties of HS.SDA3.Encounter, some of which are simple (for example, strings), some of which are complex objects (containing properties of their own), and some of which are lists of complex objects:

Table 4–2: Properties in HS.SDA3.Encounter

Simple Property	Complex Object	List of Complex Objects
AccountNumber	AdmissionSource	AttendingClinicians
AssignedBed	AdmisssionType	ConsultingClinicians
AssignedRoom	AdmitReason	HealthFunds
AssignedWard	AdmittingClinician	
EncounterMRN	DischargeLocation	
EncounterNumber	HealthCareFacility	
EncounterType	ReferringClinician	
EndTime		
PreAdmissionNumber		
PriorVisitNumber		
VisitDescription		
VisitStatus		

4.4 For More Information on SDA

For complete details on the SDA model, see the class reference for HS.SDA3.Container and the other classes in the HS.SDA3 package.

To access the class reference for the SDA classes:

1. Go to the documentation home page by selecting **Documentation** from the HealthShare Foundation cube.
2. Select **Class Reference** from the list at the top of the documentation home page.
3. Using the drop-down list at the top of the left-hand pane, change to the HSLIB namespace.
4. Uncheck **System Classes**.
5. Drill down the hierarchy of package names to HS, then SDA3 and then to the class of interest.

To generate an XML schema document (XSD) for SDA, run the following utility in the HSLIB namespace:

```
Do ##Class(HS.SDA3.Container).ExportXMLSchema()
```

The utility will ask for a filename. Enter the name of the file and include the `.xsd` extension.

5

Auditing

HealthShare Foundation offers all of the auditing capabilities of Caché and Ensemble. In addition, it can send audit messages to an ATNA repository.

5.1 Basic Auditing

The HS.Audit.Consolidation.Service is an Ensemble business service that batches audit events and periodically updates the Caché audit logs. To set up auditing, enable this business service in your HealthShare Foundation production.

5.2 ATNA Auditing

To add ATNA auditing:

1. Add one of the following business operations to your HealthShare Foundation 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.Audit.Consolidation.Service to the name of the business operation you installed in step 1.
4. Optionally edit the contents of the ExcludeUsers field in the HS.Audit.Consolidation.Service to control which events are audited.