FHIR Support in InterSystems IRIS for Health

Version 2019.4
2020-10-16

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

FHIR Support in InterSystems IRIS for Health
InterSystems Version 2019.4 2020-10-16

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

About This Book

This book explains how to use the FHIR® components of InterSystems IRIS for Health™.

Chapters include:

- About FHIR Support
- Configuring a Sample FHIR Production
- Elements of the Sample FHIR Production
- The FHIR Server and Client
- FHIR Message Handling
- The FHIR Object Model
- Using and Customizing SDA-FHIR DSTU2 Transformations
- Using and Customizing SDA-FHIR STU3 Transformations

An appendix details FHIR IHE profile compliance for InterSystems IRIS for Health.

For more information on InterSystems IRIS for Health, see the following books:

- *IHE Use Cases* describes how to set up IHE communication.
- *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.
- *InterSystems IRIS for Health Installation Guide* describes how to install InterSystems IRIS for Health and discusses considerations for mirroring and Docker containers.
- *InterSystems Supported Platforms* lists the platforms on which InterSystems IRIS for Health is supported.

For more information about the FHIR standard, see http://www.hl7.org/fhir/; for FHIR license and legal terms see http://www.hl7.org/fhir/license.html.

1

About FHIR Support

Support for FHIR® in InterSystems IRIS for Health includes:

- Support for DSTU2 and STU3.
- A FHIR server that can accept and respond to REST requests from FHIR clients.
- A FHIR client that can submit REST requests to a FHIR server endpoint and receive responses returned from it.
- A FHIR message model that provides an internal representation of FHIR-based requests and responses.
- The object model for FHIR resources that conforms to the current FHIR standard and implements all resource and data types.
- Transformations to convert data between SDA and FHIR DSTU2 or SDA and FHIR STU3 that allow a straightforward translation between formats.
- A resource repository that stores and indexes FHIR resources.
- Support for the FHIR IHE PIXm, PDQm, and MHD profiles.

1.1 About the FHIR Server

The FHIR server:

- Receives REST requests from FHIR clients.
- Transforms requests into internal FHIR message format.
- Handles FHIR XML and JSON payloads without additional configuration.
- Supports CRUD activities, search, and other interactions such as FHIR transactions and operations.
- Can perform further processing.
- Can forward data to the FHIR resource repository.

For more information about the FHIR server, see The FHIR Server and Client.

1.2 About the FHIR Client

The FHIR client:

- Transforms messages from SDA internal format into FHIR format.
- Transforms messages from FHIR internal format into FHIR REST requests.
- Submits REST calls to a FHIR server.
- Processes both individual resources and resource bundles.
- Supports both XML and JSON formats.

For more information about the FHIR client, see The FHIR Server and Client.

1.3 About the FHIR Message Model

The FHIR message model:

- Includes request and response formats for internal message passing.
- Contains transaction metadata (for the client and from the server).
- Contains payload metadata (with both content and resource type information).
- Supports the FHIR paradigms for both messages and REST interactions.
- Transmits its payload as a stream (not as an object), and serialization and deserialization may easily be done when required.

For more information about the FHIR message model, see FHIR Message Handling.

1.4 About the Object Model for FHIR Resources

The object model for FHIR resources:

- Implements all FHIR DSTU 2 (1.0.2) and STU3 (3.0.1) resources.
- Is generated from the FHIR schema.
- Supports all FHIR primitives.
- Supports extensions.
- Converts objects to and from streams.

For more information about the object model for FHIR resources, see The FHIR Object Model.

1.5 About Converting Data between SDA and FHIR

InterSystems IRIS for Health:

- Includes built-in data transformations for a robust set of commonly used FHIR resources and SDA data types.
- Supports customization for each individual transformation.
- For customization, supports both a visual interface or programmatic modification, so that a wider range of implementers (and not just developers) can modify them:
  - For visual editing, implementers, such as interface analysts, can use the DTL Editor for transformations.
  - For coding, each transformation is a class, so developers can use InterSystems development tools to create customized subclasses.

For more information about SDA-FHIR DSTU2 transformations, see Using and Customizing SDA-FHIR DSTU2 Data Transformations.

For more information about SDA-FHIR STU3 transformations, see Using and Customizing SDA-FHIR STU3 Data Transformations.

1.6 About the FHIR Resource Repository

The FHIR resource repository:

- Supports storage of all FHIR resource types.
- Provides an index for searches.
- Supports storing resources with FHIR extensions.
- Has FHIR conformance and capability statements available, which specify the scope of its activities.
- Makes it possible to use your production as a FHIR reference server.
- Stores data as individual FHIR resources.
- Allows the consolidation of data from legacy systems (which hold data only in monolithic formats) without modifying those systems.
- Is managed by specific business operations, which provide central points of control.
- Supports the creation of temporary repositories of data via user sessions

For more information about the FHIR resource repository, see Elements of the FHIR Production.

2

Installing and Configuring a FHIR Production

This section describes the following topics:

- Options for Your FHIR Production
- Installing and Configuring a FHIR Production
- Using the FHIR Test Utility
- Exploring the Production with a Third-Party FHIR Client

2.1 Options for Your FHIR Production

IRIS for Health ships with a FHIR production installer that facilitates initial setup.

The simplest FHIR production you can install includes server, client, and web application elements for FHIR processing and messaging. You can install these elements for DSTU2, STU3, or both.

You can also choose to install a FHIR resource repository for DSTU2, STU3, or both.

If you want to install elements needed for the FHIR IHE profiles (PIXm, PDQm, and MHD), you can select them during installation.

The FHIR IHE profiles provide these capabilities:

- PIXm: A PIXm query provides an MRN (Medical Record Number) and assigning authority and receives back the name and MPIID of a single patient. InterSystems IRIS for Health supports PIXm as a FHIR proxy to the existing PIX infrastructure. PIXm receives FHIR requests from a REST client, converts them to Patient Search Request messages, forwards those messages to the existing PIX infrastructure, receives the returned Patient Search Response, converts the response to a FHIR response, and returns it back to the REST client.
- PDQm: A PDQm query provides a partial set of patient demographics and receives back full demographics for one or more MPIIDs (each of which represents a patient) that match the provided demographics. InterSystems IRIS for Health supports PDQm as a FHIR proxy to the existing PDQ infrastructure. PDQm receives FHIR requests from a REST client, converts them to Patient Search Request messages, forwards those messages to the existing PDQ infrastructure, receives the returned Patient Search Response, converts the response to a FHIR response, and returns it back to the REST client.

- MHD: A MHD query is a way to search a document registry and repository. Server-side HealthShare MHD components act as a proxy to an XDS.b infrastructure. A MHD Find Manifests query provides a Find Document Manifests Request and receives back a Find Document Manifests Response. A MHD Find References query provides a Find Document References Request and receives back a Find Document References Response. The MHD server receives FHIR requests from a REST client, converts them to XDS.b request messages, forwards those messages to the existing XDS.b infrastructure, receives the returned XDSb response, converts the response to a FHIR response, and returns it back to the REST client.

FHIR IHE support in InterSystems IRIS for Health is substantially compliant with the IHE profile specifications. For a list of differences from the specification, see "Compliance with FHIR IHE Profile Specifications".

Important: If you want to mirror your FHIR production, see the "Mirroring InterSystems IRIS for Health" chapter of the *InterSystems IRIS for Health Installation Guide*.

2.2 Installing and Configuring a FHIR Production

InterSystems IRIS for Health includes a utility that creates and configures a production as a FHIR server and client. This utility is a method of the HS.Util.Installer class in the HSLIB namespace.

2.2.1 Before You Begin

You will need to determine the appropriate values for the following settings:

Business Host	Use	Setting	Value	Notes

Business Host	Use	Setting	Value	Notes
HSFHIRPIXmConsumeOperations[FHIRVersion]	Client side (requests to a FHIR server)	ServiceName	Name of Service Registry entry that contains the endpoint for the FHIR server to which requests will be posted	
HSFHIRPIXmManageProcess[FHIRVersion]	Server side (requests to an MPI or patient registry)	MPIOperations	Name of interoperability component that takes an HS.Message.PatientSearchRequest as input, derives the information from it, and forwards that information on to an MPI or patient registry for performing patient search. The component must also be able to receive the response back from the MPI or patient registry, convert the response back to HS.Message.PatientSearchResponse, and return it to HS.FHIR.PIXm.Manager.Process..	Custom development of the interoperability component is required.
HSFHIRPDQmConsumeOperations[FHIRVersion]	Client side (requests to a FHIR server)	ServiceName	Name of Service Registry entry that contains the endpoint for the FHIR server to which requests will be posted.	
HSFHIRPDQmSupplyProcess[FHIRVersion]	Server side (requests to an MPI or patient registry)	MPIOperations	Name of interoperability component that takes an HS.Message.PatientSearchRequest as input, derive the information from it, and forward that information on to an MPI or patient registry for performing patient search. The component must also be able to receive the response back from the MPI or patient registry, convert the response back to HS.Message.PatientSearchResponse, and return it to HS.FHIR.PIXm.Manager.Process.	Custom development of the interoperability component is required.

Business Host	Use	Setting	Value	Notes
HS.FHIRMHD.ConsumerOperations[FHIRVersion]	Client side (requests to a MHD responder)	ServiceName	Name of Service Registry entry that contains the endpoint for the MHD responder to which query or document retrieval requests will be posted	
HS.FHIRMHD.DocumentSourceOperations[FHIRVersion]	Client side (requests to an MHD recipient)	ServiceName	Name of Service Registry entry that contains the endpoint for the MHD recipient to which document registration requests will be posted	
HS.IHE.XDSb.DocumentSource.Operations	Server side (requests to an XDS.b registry)	XDSbRegistryServiceName	Name of Service Registry entry that contains the endpoint for the XDS.b registry to which document registration requests will be posted	
HS.IHE.XDSb.DocumentSource.Operations	Server side (requests to an XDS.b repository)	XDSbRepositoryServiceName	Name of Service Registry entry that contains the endpoint for the XDS.b repository to which query or document retrieval requests will be posted	
HS.IHE.XDSb.Consumer.Operations	Server side (requests to an XDS.b registry)	XDSbRegistryServiceName	Name of Service Registry entry that contains the endpoint for the XDS.b registry to which document registration requests will be posted	
HS.IHE.XDSb.Consumer.Operations	Server side (requests to an XDS.b repository)	XDSbRepositoryServiceName	Name of Service Registry entry that contains the endpoint for the XDS.b repository to which query or document retrieval requests will be posted	

2.2.2 Running the FHIR Installer and Adjusting Required Settings

Important: InterSystems strongly recommends that you install a production on a test server to familiarize yourself with FHIR functionality in InterSystems IRIS for Health prior to creating a production system.

You do not have to install all of the included components right away; the installer can be rerun, at which time you select the existing namespace into which to install the new components.

To run the installer:

1. If you are configuring a new installation of InterSystems IRIS for Health, make sure that the license key is activated.

2. Log into the IRIS Terminal with a role that has the **%Admin_Manage:Use** privilege.
3. Go to the HSLIB namespace:

```
zn "HSLIB"
```

4. Run the **Install** method of the HS.HC.Util.Installer.FHIR class:

```
do ##class(HS.HC.Util.Installer.FHIR).Install()
```

The method prompts you to enter:

- The name of the namespace in which the FHIR production will run: for example, **FHIRNamespace**. If you enter the name of an existing namespace, it must be a FHIR namespace.
- Whether you want to install DSTU2. If you select Y, you are then presented with options for:
  - The URL for the FHIR web application (you can accept the default).
  - The URL for the Open ID Connect FHIR web application (you can accept the default)
- Whether to install a FHIR resource repository for the selected FHIR version.
- Whether to install PIXm for the selected FHIR version. (You can install PIXm later if you prefer.) If you choose to install this option, you will be prompted to enter:
  - The URL for the PIXm web application (you can accept the default)
  - The URL for the Open ID Connect PIXm web application (you can accept the default).
- Whether to install PDQm for the selected FHIR version. (You can install PDQm later if you prefer.) If you choose to install this option, you will be prompted to enter:
  - The URL for the PDQm web application (you can accept the default)
  - The URL for the Open ID Connect PDQm web application (you can accept the default).
- Whether to install MHD for the selected FHIR version. (You can install MHD later if you prefer.) If you choose to install this option, you will be prompted to enter:
  - The URL for the MHD *recipient* web application (you can accept the default)
  - The URL for the Open ID Connect MHD *recipient* web application (you can accept the default).
  - The URL for the MHD *responder* web application (you can accept the default)
  - The URL for the Open ID Connect MHD *responder* web application (you can accept the default).
- Whether you want to install STU3. If you enter Y for yes, you will be asked all of the above questions again.

After you enter the required input, the method performs the following actions *for each selected FHIR version*:

- It creates the specified namespace, such as **FHIRNAMESPACE**, or uses the name of the existing namespace you specified.
- It creates and starts a production with the name <*namespace*>PKG.FoundationProduction, such as **FHIRNAMESPACEPKG.FoundationProduction**.
- It creates two web applications, one with an HTTP endpoint and the other with an HTTPS endpoint, for the FHIR versions that you specified.

  Note: The applications with HTTP endpoints are password-protected by default.
- It creates Open ID Connect web applications for the FHIR versions that you specified.

- It creates Service Registry entries for the web applications. Each of these entries specifies a unique loopback URL when the production (as a FHIR client) makes calls into itself (as a FHIR server).
- It creates a sample facility called **EMERGENCY** with associated assigning authorities. This allows you to create a sample patient with the FHIR test utility; for more information, see the section "Using the FHIR Test Utility."
- It creates credentials for the FHIR production. The credentials are named `HS_Services`. This user account is used by the FHIR test utility.
- If you opted to install a FHIR resource repository, it creates the repository and submits a FHIR `Conformance` (DSTU2) or `CapabilityStatement` (STU3) resource to the repository.
- If you opted to install PIXm, it creates a web application and associated Open ID Connect web application, a client-side business operation, and a server-side business service and process.
- If you opted to install a PDQm web application, it creates the application and associated Open ID Connect web application, a client-side business operation, and a server-side business service and process.
- If you opted to install an MHD web application, it creates the following for recipient and responder: a web application and an associated Open ID Connect web application, client-side business operations, and server-side business services and processes.

Important: All of the endpoints you chose to install are available for use in the FHIR test utility.

The remainder of the FHIR documentation uses entities created by the **Install** method as examples, including **FHIRNAMESPACE** and the `/csp/healthshare/fhirnamespace/fhir/*` web applications.

The **FHIRNAMESPACEPKG.FoundationProduction** production (subsequently referred to as the **FHIRNAMESPACE** production) is now up and running.

You can now adjust the business host settings described in "Before You Begin". You must also set up all facilities, assigning authorities, and OIDs that will be referenced in your organization's health care data; for more information, see *Registry Guide for InterSystems IRIS for Health and Health Connect*.

2.2.3 Installing FHIR IHE Components Separately

If you did not install components for PIXm, PDQm, or MHD when you ran the main FHIR installer, you can install them later.

2.2.3.1 Installing PIXm Components Separately

To install PIXm components, run the following command in Terminal in any FHIR namespace:

```
set status = ##class(HS.HC.Util.Installer.Kit.FHIR.PIXm).Add(, "fhirNamespace",
"fhirVersion", "pixmWebApp", "pixmOIDCWebApp")
```

where:

- *fhirNamespace* — the name of the desired namespace
- *fhirVersion* — DSTU2 or STU3
- *pixmWebApp* — the path of the web application for the PIXm Patient Identifier Cross-reference Manager actor. If left empty, defaults to `/csp/healthshare/`*`fhirNamespace`*`/pixm/`*`fhirVersion`*.
- *pixmOIDCWebApp* — the path of the OIDC web application for the PIXm Patient Identifier Cross-reference Manager actor. It requires an Open ID Connect/OAuth 2.0 access token. If left empty, defaults to `/csp/healthshare/`*`fhirNamespace`*`/pixm-oidc/`*`fhirVersion`*.

2.2.3.2 Installing PDQm Components Separately

To install PDQm components, run the following command in Terminal in any FHIR namespace:

```
set status = ##class(HS.HC.Util.Installer.Kit.FHIR.PDQm).Add(, "fhirNamespace",
"fhirVersion", "pdqmWebApp", "pdqmOIDCWebApp")
```

where:

- *fhirNamespace* — the name of the desired namespace
- *fhirVersion* — DSTU2 or STU3
- *pdqmWebApp* — the path of the web application for the PDQm Patient Demographics Supplier actor. If left empty, defaults to `/csp/healthshare/fhirNamespace/pdqm/fhirVersion`.
- *pdqmOIDCWebApp* — the path of the OIDC web application for the PDQm Patient Demographics Supplier actor. It requires an Open ID Connect/OAuth 2.0 access token. If left empty, defaults to `/csp/healthshare/fhirNamespace/pdqm-oidc/fhirVersion`.

2.2.3.3 Installing MHD Components Separately

To install MHD components, run the following command in Terminal in any FHIR namespace:

```
set status = ##class(HS.HC.Util.Installer.Kit.FHIR.MHD).Add(, "fhirNamespace",
"fhirVersion", "recipientWebApp", "recipientOIDCWebApp", "responderWebApp",
"responderOIDCWebApp")
```

where:

- *fhirNamespace* — the name of the desired namespace
- *fhirVersion* — DSTU2 or STU3
- *recipientWebApp* — the path of the web application for the MHD Document Recipient actor. If left empty, defaults to `/csp/healthshare/fhirNamespace/mhdrecipient/fhirVersion`.
- *recipientOIDCWebApp* — the path of the OIDC web application for the MHD Document Recipient actor. It requires an Open ID Connect/OAuth 2.0 access token. If left empty, defaults to `/csp/healthshare/fhirNamespace/mhdrecipient-oidc/fhirVersion`.
- *responderWebApp* — the path of the web application for the MHD Document Responder actor. If left empty, defaults to `/csp/healthshare/fhirNamespace/mhdresponder/fhirVersion`.
- *responderOIDCWebApp* — the path of the OIDC web application for the MHD Document Responder actor. It requires an Open ID Connect/OAuth 2.0 access token. If left empty, defaults to `/csp/healthshare/fhirNamespace/mhdresponder-oidc/fhirVersion`.

2.3 Using the FHIR Test Utility

The FHIR sample production includes a FHIR test utility. The FHIR test utility is a sample FHIR client, which has a simple user interface that allows you to send requests to your FHIR production as if it were an independently operating FHIR server.

2.3.1 Creating and Searching for a Patient Resource

To create a patient resource:

1. Log in to the Management Portal as a user with the **%HS_Administrator** role.
2. Go to the **FHIR Test Utility** page (within your FHIR namespace, **Health** > **FHIR Test Utility**).
3. Click **Patient Create / Update**, which displays the **Patient Create / Update** page.
4. From the **Endpoint** list, select either **FHIR.Service.DSTU2** or **FHIR.Service.STU3**.

   Note: The availability of these services depends on the options you selected when you installed the sample production.

5. Select the **FHIR Format** in which the patient resource will be submitted.
6. Select the **FHIR Version** in which the patient resource will be submitted.
7. On that page, enter information for the test patient.
   - Be sure to specify values for **First Name**, **Last Name**, **Gender** and **Date of Birth**.
   - For **Assigning Authority**, select **Emergency**, which in turn populates values for the **AssigningAuthority OID**, **Facility**, and **MRN** fields.
8. Click **Submit Request**.
9. There should be a response in the **Response Payload** area. Verify that the response contains information for the sample patient resource you created.
10. Note the MRN for the patient; you will use it in the next step.

To search for a patient:

1. In the FHIR Test Utility, click **Patient Search**, which displays the **Patient Search** page.
2. On the **Patient Search** page:
   a. From the **Endpoint** list, select either **FHIR.Service.DSTU2** or **FHIR.Service.STU3**, depending on which endpoint you chose when you created the Patient resource.

      Note: The availability of these services depends on the options you selected when you installed the sample production.

   b. Select the **FHIR Format** in which you want to see the response containing the Patient resource.
   c. Select the **FHIR Version** in which you want to see the response containing the Patient resource.
3. In the **Identifier** field, enter the MRN for the patient you created in the steps above.
4. Select **Submit Request**.
5. Verify that the response contains information for the sample patient. This confirms that the production has been properly configured.

2.3.2 Installing and Using a Sample Open ID Connect Client

A sample Open ID Connect (OIDC) client is included with your InterSystems IRIS for Health installation. You can try it in conjunction with the OIDC endpoints in the FHIR Test Utility.

To install the client and use it with the FHIR OIDC endpoints:

1. Install an SSL certificate on the web server used for InterSystems IRIS for Health and configure the web server for HTTPS access.
2. Configure SSL Access in your production:

a. Log in to the Management Portal as a user with the **%HS_Administrator** role.

b. Navigate to **Health** > **Installer Wizard**.

c. Click **Configure SSL Access**.

d. In the **SSL Access** dialog box:

1. For **Enter SSL Port**, enter the web server port for InterSystems IRIS for Health.
2. Select **These Settings are Active**.
3. Click **Save**.

For more information about SSL access settings, see the reference documentation for WebServerURLPrefix in *System Administration*.

3. Log into the Terminal as a user with the **%HS_Administrator** role and run the following commands in the FHIR namespace:

- `do ##class(HS.HC.Util.Installer.OAuth2).ConfigureSampleOAuth()` — Creates an OAuth 2.0 authorization server, two OAuth 2.0 clients (one of which you will use to create access tokens), and a server registration for each of those clients.
  - The name of the client you will use for generating access tokens is **<instancename>.Client**. For example, if your InterSystems IRIS for Health instance name is I4H200, the name of your OIDC client would be **I4H200.Client**.
  - You can view the current list of OAuth 2.0 clients by navigating to **System** > **Security** > **OAuth 2.0 Client** > **Client Configurations**.
- `do ##class(HS.HC.Util.Installer.FHIR.OAuth2).AddOAuthToCapability()` — Adds authorization and access token endpoints to the conformance and/or capabilities statement for the FHIR resource repository associated with the current namespace.

4. Now you can obtain and use an access token in the FHIR Test Utility:

a. In the FHIR Test Utility, click **Get Access Token** in the left-hand menu.

b. For **Client**, enter the name of your OAuth 2.0 client (following the example above, **I4H200.Client**).

c. For **Scopes**, enter the desired scopes.

d. For **Endpoint**, choose an OIDC endpoint.

e. Click the **Get Access Token** button.

- You will be asked to provide authorization credentials: supply the InterSystems IRIS for Health username and password you used to log in to the Management Portal.
- After you confirm the desired scopes, the access token is created and stored in an InterSystems IRIS for Health global. You can use the token for requests made to the endpoint you selected above: simply enter the name of your OAuth 2.0 client in the **Client** field when you make the request.
- The expiration time of the token is determined by the **Access token interval** setting in the OAuth 2.0 server definition. To view or change this setting:
  - Navigate to **System** > **Security** > **OAuth 2.0 Authorization Server Configuration**.
  - Click the row for the registration for **<instancename>.Client**.
  - Click the **Intervals** tab and adjust the **Access token interval** setting, if necessary. Then click **Save**.

2.4 Exploring the Production with a Third-Party FHIR Client

You can also test the FHIR production as a FHIR server by connecting to it with a third-party client, such as cURL, SoapUI, Advanced REST Client, or Postman. For example, to perform a test search using the Postman application:

1. If you have not already done so, download and install Postman.

   Important: The use of Postman in this example does not constitute an endorsement of this product by InterSystems.

2. Start Postman.
3. Enter the URL to search for a patient on the FHIR server, which, for STU3, is of the form:

   `http://`*host*`:`*portnumber*`/csp/healthshare/`*namespace*`/fhir/stu3/Patient/1`

   where:

   - *host* — The host of the FHIR server.
   - *portnumber* — The port number for the FHIR server, if one is being specified.
   - *namespace* — The FHIR namespace on the FHIR server, such as FHIRNAMESPACE.

   This refers to patient #1, who is the patient you created in the previous section. The form of this URL is from the FHIR standard as specified at http://www.hl7.org/fhir/http.html#read.
4. Select **GET** for the HTTP method.
5. In the **Authorization** tab:

   a. Choose **Basic Auth**.

   b. Enter the username and password for the account you used to log in to the Management Portal to create the Patient resource.

   c. Click **Update Request**.

6. In the **Headers** tab, specify that the client is requesting FHIR content in JSON format by entering

   ```
   application/fhir+json
   ```

   as the value for the **Accept** header. To use XML format, change the string to

   ```
   application/fhir+xml
   ```

7. Select **Send**.

The response contains information about patient #1 in the format specified, either JSON or XML.

3

Elements of the FHIR Production

A FHIR production has several key elements:

- The FHIR Business Hosts
- The FHIR Web Applications
- The FHIR Resource Repository

3.1 The FHIR Business Hosts

When initially configured, the FHIR production contains the following business hosts for the versions of FHIR you selected during installation:

- Business Services:
  - **HS.FHIR.Service**.[FHIRVersion] — Accepts HS.Message.FHIR.Request messages from the FHIR web application and passes data to the business host specified by TargetConfigName (which **InstallFHIRServer** sets to HS.FHIR.Server.Process). Note that the web application accepts each REST connection and request to your production as a FHIR server, formulates it into an HS.Message.FHIR.Request message, and passes that to HS.FHIR.Service; for more information about the web application, see "The FHIR Web Applications.")
  - **HS.Test.Service** — Acts as a simple test service that receives message objects and routes them to the appropriate operation.
- Business Processes:
  - **HS.FHIR.Server.Process**.[FHIRVersion] — Performs data handling for your production as a FHIR server. Its actions depend upon either its own settings, the request's indicated interaction, or both. It operates on data in HS.Message.FHIR.Request format.
- Business Operations:
  - **HS.FHIR.Repository.Operations**.[FHIRVersion]— Present if you installed a FHIR resource repository: performs storage and retrieval operations for the repository.
  - **HS.FHIR.REST.Operation** — Initiates REST connections from your production as a FHIR client. It sends its requests to the endpoint that the ServiceName setting specifies.
  - **HS.Util.Trace.Operations** — Uses standard InterSystems tools to provide message monitoring functionality within a FHIR production.

To view these business hosts, go to the production configuration page (in the FHIR namespace, select **Interoperability** > **Configure** > **Production**). The production displays information about each business host in its description and for each of its settings.

You can also add pre-built business hosts to your production that transform FHIR into SDA and vice versa. For more information, see the section that corresponds to your FHIR version:

- Configuring Your Production for SDA-FHIR DSTU2 Conversions
- Configuring Your Productions for SDA-FHIR STU3 Conversions

3.2 The FHIR Web Applications

Any one of the web applications that were installed can be used to access the FHIR server. To view and change these applications, switch to your FHIR production namespace and go to **System Administration** > **Security** > **Applications** > **Web Applications**. The root URL of the FHIR web applications is `/csp/healthshare/`*`fhirnamespace`*`/fhir/`.

Important: Do not change settings for the top-level web application for the FHIR namespace, `/csp/healthshare/`*`fhirnamespace`*.

To view or change settings for any application, select its name. An **Edit** button is present on the resulting page, along with the settings that can be changed.

This page includes the following fields that are related to FHIR:

- **Namespace** — The value that you selected when running the setup method (in this documentation, FHIR).
- **Allowed Authentication Methods** — Password authentication is the default. For more information about password authentication, see the section "InterSystems Authentication" in the "Authentication" chapter of the *Security Administration Guide*.
- **Dispatch Class** — The class that accepts the CSP request from the FHIR client, formulates it into HS.Message.FHIR.Request format, and forwards it to HS.FHIR.Service.This class also receives the HS.Message.FHIR.Response back and places it into CSP format.

3.3 The FHIR Resource Repository

The FHIR resource repository in InterSystems IRIS for Health:

- Can store all resource types
- Can store in DSTU2, STU3, or both
- Supports the FHIR REST API
- Allows for the use of a custom alternate storage model
- Supports custom operations and search parameter definitions
- Supports the creation and deletion of temporary data repositories with session IDs

3.3.1 Bulk Loading Data into Resource Repository

If you have a large quantity of FHIR data that needs to be stored in IRIS for Health's FHIR resource repository, making individual REST calls to store the data can be inefficient. In these cases, you can use the Data Load utility to programmatically load resources and bundles directly into the repository, which is faster than sending the data to a FHIR production over HTTP. This data is stored into the FHIR resource repository by sending it to the local production's FHIR repository business operation.

FHIR data can be fed to the Data Load utility in two ways:

- By specifying an operating system directory with a batch of text files, each of which contains a FHIR resource or Bundle.
- By sending a stream object that contains FHIR data.

In both methods, the FHIR data can be individual resources, bundles or both, and can be expressed in JSON, XML, or both. The class for the Data Load utility is HS.HC.Util.FHIR.DataLoad.

3.3.1.1 Bulk Loading from Files

The most common use of the Data Load utility is to quickly store FHIR data from text files into the FHIR resource repository. For example, open source patient data generators often place the synthetic data in text files. With the Data Load utility, you use the InterSystems Terminal to specify the directory where these files are located using the **SubmitResourceFiles** method of HS.HC.Util.FHIR.DataLoad.

For example, suppose you have text files containing resources and bundles of FHIR STU3 data in a directory c:\synthetic_data. To load this data into the FHIR resource repository:

1. Open the InterSystems Terminal.
2. Change to the namespace that contains FHIR production with the repository where you want to load the data. For example, enter:

   ```
   ZN "fhirnamespace"
   ```

3. Enter:

   ```
   do ##class(HS.HC.Util.FHIR.DataLoad).SubmitResourceFiles("c:\synthetic_data", "STU3")
   ```

For additional parameters that control how many files are processed and whether statistics are generated, see the class reference for the **SubmitResourceFiles** method.

3.3.1.2 Bulk Loading from a Stream

While not a common use of the Data Load utility, your application could populate a stream object with FHIR data and store that data directly into the FHIR repository by passing it to the local production's FHIR repository business host. For more information, see the class reference for the **SubmitResourceStream** method of HS.HC.Util.FHIR.DataLoad.

3.3.2 Customizing FHIR Search Parameter Definitions

The FHIR resource repository provides a means for adding, removing, or modifying search parameter definitions which is tightly coupled with the default product implementation.

This procedure involves distinct items that have similar nomenclatures:

- "Storage classes" are individual classes for each FHIR resource type. Each class has an XData block that holds the search parameter definitions for that resource type.

- The default product class packages for the storage classes are HS.FHIR.vDSTU2.Repository.Resource and HS.FHIR.vSTU3.Repository.Resource.
- HS.FHIR.vDSTU2.Repository.Storage and HS.FHIR.vSTU3.Repository.Storage are the FHIR version-specific classes that act as the outermost layer of abstraction for the default product FHIR resource repository logic.

Both HS.FHIR.vDSTU2.Repository.Storage and HS.FHIR.vSTU3.Repository.Storage have a class parameter, CUSTOMSTORAGEPKG, whose value is empty by default. The parameter value determines the class package to use for customized storage classes for the respective storage class. When you override CUSTOMSTORAGEPKG, if you are storing resources from multiple FHIR versions in the same namespace, take care to avoid custom package name clashes between the FHIR versions.

FHIR Conformance and CapabilityStatement resources also hold relevant definitions for resource search parameters under `/f:<resourcetype>/f:rest[]/f:resource[]/f:searchParam[]`. The format of the definitions in the storage classes mirrors the format found in the HL7 FHIR distribution files for search parameter definitions. The definition of a search parameter is driven by the XPATH location within the relevant resource type of the indexed data.

Important: Definition of custom search parameters should be the subject of careful planning in advance.

If your repository contains a large amount of data, rebuilding search table definitions and search indices may take several hours.

3.3.2.1 Customizing DSTU2 Search Parameter Definitions

To customize DSTU2 search parameter definitions:

1. Stop the FHIR production.
2. Determine the name of a new class package that will contain customized versions of DSTU2 storage classes.
3. Create a custom class that extends HS.FHIR.vDSTU2.Repository.Storage, whose package name should be the one you chose above. In the custom class, add an override of class parameter CUSTOMSTORAGEPKG, whose value indicates the class package that you created above.
4. In your FHIR production, find the HS.FHIR.vDSTU2.Repository.Operations business operation (which is sometimes called HS.FHIR.Repository.Operations.DSTU2 when installed via an installer).
   - Set the StorageClass setting value to the name of your custom version of HS.FHIR.vSTU3.Repository.Storage.
   - Apply changes.
5. For each resource type that you want to customize, copy the corresponding HS.FHIR.vDSTU2.Repository.Resource.[resourceType] class into your custom class package.
6. In each copied class, edit the DSTU2SearchParameters XData block as needed to add, change, or remove search parameter definitions.
   - To remove a definition, remove its corresponding `<entry>` node in its entirety
   - To add a definition:
     a. Copy and paste the entire `<entry>` node for an existing definition in the XData block.
     b. Change the value attribute of the `name` element to the desired name (*parameterName*) of the new parameter: for example, `custom-search`.
     c. Change the value attribute of the `id` element to *resourceType-parameterName*: for example, `Condition-custom-search`.
     d. Change the value attribute of the `type` element to the appropriate search parameter type.
     e. Change the value attribute of the `xpath` element to the desired XPath expression.

7. Compile your custom classes.
8. Modify the main InterSystems DSTU2 Conformance resource to reflect any added or removed search parameter definitions. (If the only change was to the type or XPath definition of an existing search parameter, then no change to the Conformance resource is required.)
   a. Use a REST utility to read the **`InterSystemsFHIR`** Conformance resource, using URL parameters `_format=xml` and `_pretty=1` as follows:

      ```
      GET <baseURL>/Conformance/InterSystemsFHIR?_format=xml&_pretty=1
      ```

   b. Copy the retrieved Conformance resource content into a text editor.
   c. In the resource content, find the search parameter definitions for the resource type you are updating, under `/f:Conformance/f:rest[]/f:resource[]/f:searchParam[]`.
   d. Edit the search parameter definitions as needed.
   e. Use a REST utility to update the Conformance resource:

      ```
      PUT [baseURL]/Conformance/InterSystemsFHIR
      ```

9. Open a Terminal session and log in as a user with the **`%HS_Administrator`** role.
   a. Switch to your FHIR production.
   b. Rebuild the search table for the resource type whose search parameters you changed:

      ```
      Set status = ##class(HS.FHIR.vDSTU2.SearchTableBuilder).GenSearchTables("InterSystemsFHIR",
      <resource_type>)
      ```

      The error message "ERROR #5202: Nothing to compile" can safely be ignored.
   c. Rebuild the indices for the stored resources for that resource type, where `<repository_storage_class>` is the subclass of HS.FHIR.vDSTU2.Repository.Storage that you created above:

      ```
      Set status = ##class(HS.FHIR.vDSTU2.Repository.Resource.Abstract).RebuildSearchIndices(0,
      <resource_type>, <repository_storage_class>)
      ```

10. Start the FHIR production.
11. Test your customized search parameters by performing a search with them.

3.3.2.2 Customizing STU3 Search Parameter Definitions

To customize STU3 search parameter definitions:

1. Stop your FHIR production.
2. Determine the name of a new class package that will contain customized versions of STU3 storage classes.
3. Create a custom class that extends HS.FHIR.vSTU3.Repository.Storage, whose package name should be the one you chose above. In the custom class, add an override of class parameter CUSTOMSTORAGEPKG, whose value indicates the class package you created above.
4. In your FHIR production, find the HS.FHIR.vSTU3.Repository.Operations business operation (which is sometimes called HS.FHIR.Repository.Operations.STU3 when installed via an installer).
   - Set the StorageClass setting value to the name of your custom version of HS.FHIR.vSTU3.Repository.Storage.
   - Apply changes.

5. For each resource type that you want to customize, copy the corresponding HS.FHIR.vSTU3.Repository.Resource.[resourceType] class to your custom class package.
6. In each copied class, edit the SearchParameters XData block as needed to add, change, or remove search parameter definitions.
   - To remove a definition, remove its corresponding `<entry>` node in its entirety
   - To add a definition:
     a. Copy and paste the entire `<entry>` node for an existing definition in the XData block.
     b. Change the value attribute of the `name` element to the desired name (*parameterName*) of the new parameter: for example, `custom-search`.
     c. Change the value attribute of the `id` element to *resourceType-parameterName*: for example, `Condition-custom-search`.
     d. Change the value attribute of the `type` element to the appropriate search parameter type.
     e. Change the value attribute of the `xpath` element to the desired XPath expression.
7. Compile your custom classes
8. Modify the main InterSystems STU3 CapabilityStatement resource to reflect any added or removed search parameter definitions. (If the only change was to the type or XPath definition of an existing search parameter, then no change to the CapabilityStatement resource is required.)
   a. Use a REST utility to read the **`InterSystemsFHIR`** CapabilityStatement resource, using URL parameters `_format=xml` and `_pretty=1` as follows:

      ```
      GET <baseURL>/CapabilityStatement/InterSystemsFHIR?_format=xml&_pretty=1
      ```

   b. Copy the retrieved CapabilityStatement resource content into a text editor.
   c. In the resource content, find the search parameter definitions for the resource type you are updating, under `/f:CapabilityStatement/f:rest[]/f:resource[]/f:searchParam[]`.
   d. Edit the search parameter definitions as needed.
   e. Use a REST utility to update the CapabilityStatement resource:

      ```
      PUT [baseURL]/CapabilityStatement/InterSystemsFHIR
      ```

9. Open a Terminal session and log in as a user with the **`%HS_Administrator`** role.
   a. Switch to your FHIR production.
   b. Rebuild the search table for the resource type whose search parameters you changed:

      ```
      Set status = ##class(HS.FHIR.vSTU3.SearchTableBuilder).GenSearchTables("InterSystemsFHIR",
      <resource_type>)
      ```

      The error message “ERROR #5202: Nothing to compile” can safely be ignored.
   c. Rebuild the indices for the stored resources for that resource type, where `<repository_storage_class>` is the subclass of HS.FHIR.vSTU3.Repository.Storage that you created above:

      ```
      Set status = ##class(HS.FHIR.vSTU3.Repository.Resource.Abstract).RebuildSearchIndices(0,
      <resource_type>, <repository_storage_class>)
      ```

10. Start the FHIR production.
11. Test your customized search parameters by performing a search with them.

3.3.3 Creating and Deleting Temporary Data Repositories with Session IDs

You can enable the creation and deletion of temporary FHIR data repositories by associating session IDs with repository operations.

To enable the use of session IDs:

1. Stop your FHIR production.
2. In your FHIR production, change the **`SessionMode`** setting of HS.FHIR.Service.[FHIRVersion] to one of the following values:
   - Parameter, which requires that each request be accompanied by a server-generated session ID
   - OAuth, which uses an OAuth token as the basis for a session ID

   See the class documentation for HS.FHIR.Service.DSTU2 or HS.FHIR.Service.STU3 for more information.
3. Extend the HS.FHIR.[FHIRVersion].Repository.Operations class and, in the custom class, set the value of the ApplySessionIdToResources property to 1.
4. Compile your custom class.
5. Add your custom class to your production as a business operation.
6. Enable the business operation for your custom class and apply changes.
7. Change the target of HS.FHIR.Server.Process.[FHIRVersion] so that it points to your custom class.
8. Disable the standard Repository.Operations business operation and apply changes. You can remove it later if you wish.
9. Start the FHIR production.

To delete sessions, you can use **HS.FHIR.SessionData.FHIRSession:DeleteAllSessions()** or **HS.FHIR.SessionData.FHIRSession:DeleteExpiredSessions()**. You have the option of running your choice of these methods on an ad hoc basis in Terminal in your FHIR namespace, or you can set up a scheduled task to run either one. For more information on scheduled tasks, see the "Using the Task Manager" section of the "Managing InterSystems IRIS" chapter of the *System Administration Guide*.

4

The FHIR Server and Client

4.1 The FHIR Client

The InterSystems IRIS for Health FHIR client:

- Processes both individual resources and bundles.
- Supports both XML and JSON formats.
- Uses the HS.FHIR.REST.Operation business operation as a principal component to:
  - Send REST FHIR requests to and receive REST FHIR responses from the FHIR server.
  - Communicate with client code that formulates requests and receive responses using the **HS.Message.FHIR.Request** and **HS.Message.FHIR.Response** message classes, which are described in the "FHIR Message Handling" section.
- Can be enabled to support data transformations between FHIR DSTU2 format and SDA.
- Can be enabled to support data transformations between FHIR STU3 format and SDA.

When a FHIR client formulates a request and sends it to a FHIR server (whether an InterSystems IRIS for Health FHIR server or that of another organization), the message flow is:

1. The client code formulates a HS.Message.FHIR.Request message.
2. The client sends that message to the HS.FHIR.REST.Operation business operation.
3. HS.FHIR.REST.Operation converts the message into a FHIR REST request and sends it to a FHIR server as an HTTP request message.
4. The FHIR server returns an HTTP response message, which goes back to HS.FHIR.REST.Operation.
5. HS.FHIR.REST.Operation converts the message into an HS.Message.FHIR.Response message and returns it to the client code.

4.1.1 HS.FHIR.REST.Operation Routing

For a FHIR client application, the HS.FHIR.REST.Operation business operation specifies how each message is processed and where it is sent. The ServiceName property specifies the message destination, using HTTP entries from the Service Registry. For more information about the Service Registry, see the "Managing the Service Registry" chapter in the *Registry Guide for InterSystems IRIS for Health and Health Connect*.

HS.FHIR.REST.Operation evaluates how it routes a message in the following order:

1. If the HS.Message.FHIR.Request message specifies a ServiceName in an AdditionalInfoItem, then the message is directed to the location of that service.
2. If there is no ServiceName specified in an AdditionalInfoItem, the ServiceName property in the operation specifies the destination.

4.1.2 Configuring Credentials for a FHIR Client Running In IRIS for Health

A FHIR client running in IRIS for Health must have appropriate credentials so the FHIR server can authenticate it. To configure client credentials:

1. On the **Credentials** page (**Interoperability** > **Configure** > **Credentials**), specify credentials that match those on the FHIR server.

   Note: Because the **Install** method for the FHIR production configures both a FHIR server and a FHIR client, a sample system uses a single set of credentials for both entities. Hence, on such a system, no action is required.

2. In the Management Portal, in the client's namespace, add an HTTP service to the Service Registry or modify the value of the `HealthShareFHIR.Service` entry; follow the instructions in the "Adding or Modifying a Service" section in the "Managing the Service Registry" chapter of the *Registry Guide for Health Connect and InterSystems IRIS for Health*. The key settings for the service are:
   - *Name* – Required. The name of the service. To continue using `HealthShareFHIR.Service`, select it for editing; to create a new service, click **Add Service** and specify its name.
   - *Host* – Required. The name of the machine on which the FHIR server is running.
   - *Port* – Optional. The port number on which to connect to the FHIR server.
   - *URL* – Any string, such as a CSP application name, that appears after the host (or port number) in the web address.
   - *HTTPCredentialsConfig* — The set of credentials that the client uses to authenticate itself to the server.

The FHIR client now has credentials and provides them to the FHIR server for authentication.

4.2 The FHIR Server

InterSystems IRIS for Health supports the ability to create a FHIR server that uses the built-in FHIR functionality.

An InterSystems IRIS for Health FHIR server:

- Communicates with FHIR clients via HTTP
- Handles a broad set of FHIR interactions for all resource types
- Evaluates access tokens and honors them per interaction and resource type

- Supports storage of all FHIR resource types in a resource repository
- Uses the **HS.Message.FHIR.Request** and **HS.Message.FHIR.Response** message classes for communications among its business hosts. For more information about these messages, see the "FHIR Message Handling" section.
- Can be enabled to support data transformations between FHIR DSTU2 format and SDA
- Can be enabled to support data transformations between FHIR STU3 format and SDA

A FHIR server configured with the installed defaults processes a FHIR request message as follows. Each business host described below has versions for DSTU2, STU3, or both, depending on what was installed.

1. It accepts connections from a FHIR client through the HS.FHIR.Service business service.
2. The HS.FHIR.Service for the chosen FHIR version receives a FHIR request from a FHIR client.
3. HS.FHIR.Server.Process performs routing for the request based on its **`TargetConfigName`** setting. By default, this is set to HS.FHIR.Repository.Operations or, if you did not install a repository for the current FHIR version, to HS.FHIR.REST.Operation.
   - The **`CustomProcessor`** setting of HS.FHIR.Service is empty by default. If you want to do custom preprocessing, supply the name of the class here.
4. If the request was routed to HS.FHIR.Repository.Operations:
   - The resource type and interaction are verified against the conformance or capabilities statement.
   - The request is forwarded to the storage class named in the **`StorageClass`** setting, which by default is HS.FHIR.vSTU3.Repository.Storage. You may use a custom class here.
   - The storage class stores data in accordance with its parameters and returns a response to HS.FHIR.Repository.Operations with an acknowledgement of success or error; it may also include the appropriate return payload, such as the results of a search or other query.
   - HS.FHIR.Repository.Operations adds appropriate HTTP response headers and forwards the response to HS.FHIR.Server.Process.
   - For Bundles of type "transaction", resource IDs are assigned as described in "Resource ID Assignment".
5. If the request was routed to HS.FHIR.REST.Operation:
   - The request is converted into an instance of %Net.HttpRequest.
   - It is sent to the endpoint corresponding to the **`ServiceName`** specified in the FHIR request's AdditionalInfo property. If that property is empty or missing, it is sent to the service specified in the **`ServiceName`** setting of HS.FHIR.REST.Operation.
   - The corresponding response is converted into an object of type HS.Message.FHIR.Response and forwarded to HS.FHIR.Server.Process.
6. HS.FHIR.Server.Process receives the response and returns it to HS.FHIR.Service, which then returns it to the FHIR client.

4.2.1 Supported FHIR Interactions

4.2.1.1 Supported DSTU2 Interactions

For a full list of interactions, see the FHIR specification for interactions.

- create: Fully supported

- read: The `_summary` parameter and conditional read are not supported.
- vread: The `_summary` parameter and conditional read are not supported.
- update: Fully supported
- delete: Fully supported
- history: Type-level and system-level history, `_count` parameter, and paging are not supported.
- search: Supported, with the following exceptions:
  - Composite search parameter type
  - Chaining and reverse chaining search parameters
  - `token` parameter with system value only (`[parameter]=[system]`)
  - Compartment search on all resources for a specified compartment (`GET [base]/[Compartment]/[id]/{*?[parameters]{&_format;=[mime-type]}}`).
  - Supported modifiers are limited to: `exact` (strings), `contains` (strings), `above` (uri), `below` (uri), and `type` (reference).
  - Common search parameters: `_text`, `_list`, `_query`, `_filter`
  - Search result parameters: `_revinclude`, `_summary`, `_elements`, `_contained`, `_containedType`
  - Prefixes: `sa`, `eb`, `ap`
- batch/transaction: Document, message, and collection Bundle types are not supported
- operations: `$validate` and `$everything` are supported as follows:
  - The validation modes (create, update, delete) are not supported.
  - Validation by profile is not supported.
  - When a FHIR `$validate` request includes a resource payload, the resource may be enclosed in a Parameters resource.
  - Validation by resource type plus resource ID to validate an existing stored resource is supported.
- conformance: The REST interaction itself is fully supported.

4.2.1.2 Supported STU3 Interactions

For a full list of interactions, see https://www.hl7.org/fhir/http.html#interactions.

- create: Fully supported
- read: The `_summary` parameter and conditional read are not supported.
- vread: The `_summary` parameter and conditional read are not supported.
- update: Fully supported
- patch: Supported with the following exceptions:
  - FHIRPath Patch operations are not supported
  - The HTTP `Accept` header cannot be `application/xml-patch+xml`

  The entire Patch request is rejected:
  - When the existing resource includes a narrative

  - When the request attempts to patch the id value of the resource

  Attempts to patch the metadata `Meta.versionID` and `Meta.lastUpdated` are ignored.

- delete: Fully supported
- history: Type-level and system-level history, `_count` and `_at` parameters, and paging are not supported.
- search: Supported, with the following exceptions:
  - Cannot search across multiple resource types
  - Composite search parameter type
  - Chained `_has` parameters. Reverse chaining "one level down" is supported via the `_has` parameter.
  - token parameter with system value only (`[parameter]=[system]`)
  - Compartment search on all resources (`GET [base]/[Compartment]/[id]/{*?[parameters]{&_format;=[mime-type]}}`)
  - Supported modifiers are limited to: `exact` (strings), `contains` (strings), `above` (uri), `below` (uri), and `type` (reference).
  - Common search parameters: `_text`, `_list`, `_query`, `_filter`, `_type`
  - Search result parameters: `_revinclude`, `_summary`, `_elements`, `_contained`, `_containedType`
  - Prefixes: sa, eb, ap
- batch/transaction: Conditional references as well as document, message, and collection Bundle types are not supported.
- operations: `$validate` and `$everything` are supported as follows:
  - The validation modes (create, update, delete) are not supported.
  - Validation by profile is not supported.
  - When a FHIR `$validate` request includes a resource payload, the resource may be enclosed in a Parameters resource.
  - Validation by resource type plus resource ID to validate an existing stored resource is supported.
- capabilities: The REST interaction itself is fully supported.

4.2.2 Resource ID Assignment

For Bundles of type `transaction`, resource IDs are assigned as follows:

- For `create` interactions, a new sequential number is assigned, unless conditional `create` is specified and a matching resource is found.
- For `update` interactions:
  - A resource ID in the incoming Bundle is considered a UUID if any of these is true of the containing Bundle entry:
    - The value of fullURL is a UUID prefixed with `urn:uuid` (for example: `urn:uuid:e12edae2-8892-11e8-bce1-7831c1b7edac`).
    - The value of fullURL is the resource type to which an unprefixed UUID is appended (for example: `Encounter/e12edae2-8892-11e8-bce1-7831c1b7edac`).
    - The value of the resource ID is an unprefixed UUID (for example: `e12edae2-8892-11e8-bce1-7831c1b7edac`).

    - The value of the entry request URL is a prefixed UUID (for example: `urn:uuid:e12edae2-8892-11e8-bce1-7831c1b7edac`).
    - The value of the entry request URL is the resource type to which an unprefixed UUID is appended (for example: `Encounter/e12edae2-8892-11e8-bce1-7831c1b7edac`).

– If the value of a resource ID is not empty and not a UUID, that resource ID is used for the update interaction on the individual entry.
– If the value of a resource ID is a UUID, a new sequential number is generated and replaces the existing resource ID. All references to that resource via the UUID elsewhere in the Bundle are updated with the new sequential number. These replacements occur before the transaction is submitted for processing to the FHIR resource repository.

4.2.3 Hosting Multiple FHIR Servers on a Single Instance

InterSystems IRIS for Health supports hosting multiple FHIR servers on a single instance. Because the servers are in Foundation namespaces, each has its own separate registry. When configuring InterSystems IRIS for Health to support multiple servers, you must specify an individual namespace for each server.

5

FHIR Message Handling

Internal communications among the FHIR production's business hosts use two message classes:

- The HS.Message.FHIR.Request request message class.
- The HS.Message.FHIR.Response response message class.

As both a FHIR client and FHIR server, InterSystems IRIS for Health uses the HS.Message.FHIR.Request and HS.Message.FHIR.Response messages to pass requests and responses among business hosts; these requests and responses are also the basis for REST HTTP messages to external FHIR clients or servers.

5.1 About Properties of HS.Message.FHIR.Request and HS.Message.FHIR.Response

HS.Message.FHIR.Request and **HS.Message.FHIR.Response** inherit from **HS.Message.FHIR.Base** and **HS.Util.EnsResponse**. Among their notable properties are:

- BaseURL – On the server side, the URL of the web application from which the message originated, such as `/csp/healthshare/fhirnamespace/fhir/dstu2`. It is not used on the client side.
- ContentType – For response messages, specifies the format of the payload, if any exists. For request messages that carry a payload, specifies the format of the payload; for read-type request messages, such as "search" messages, specifies the requested format of the data in the response. Possible values include:
  - `application/json+fhir` and `application/xml+fhir` (DSTU2)
  - `application/fhir+json` and `application/fhir+xml` (STU3)
- AdditionalInfo – Additional information required for a particular request. It is most commonly used to specify the service endpoint to which a request is sent. For more information, see "The AdditionalInfo Property," below.
- Interaction – The FHIR interaction that the request message specifies, such as "create" or "search". This action occurs in relation to the FHIR resource type for the request (specified by the Type property). The value of this attribute determines the HTTP method (also known as the HTTP verb) that the request uses, such as POST or GET.
- Parameters – Any number of URL parameters that are part of the REST request, such as search parameters or data format.
- Payload – The FHIR resources, in XML or JSON
- Type – The type of resource included in the request

- Id – The ID of the resource included in the request
- Vid – The version ID of the resource included in the request.
- Headers – The HTTP headers included with the request or response.

5.1.1 The AdditionalInfo Property

On the client side, the AdditionalInfo property may specify the value of the `ServiceName` key, which identifies the name of the Service Registry entry that contains the desired endpoint.

On the server side, there can be several AdditionalInfoItem elements:

- `ClientAddr` — The IP address of the FHIR client
- `Secure` — Whether or not HTTPS is in use, where 1 means TRUE and 0 means FALSE
- `ConformanceId` – Resource ID of the DSTU2 Conformance resource applicable to the endpoint to which the request was submitted
- `CapabilityStatementId` – Resource ID of the STU3 CapabilityStatement resource applicable to the endpoint to which the request was submitted
- `JSONWebToken` – JSON string version of the OAuth access token, if any
- `USER:UserID` – The authenticated user's user ID
- `USER:Roles` – The authorized roles assigned to the user

5.2 A Sample Request Message

The following code provides a sample request message for FHIR STU3:

```
 // Instantiate a file stream object
 Set tFHIRStream = ##class(%Stream.FileCharacter).%New()

 // Set the source for the file stream to an XML file of our choosing
 Set tFHIRStream.Filename = "SampleFHIRPatientResource.xml"

 // Instantiate the FHIR request object
 Set tRequest = ##class(HS.Message.FHIR.Request).%New()

 // For a create, no resource id is specified. If this were an update, then the resource id must be
specified.
 Set tRequest.Interaction = "create"

 // The HTTP PREFER header may be set to trigger the server
 // to echo back the resource that was saved to the database.
 Set tSC = tRequest.Headers.SetAt("return=representation","PREFER") Quit:$$$ISERR(tSC)

 // The FHIR payload is a stream property in the message.
 // In this example, the stream is derived from a file
 // that contains a FHIR resource in XML format.
 Set tSC = tRequest.Payload.CopyFrom(tFHIRStream) Quit:$$$ISERR(tSC)

 // This request is for a Patient resource.
 Set tRequest.Type = "Patient"

 // Options are json or xml. Here, it is set
 // to xml, which reflects the content of the sample resource.
 Set tRequest.ContentType = "application/fhir+xml"

 // If you wish to override the setting
 // in the FHIR operation, set the value of ServiceName.
 Do tRequest.AdditionalInfo.SetAt("HealthShareFHIR.Service", "MyService")
```

```
 // Send the request to the FHIR REST operation.
 Set tSC = ..SendRequestSync("HS.FHIR.REST.Operation", tRequest, .tResponse) Quit:$$$ISERR(tSC)
```

5.3 Handling a Response Message

```
// Check the HTTP Status
 Set tHttpStatus = tResponse.Status

 // Get the returned payload if any
 If tResponse.Payload.Size {
    Write !,"Returned Payload is: "_tResponse.Payload.Read($$$MaxLocalLength),!
 } Else {
    Write !,"No Payload returned in response",!
 }

 If ##class(HS.FHIR.Helper).HttpSuccess(tResponse.Status) {
        Write !,"Request was successful",!
    } Else {
        Write !,"Request failed",!
    }
```

6

The FHIR Object Model

InterSystems IRIS for Health supports the full FHIR object model for both DSTU2 and STU3. This model consists of *primitive data types*, *complex data types*, and *resources*. For information about primitive data types and complex data types, see https://www.hl7.org/fhir/datatypes.html; for information about resources, see http://hl7.org/fhir/resourcelist.html.

The InterSystems IRIS for Health implementation adds structure to the FHIR object model by organizing all its objects into one of three packages:

- HS.FHIR.[FHIRVersion].Model.Datatype — Typical data type primitives, such as string and integer.
- HS.FHIR.[FHIRVersion].Model.Resource — Resources and data types that are specific to them for the given FHIR version. This package contains classes that represent FHIR resources within your production. For each class that uses its own specific data types, there is also a package with the same name as the class; members of that package are any of the class-specific data types. For example, for the FHIR `Bundle` resource, there is a class, HS.FHIR.Model.Resource.Bundle, and a package with the same name that has five classes within it, one for each of the data types that are specific to the `Bundle` resource:
  - HS.FHIR.[FHIRVersion].Model.Resource.Bundle.Entry
  - HS.FHIR.[FHIRVersion].Model.Resource.Bundle.Link
  - HS.FHIR.[FHIRVersion].Model.Resource.Bundle.Request
  - HS.FHIR.[FHIRVersion].Model.Resource.Bundle.Response
  - HS.FHIR.[FHIRVersion].Model.Resource.Bundle.Search
- HS.FHIR.[FHIRVersion].Model.Base — All other classes. This includes:
  - Complex data types that are used by multiple classes. This includes data types such as Address, Coding, and HumanName.
  - The InterSystems IRIS for Health abstract base class.

Note: InterSystems IRIS for Health implements all the elements of the FHIR object model. For DSTU2, it includes helper methods for setting the values of various properties.

InterSystems IRIS for Health includes a second object model that is optimized for transformations. These classes can be found in HS.FHIR.DTL.[FHIRVersion].Model.*. You may find it helpful to use these classes when you are doing transformations, but they should not be used for any other purpose. Note also that they support JSON only.

7

Using and Customizing SDA-FHIR DSTU2 Data Transformations

Important: To complete the steps in the sections of this chapter, you must already have created a Foundation namespace. An existing FHIR namespace will serve this purpose, but if you want to create a separate Foundation namespace, see the "Installing and Activating a Foundation Production" chapter of the *InterSystems IRIS for Health Installation Guide*.

This chapter includes the following topics:

InterSystems IRIS for Health supports transformations between SDA objects and FHIR DSTU2 resources using the data transformation language (DTL). This mapping can provide useful functionality in many different use cases, including:

- Taking content from an SDA-aware system and providing it to a FHIR DSTU2 system.
- Taking content from an SDA-aware system and storing it in a FHIR DSTU2 repository.
- Taking content from multiple SDA-aware systems and normalizing it for use or storage in a FHIR DSTU2 system.
- Taking content from a FHIR DSTU2 system and providing it to an SDA-aware system.

Because DTL provides one-to-one mapping and certain FHIR resources do not always map one-to-one to SDA objects, InterSystems IRIS for Health provides business processes that perform more complex transformations. For example, InterSystems IRIS for Health transforms an SDA Medication to the FHIR DSTU2 `MedicationDispense`, `MedicationOrdered`, and `Medication` resources. This is true for the FHIR DSTU2 `DiagnosticReport`, `Medication`, `Vaccination` resources, among others.

Because these transformations are written in the Data Transformation Language (DTL), members of your organization can easily customize them using the DTL editor, which is a visual interface (rather than by having to write code). For more information about the DTL editor, see the *Developing DTL Transformations* guide.

7.1 Configuring Your Production for SDA-FHIR DSTU2 Conversions

To configure an InterSystems IRIS for Health FHIR production for SDA-FHIR DSTU2 conversions, the process is:

1. Log into the Terminal with a role that has the **%Admin_Manage:Use** privilege.
2. Go to any Foundation namespace:

   ```
   zn "FHIRNAMESPACE"
   ```

3. Run the **Add** method of the HS.HC.Util.Installer.Kit.FHIR.ToSDA class, specifying the name of the FHIR DSTU2 namespace for your production. For example, with the FHIRNAMESPACE namespace, the call is:

   ```
   do ##class(HS.HC.Util.Installer.Kit.FHIR.ToSDA).Add(,"FHIRNAMESPACE","DSTU2")
   ```

   For more information, see the section "About HS.HC.Util.Installer.Kit.FHIR.ToSDA."
4. Run the **Add** method of the HS.HC.Util.Installer.Kit.FHIR.FromSDA class, specifying the name of the FHIR DSTU2 namespace for your production. For example, with the FHIR namespace, the call is:

   ```
   do ##class(HS.HC.Util.Installer.Kit.FHIR.FromSDA).Add(,"FHIRNAMESPACE","DSTU2")
   ```

   For more information, see the section "About HS.HC.Util.Installer.Kit.FHIR.FromSDA."

   Note: Because the HS.HC.Util.Installer.Kit.FHIR.FromSDA class's **Add** method has run after the HS.HC.Util.Installer.Kit.FHIR.ToSDA class's **Add** method, the lookup tables already exist.

7.1.1 About HS.HC.Util.Installer.Kit.FHIR.ToSDA

The **Add** method of the HS.HC.Util.Installer.Kit.FHIR.ToSDA class establishes the infrastructure to support converting data from FHIR DSTU2 format to SDA format. This includes:

- Adding a new business service, HS.FHIR.ToSDAService.DSTU2. This service accepts FHIR DSTU2 input.
- Adding a new business process, HS.FHIR.ToSDA.DTL.Process.DSTU2, which accepts input from HS.FHIR.ToSDAService.DSTU2 and transforms it to SDA output.
- Creating DTL lookup tables. These provide the mappings for specific field values, including those where there is not a one-to-one mapping from FHIR DSTU2 to SDA.

Important: Initially, the TargetConfigName setting of HS.FHIR.ToSDA.DTL.Process.DSTU2 does not have a value. Set it to the desired value.

7.1.2 About HS.HC.Util.Installer.Kit.FHIR.FromSDA

The **Add** method of the HS.HC.Util.Installer.Kit.FHIR.FromSDA class establishes the infrastructure to support converting data from SDA format to FHIR DSTU2 format. This includes:

- Adding a new business service, EnsLib.File.PassthroughService. This business host accepts a file containing SDA content in XML format.
- Adding a new business process, HS.FHIR.FromSDA.DTL.Transaction.Process. This business host transforms SDA input to FHIR DSTU2 output.

- Creating DTL lookup tables. These provide the mappings for specific field values, including those where there is not a one-to-one mapping from SDA to FHIR DSTU2.

Important: Set the value of the File Path setting of the EnsLib.File.PassthroughService business service to specify where it receives its SDA input. Select any directory that you wish to choose for SDA file input.

Set the value of the TargetConfigName setting of HS.FHIR.FromSDA.DTL.Transaction.Process to specify where it should send its FHIR output.

7.2 SDA-FHIR DSTU2 Business Hosts

The setup methods create the following business hosts:

- HS.FHIR.ToSDAService.DSTU2
- HS.FHIR.ToSDA.DTL.Process.DSTU2
- EnsLib.File.PassthroughService
- HS.FHIR.FromSDA.DTL.Transaction.Process

7.2.1 HS.FHIR.ToSDAService.DSTU2

This service accepts FHIR input that is sent to the endpoint `host:port/csp/healthshare/fhirnamespace/fhirtosda3/dstu2`. It routes that input to HS.FHIR.ToSDA.DTL.Process.DSTU2.

7.2.2 HS.FHIR.ToSDA.DTL.Process.DSTU2

The HS.FHIR.ToSDA.DTL.Process.DSTU2 business process:

- Accepts a FHIR DSTU2 resource or bundle as input.
- Converts the FHIR DSTU2 content to an SDA container.
- Forwards the container to the business host specified by the TargetConfigName setting.
- Receives the response from the business host.
- Returns a FHIR DSTU2 response (based on what it received) to the business host that originally called it.

7.2.3 EnsLib.File.PassthroughService

This is a typical file service, with relevant settings that include:

- File Path — The directory on disk from which it retrieves its file input, which is part of the **Basic Settings**. You must specify a value for this property, as it does not have one by default.
- TargetConfigNames — The business host to which it sends its output, which is also part of the **Basic Settings**. The setup method specifies HS.FHIR.FromSDA.DTL.Transaction.Process as the value of this property.

The output of this business service is content in Ens.StreamContainer format.

7.2.4 HS.FHIR.FromSDA.DTL.Transaction.Process

The HS.FHIR.FromSDA.DTL.Process business process:

- Accepts an SDA container as input and loops through each contained object.
- Converts the SDA container to FHIR DSTU2 content, in the form of a FHIR `Bundle` resource.
- Forwards the FHIR DSTU2 content to the business host specified by the TargetConfigName setting.
- Receives a response from the business host.
- Returns a response (based on what it received) to the business host that originally called it.

The following properties are relevant to the SDA-to-FHIR DSTU2 conversion:

- TargetConfigName — The destination to which HS.FHIR.FromSDA.DTL.Transaction.Process sends its output. This is another business host in this production. This property has no initial value.
- TransmissionMode — How the business process transmits the FHIR DSTU2 bundle for further processing:
  - transaction — The default; the business process sends the bundle of resources in a single interaction and the processing succeeds or fails for the whole of the bundle; if processing any single resource fails, processing for the other resources (and the entire bundle) stops.
  - batch — The business process sends the bundle of resources in a single interaction and each resource is processed independently; if processing any single resource fails, processing for the other resources still proceeds. (This is analogous to the manner in which InterSystems IRIS for Health processes an SDA container.)
  - individual — The business process sends each resource from the bundle separately as its own interaction.
- CustomDTLPackage — The class package from which the business process loads any custom data transformations. This package contains only the custom data transformations; if a transformation is not in the package, the business process uses the standard version. You must specify this package before creating any custom data transformations.
- FHIRFormat — Whether the content is in XML or JSON format.
- FormatFHIROutput — Whether or not content is formatted for readability. If selected, this setting has a performance impact, and as such should be enabled only during development and testing.

7.3 A Sample Conversion from SDA to FHIR DSTU2

To explain how InterSystems IRIS for Health transforms data, this section walks through a sample of converting sample data from SDA format to FHIR format.

To begin the process, first perform the following setup tasks:

1. Set the value of the TargetConfigName setting of HS.FHIR.FromSDA.DTL.Transaction.Process. For demonstration purposes, use HS.FHIR.Repository.Operations; this simply stores the content converted SDA to FHIR DSTU2 in the resource repository.
2. Set the value of the File Path setting of the EnsLib.File.PassthroughService business service.

The process to convert SDA to FHIR DSTU2 is as follows:

1. Place an XML-formatted SDA file in the input directory for EnsLib.File.PassthroughService (as specified by the File Path property). This begins processing.

2. EnsLib.File.PassthroughService places the SDA in an Ens.StreamContainer instance and sends it to HS.FHIR.FromSDA.DTL.Transaction.Process.
3. HS.FHIR.FromSDA.DTL.Transaction.Process processes the SDA:
   a. It converts the SDA container to a FHIR DSTU2 Bundle and converts the supported objects into FHIR resources. It places each FHIR resource in an entry in the bundle, along with a request property, which specifies the action associated with the resource when the bundle is processed.
   b. It sends the FHIR Bundle to HS.FHIR.vDSTU2.Repository.Operations in an HS.Message.FHIR.Request message.
   c. If the original SDA container had any SDA objects for which there were no transformations available, the HS.FHIR.FromSDA.DTL.Transaction.Process generates warnings for each object. These are visible in the production's **Message Viewer** page (**Interoperability** > **View** > **Messages**) and in its **Event Log** (**Interoperability** > **View** > **Event Log**).
4. HS.FHIR.vDSTU2.Repository.Operations returns an HS.Message.FHIR.Response message to HS.FHIR.FromSDA.DTL.Transaction.Process, reporting on the success or failure of the interaction. (Each Bundle entry in the request has a corresponding entry in the response.)
5. HS.FHIR.FromSDA.DTL.Transaction.Process returns a status of success or failure.

Looking in more detail at the conversion process, the content of the HS.Message.FHIR.Request specifies:

- A Type value of `Bundle`, which specifies that FHIR DSTU2 processing occurs for a group of FHIR resources. (The setting for this value is near the top of the content tab for the message in the message trace.)
- An Interaction parameter, whose value is as specified in HS.FHIR.FromSDA.DTL.Transaction.Process.
- If TransmissionMode is set to batch or transaction, a Bundle whose `type` value is batch or transaction.
- If TransmissionMode is set to individual, a set of requests each one of which contains a single resource.

Similarly, the content of the HS.Message.FHIR.Response includes:

- A Type value of `Bundle`.
- Within the `Bundle`:
  - If TransmissionMode was set to batch, a `type` value of "batch-response". Each entry in the bundle includes status information about success of the interaction for the resource and any relevant location information regarding the resource (such as, in this example, the location of the resource within the resource repository).
  - If TransmissionMode was set to transaction, a `type` value of "transaction-response". Each entry in the bundle includes status information about success of the interaction for the resource and any relevant location information regarding the resource (such as, in this example, the location of the resource within the resource repository).
  - If TransmissionMode was set to individual, the response indicates the success or failure of that individual request.
  - The data of each resource.

7.3.1 Information from the Message Trace

To see the details of the process, view the message trace for processing the SDA container. To see more details about messages, for HS.FHIR.FromSDA.DTL.Transaction.Process set the value of the TraceOperations property to `*FULL*`.

With trace operations on, the message trace for transforming SDA to FHIR DSTU2 now includes far more detail. It is now visible that, during processing, HS.FHIR.FromSDA.DTL.Transaction.Process creates and modifies a working copy of the SDA. The working copy of the FHIR output temporarily uses contained resources; the final output does not include contained resources.

7.4 About the SDA-FHIR DSTU2 Data Lookup Tables

Each transformation has access to the FromSDA and ToSDA data lookup tables, which specify how particular values are mapped between SDA and FHIR DSTU2. A data lookup table contains a list of key-value pairs that may:

- Convert codings from one data format to another, such as between single-letter codes for SDA to natural language words for FHIR DSTU2
- Standardize or normalize multiple values, such as between multiple, equivalent values in one system to a single code in the other
- Perform other mappings

For example, the FromSDAAllergyCategory data lookup table maps multiple possible SDA values, including 414285001, food, Food, and FoodAllergy, to the single FHIR DSTU2 `food` value.

To view a data lookup table:

1. In the FHIR namespace, go to the **Lookup Tables** page (**Interoperability** > **Configure** > **Data Lookup Tables**).
2. On the **Lookup Tables** page, select **Open**, which displays the list of lookup tables.
3. To view a particular lookup table, click on its name. This displays its page in the portal, which is a set of key-value pairs.

To customize a data lookup table:

1. In the FHIR namespace, go to the **Lookup Tables** page (**Interoperability** > **Configure** > **Data Lookup Tables**).
2. On the **Lookup Tables** page, select **Open**, which displays the list of lookup tables.
3. To customize a particular lookup table, click on its name. This displays its page in the portal, which is a set of key-value pairs.
4. Click **Save As** to save the lookup table with a new name.
5. Save the customized table with the name such as `Local.FromSDAAllergyCategory`. Customized names always begin with `Local.`, including the period.
6. Once the customized table has a new name, you can customize an individual key-value pair:
   - Delete it by clicking the red X to its left.
   - Modify it by clicking it; enter new values on the right for its **Key** or **Value**; and then select **Apply** on the right.
7. Click **Save** to save the customized lookup table.

For more information about lookup tables, see the "Defining Data Lookup Tables" section in the "Defining Other Options for Productions" chapter of *Configuring Productions*.

7.5 Customizing a Transformation from SDA to FHIR DSTU2

This section demonstrates two actions that modify a transformation:

- Removing Elements of a Transformation

- Modifying the Transformation for a Particular Property

7.5.1 Removing Elements of a Transformation

This section demonstrates removing transformations for three elements:

- SDA Race, which is transformed to FHIR DSTU2 `us-core-race`
- SDA EthnicGroup, which is transformed to FHIR DSTU2 `us-core-ethnicity`
- SDA Religion, which is transformed to FHIR DSTU2 `us-core-religion`

In sample SDA, these appear in a form such as:

```
<Container>
 ...
   <Religion>
        <Code>PRE</Code>
        <Description>Christian: Presbyterian</Description>
        <SDACodingStandard>ReligiousAffiliation</SDACodingStandard>
   </Religion>
   <MaritalStatus>
        <Code>S</Code>
        <Description>Single</Description>
        <SDACodingStandard>MaritalStatus</SDACodingStandard>
   </MaritalStatus>
   <Gender>
        <Code>M</Code>
        <Description>M</Description>
   </Gender>
   <Race>
        <Code>2106-3</Code>
        <Description>White</Description>
        <SDACodingStandard>Race & Ethnicity - CDC</SDACodingStandard>
   </Race>
   <Races>
        <Race>
                <Code>2106-3</Code>
                <Description>White</Description>
                <SDACodingStandard>Race & Ethnicity - CDC</SDACodingStandard>
        </Race>
   </Races>
   <EthnicGroup>
        <Code>Not Hispanic or Latino</Code>
        <Description>Is Not Hispanic or Latino</Description>
   </EthnicGroup>
   ...
</Container>
```

After conversion to FHIR DSTU2, they appear as:

```
<Patient>
  <extension url="http://hl7.org/fhir/StructureDefinition/us-core-race">
    <valueCodeableConcept>
      <coding>
        <system value="urn:oid:2.16.840.1.113883.6.238"/>
        <code value="2106-3"/>
        <display value="White"/>
      </coding>
    </valueCodeableConcept>
  </extension>
  <extension url="http://hl7.org/fhir/StructureDefinition/us-core-ethnicity">
    <valueCodeableConcept>
      <coding>
        <code value="Not Hispanic or Latino"/>
        <display value="Is Not Hispanic or Latino"/>
      </coding>
    </valueCodeableConcept>
  </extension>
  <extension url="http://hl7.org/fhir/StructureDefinition/patient-mothersMaidenName">
    <valueString value="Rogers"/>
  </extension>
  <extension url="http://hl7.org/fhir/StructureDefinition/us-core-religion">
    <valueCodeableConcept>
      <coding>
        <system value="urn:oid:2.16.840.1.113883.5.1076"/>
        <code value="PRE"/>
```

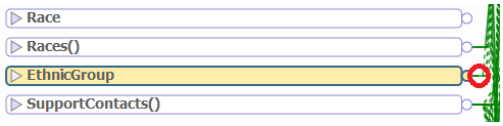
```
        <display value="Christian: Presbyterian"/>
      </coding>
    </valueCodeableConcept>
  </extension>
  <identifier>
    <type>
      <coding>
        <system value="http://hl7.org/fhir/v2/0203"/>
        <code value="MR"/>
        <display value="Medical record number"/></coding></type></identifier>
</Patient>
```

To remove a transformation for these elements:

1. Specify the location from which HS.FHIR.FromSDA.DTL.TransactionProcess loads custom data transformations by setting its CustomDTLPackage property, such as to the `User.Custom.FromSDA` package.
2. In the FHIR namespace, go to the DTL editor (**Interoperability** > **Build** > **Data Transformations**).
3. In the **DTL Editor**, open the transformation that converts an SDA Patient to the FHIR DSTU2 `Patient` resource:
   a. At the top of the page, select **Open**, which displays the **Finder Dialog** for DTL data transformations.
   b. In the **Finder Dialog**, select **HS** > **FHIR** > **DTL** > **FromSDA** > **Patient**

   This displays the editable data transformation that converts an HS.SDA3.Patient object to an HS.FHIR.Model.Resource.Patient.
4. To create custom data transformation, select **Save As** from the top of the page, which displays the **Save Data Transformation** dialog.
5. In the **Save Data Transformation** dialog:
   a. From the **Package** field, enter the name of the custom package location set above; here, this is User.Custom.FromSDA.
   b. In the **Description** field, enter a description of the modified transformation.
   c. Click **OK**.

   This saves the modified transformation into the custom package; here, this is User.Custom.FromSDA.Patient.
6. Close the confirmation dialog.
7. From the left column (**HS.SDA3.Patient**), select **EthnicGroup**. This displays the connections between the **EthnicGroup** (as SDA) and the relevant properties that are part of the `Patient` FHIR resource.
8. Click on the connection from **EthnicGroup** to FHIR DSTU2, which is the green line to the right of the **EthnicGroup** element:



   This both highlights the connection to the resulting FHIR and, below the connections, also highlights the line of code that performs this part of the transformation. The line of code that performs this section of the transformation is part of a block of code that composes the entire transformation, and begins with a line such as:

```
if ##class(HS.FHIR.DTL.Utils).CTIsDefined(source,"EthnicGroup")
```

   and ends with lines such as:

```
set iExtension = iExtension + 1
else
endif
```

In the DTL code that appears at the bottom of the window, these may be roughly from line 126 to line 135.

9. Delete all the code, from the **if** to the **endif**, which removes all the logic that performs the transformation. To do this, click the red **X** to the left of the if statement. Once you confirm your choice, this removes the entire statement (from the **if** to the **endif**) from the transformation.
10. Perform the analogous actions for Religion and Races().
11. Save the changes to the transformation by clicking **Save** button at the top of the page.
12. Compile the modified data transformation by clicking the **Compile** button at the top of the page.

Having modified the transformation, the FHIR DSTU2 output based on the same SDA input now includes the following lines:

```
<Patient>
  <extension url="http://hl7.org/fhir/StructureDefinition/patient-mothersMaidenName">
    <valueString value="Rogers"/>
  </extension>
  <identifier>
    <type>
      <coding>
        <system value="http://hl7.org/fhir/v2/0203"/>
        <code value="MR"/>
        <display value="Medical record number"/></coding></type></identifier>
</Patient>
```

These lines do not include `us-core-race`, `us-core-ethnicity`, and `us-core-religion`, because of the modifications to the transformation.

7.5.2 Modifying the Transformation for a Particular Property

This section demonstrates modifying a transformation for the SDA EncounterType, which is transformed to a property of the `Encounter` resource. The default SDA-to-FHIR DSTU2 transformation handles three encounter types:

- Inpatient
- Outpatient
- Emergency

To look at the existing mapping, open the FromSDAEncounterEncounterType transformation according to the instructions in the "About the SDA-FHIR DSTU2 Translation Tables" section.

In SDA, a sample that includes EncounterType is:

```
...
        <Encounter>
                <EncounterNumber>urn:uuid:63e41adf-415a-459d-a803-90484722a896</EncounterNumber>
                <AdmissionType>
                        <Code>R</Code>
                        <Description>Routine</Description>
                </AdmissionType>
                <EncounterType>O</EncounterType>
...
```

After conversion to FHIR DSTU2, the relevant section of the `Encounter` resource appears as:

```
...
    <resource>
      <Encounter>
        <identifier>
          <use value="official"/>
          <value value="urn:uuid:63e41adf-415a-459d-a803-90484722a896"/>
        </identifier>
        <class value="outpatient"/>
        <type>
          <coding>
            <code value="R"/>
            <display value="Routine"/>
          </coding>
        </type>
...
```

As an example, this section adds a fourth EncounterType, `W`, for a visit to a walk-in clinic. It modifies the FromSDAEncounterEncounterType transformation to translate this EncounterType value to the FHIR DSTU2 value, `emergency`. The process of modifying the transformation involves several actions:

1. Determine if any change is required in the transformation. For example, if a walk-in visit were treated as an outpatient visit, no change would be required, because the `Encounter` transformation sets the value of the FHIR to `outpatient` by default. To confirm this:

   a. Open the Encounter transformation in the DTL Editor (**Interoperability** > **Build** > **Data Transformations**, which opens the DTL Editor).

   b. In the **DTL Editor**, open the Encounter transformation (**Open** > **HS** > **FHIR** > **DTL** > **FromSDA** > **Encounter**).

   c. On the **Source** side (**HS.SDA3.Encounter**), select the **EncounterType** property and then click on its connection; in the right pane, the **Action** tab's **Value** field holds the code that performs transformation.

   d. Look at the code that sets this value. It uses the **HS.FHIR.DTL.Utils.LookupFHIR** method, and this method's third argument specifies the default transformed value, which is `outpatient`. Hence, if an SDA EncounterType does not have a specified transformation, the Encounter transformation uses the default value. Because the SDA EncounterType transformation is being modified to transform the `W` in SDA to `emergency` in FHIR DSTU2, you must modify the FromSDAEncounterEncounterType transformation.

   Note: Processing the modified SDA (containing the `W` encounter) prior to creating a custom transformation result in FHIR DSTU2 output with an `outpatient` encounter, because this is the default case.

2. To create a custom version of the FromSDAEncounterEncounterType transformation, make a copy of it:

   a. Go to the **Lookup Tables** page (**Interoperability** > **Configure** > **Data Lookup Tables**)

   b. Open the FromSDAEncounterEncounterType transformation (select **Open**, then **FromSDAEncounterEncounterType** in the **Finder Dialog**). This displays the **FromSDAEncounterEncounterType** page with its **Lookup Table Viewer** pane.

   c. Click **Save As**, which displays the **Lookup Table: Save As** dialog.

   d. In that dialog, enter `Local.FromSDAEncounterEncounterType` as the name of the custom lookup table.

   e. Click **OK**.

   This creates the custom transformation and displays its name in the page's upper-left tab.

   Important: Custom transformations must always have names of the form `Local.`*TransformationName*.

3. Modify the custom transformation by adding a new key-value pair to it in the **Lookup Table Viewer** pane (on the right):

   a. In the **Key** field, enter `W` as the SDA to be transformed.

   b. In the **Value** field, enter `emergency` as the resulting FHIR.

   c. Below the **Key** and **Value** fields, click **Apply**.

d. In the main pane, click **Save**.

Given the SDA input from above, the modified transformation results in the following text:

```
...
    <resource>
      <Encounter>
        <identifier>
          <use value="official"/>
          <value value="urn:uuid:63e41adf-415a-459d-a803-90484722a896"/>
        </identifier>
        <class value="emergency"/>
...
```

This causes the walk-in visit to be treated as an emergency encounter.

8

Using and Customizing SDA-FHIR STU3 Data Transformations

This chapter includes the following topics:

- Overview of the SDA-FHIR STU3 Transformations
- SDA-FHIR STU3 Transformations: Mapping Conventions
- FHIR Bundle Handling
- FHIR Extensions
- Configuring Your Production for SDA-FHIR STU3 Conversions
- Using the SDA-FHIR STU3 Transformation API
- Customizing an SDA–FHIR STU3 Transformation
- Customizing FHIR STU3 Transformation Lookup Tables

8.1 Overview of the SDA-FHIR STU3 Transformations

Important: To complete the steps in the sections of this chapter, you must already have created a Foundation namespace. See the "Installing and Activating a Foundation Production" chapter of the *InterSystems IRIS for Health Installation Guide*.

InterSystems IRIS for Health SDA-FHIR STU3 transformations are comprised of Data Transformation Language (DTL) code, which is generated from field-to-field mapping definitions. To invoke the transformations on SDA or FHIR STU3 data, you can use business processes in a production or use the SDA-FHIR STU3 Transformation API.

To view field-level information on the transformation mappings:

1. Log in to the Management Portal as a user with the **`%Ens_EDISchemaAnnotations`** role.
2. Navigate to **Health** —> *FHIRnamespace*.
3. Expand the **Schema Documentation** menu option and click **FHIR Annotations**.
4. On this page, you can view information on FHIR STU3 –> SDA or SDA –> FHIR STU3 mappings.

- For example, if you wanted to see how the FHIR STU3 CarePlan resource and its elements map into SDA, choose **`CarePlan`** in the **FHIR3 By Name** dropdown list. The resulting table will show you the elements of the STU3 CarePlan resource in the **FHIR3 Source** column. The SDA3 mapping is shown in the **SDA3 Target** column.
- The **Help** and **FAQ** buttons provide guidance on using and interpreting the elements of the FHIR Annotations user interface.

8.2 SDA-FHIR STU3 Transformations: Mapping Conventions

8.2.1 Field-to-Field Mappings

Most mappings are field-to-field: The mapping finds a data value in a source field and assign that value to a target field.

8.2.2 Conditional Mappings

Some field-to-field mappings are applied conditionally. The DTL <if> element controls this in the code. The FHIR Annotations show the label **Condition to Set this Field** when they present this information.

8.2.3 Literal Values

Among the defined mappings are mappings of literal values to target fields. One purpose of these mappings is to provide values for required target fields when the source object definition contains no fields that could provide the data required by the target.

Often, mappings of this type are defined conditionally, to be used only when needed.

8.2.4 Mapping Single to List

When the source is single but the target is a list, transformations map the source item to the first entry in a target list that contains only one entry. This feature is handled automatically during code generation for transformations. Single to List does not require special attention in the mapping definitions.

8.2.5 Mapping List to Single (Values)

When the source field is a list of values, and the target field is a single value, the transformations concatenate the list of values into a single value, separating each value from the list with a semicolon and space.

8.2.6 Mapping List to Single (Objects)

For SDA to FHIR STU3: when the incoming SDA is a list of objects, and FHIR STU3 has one object, the mapping table contains two mapping entries for the source list field:

- One mapping maps the source list field to the target single field. The transformation generated from this mapping simply places the first list entry into the target field.

- The other mapping maps the source list field to the target FHIR extension that contains the full list of objects. The FHIR extension URL is the full source field name, including the resource name, but using all-lowercase text separated by hyphens.

For FHIR STU3 to SDA: when the incoming FHIR STU3 has a list of objects, and SDA has one object, the transformation uses the first STU3 object and drops all the others.

8.2.7 Mapping SDA CodeTableDetail to a FHIR Code

Transformations map an SDA **`CodeTableDetail`** (or one of its subclasses) to a FHIR coded object such as **`Coding`** or **`CodeableConcept`** as follows:

1. The Code value is mapped to the code field.
2. The Description is mapped to the display field.
3. If there is an OriginalText field, it is mapped to the text field.

8.2.8 Mapping Coded Values to FHIR using Lookup Tables

The mapping consults a lookup table to find the entry that maps code values from the source schema (SDA or FHIR STU3) to code values in the target schema (FHIR STU3 or FHIR DSTU2) for this mapping.

If the mapping cannot find the lookup table, or cannot find a matching entry in the lookup table and it has a non-empty default value defined, it applies its default value to the code field. Otherwise, the target receives no value from this mapping.

If the mapping is SDA to FHIR, and the source field contains a non-empty value, then by convention there are two mapping entries for this source field. Both entries execute under the same Condition to Set this Field:

- One entry does the lookup to retrieve the value to assign to the target field.
- The other stores the original source field value in a string-valued FHIR extension.

In either case, if there is a Description or OriginalText along with the Code value, it is mapped to FHIR where applicable.

8.2.9 Mapping a FHIR Code to SDA CodeTableDetail

When a FHIR STU3 primitive code or coded object such as **`Coding`** or **`CodeableConcept`** does not use a lookup to transform the code value from STU3 to SDA, it is transformed to SDA **`CodeTableDetail`** (or one of its subclasses) as follows:

- CodeableConcept.text is transformed to HS.SDA3.CodeTableTranslated.OriginalText
- CodeableConcept.coding.display (or Coding.display) is transformed to HS.SDA3.CodeTableDetail.Description
- CodeableConcept.coding.code (or Coding.code, or simply code) is transformed to HS.SDA3.CodeTableDetail.Code
- GetCodeforURI of CodeableConcept.coding.system (or Coding.system) is transformed to HS.SDA3.CodeTableDetail.SDACodingStandard
- CodeableConcept.coding.version (or Coding.version) to HS.SDA3.CodeTableDetail.CodeSystemVersionId

8.2.10 Mapping FHIR Coded Values to SDA using Lookup Tables

If you want a mapping to use a code lookup table for FHIR to SDA, the mapping table contains two mapping entries for the source field:

- One of the two entries consults a lookup table to find the entry that maps a FHIR STU3 code value to an SDA Code.
- The other mapping entry in the pair takes over when the lookup table entry is unavailable or does not provide a match. It maps the source FHIR code value (unchanged) into an SDA **`CodeTableDetail`** object, as described above. That is, if the FHIR code was inside a **`Coding`** or **`CodeableConcept`** object, the FHIR code, display, system, version, and text values all are mapped appropriately into SDA **`CodeTableDetail`** fields.

8.2.11 Mapping String Values to Numeric Values

When the target is FHIR, and a string value is mapped to a numeric value, the string may contain non-numeric text such as units of measurement or instructions. To handle this, there are two mapping entries for the source list field:

- One of the two entries always assigns the source string value to a FHIR extension that consists of one string-valued field.
- The other mapping entry tests the source string value to see if it is numeric. If so, it maps this numeric value to the target numeric field.

8.2.12 Multi-Part Literal Values for FHIR Code Objects

For some FHIR target fields that are **`Coding`** or **`CodeableConcept`** objects, a set of mappings from literal values forms a multi-part value that is assigned to the field when needed. The full set of fields that such an object can contain are: `code`, `system`, `display`, `text`, `version`, and `userSelected`.

Where this is the case, the DTL annotation element for the code field explains that this code resides within a **`Coding`** or **`CodeableConcept`** object that is receiving a multi-part literal value. The FHIR Annotations show that the set of literal value mappings relating to this code all have the same value in the Condition to Set this Field.

8.3 FHIR Bundle Handling

When transforming FHIR STU3 to SDA:

- Information about containment relationships is reflected in the FHIR Annotations for the Bundle resource.
- If a resource referenced by ID within the incoming STU3 bundle is not present in the bundle, transformation of the bundle continues. After processing the complete bundle, the transformation code sends an alert to the event log for the active namespace. The alert message identifies each missing resource and indicates that its data will not appear in the SDA data that results from this transformation.

When transforming SDA to FHIR STU3, if the outgoing STU3 bundle contains some resources that did not pass validation, the transformation code sends an alert to the event log for the active namespace. The alert message describes the validation issues.

8.4 FHIR Extensions

FHIR extensions for STU3 are available; the URL prefix for them is `http://intersystems.com/fhir/extn/`.

8.4.1 Support for SDA Fields with no Place In the STU3 Schema

When the target of a transformation is FHIR STU3, one or more SDA source data fields may have no home in the target schema. In that case, transformations map the data to a FHIR STU3 extension: the URL is the full source field name, including the resource name, but using all-lowercase text separated by hyphens.

An extension like this consists of a single string-valued field that holds the SDA data.

The URL prefix in this case is `http://intersystems.com/fhir/extn/sda3/lib`.

For example:

- SDA full path name listing class and property: HS.SDA3.Administration:AdministeredAmount
- Extension name: `administration-administered-amount`
- Full URL for the FHIR extension: `http://www.intersystems.com/fhir/extn/sda3/lib/administration-administered-amount`

8.4.2 Support for SDA CustomPairs

The transformations support the legacy CustomPairs property in SDA classes of type HS.SDA3.SuperClass.

CustomPairs is a collection of objects of type HS.SDA3.NVPairs, each of which has two properties, Name and Value. When the transformation code encounters this property in customer SDA data, and the target is FHIR, the collection is mapped to a FHIR extension that contains a Parameters resource. This Parameters resource is a collection of paired fields: name and valueString.

In the example below, the customized SDA Encounter object has an SDA CustomPairs collection with three members, each with the name `PlanOfCareInstructionsText`:

```
{
   "resourceType": "Encounter",
   "contained":
       [
           {
                "resourceType": "Parameters",
                "id": "63",
                "parameter":
                    [
                         {
                              "name": "PlanOfCareInstructionsText",
                              "valueString": "Doctor recommends at least 30 minutes of exercise per
                                              day"
                         },
                         {
                              "name": "PlanOfCareInstructionsText",
                             "valueString": "Use sports heart rate monitor to aid in monitoring effort

                                              level"
                         },
                         {
                              "name": "PlanOfCareInstructionsText",
                              "valueString": "Read \"South Beach Diet\""
                         }
                    ]
           }
       ],
   "extension":
       [
           {
                "url": "http://intersystems.com/fhir/extn/sda3/lib/encounter-custom-pairs",

                "valueReference":
                    {
                         "reference": "#63"
                    }
           }
       ],
   "id": "914"
}
```

8.5 Configuring Your Production for SDA-FHIR STU3 Conversions

InterSystem IRIS for Health comes with built-in business processes that handle SDA-FHIR STU3 conversions. You can add them to your existing FHIR production programmatically using the InterSystems Terminal. Instructions for using the Terminal and descriptions of the business hosts added to the production are found in:

- FHIR STU3 to SDA Conversions
- SDA to FHIR STU3 Conversions

8.5.1 FHIR STU3 to SDA Conversions

To configure your existing FHIR production to handle FHIR STU3 to SDA conversions, do the following:

1. Make sure you have already installed a FHIR production. For instructions on installing a new FHIR production, see Installing and Configuring a FHIR Production.
2. Log into the Terminal with a role that has the **%Admin_Manage:Use** privilege.
3. Go to any Foundation namespace, for example, the namespace that contains your FHIR production:

   ```
   zn "FHIRNAMESPACE"
   ```

4. Enter:

   ```
   do ##class(HS.HC.Util.Installer.Kit.FHIR.FHIRToSDA3).Add(,"FHIRNAMESPACE")
   ```

   Where *FHIRNAMESPACE* is the namespace where you installed your FHIR production.

As part of this setup procedure, InterSystems IRIS for Health:

- Adds a new business process, HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process, which accepts FHIR STU3 input and transforms it to SDA output. For more information, see About HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process.

The **Add** method does not create a default business service designed to accept FHIR STU3 and send it to HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process for transformation into SDA3. If you do not need to accept FHIR STU3 for any other reason except transforming it to SDA3, you can modify the HS.FHIR.Service.STU3 business service that was created when you first installed the FHIR production. In this case, change the TargetConfigName setting of HS.FHIR.Service.STU3 to point to HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process. Otherwise, if you are already HS.FHIR.Service.STU3 for other reasons, you must create a new business service that sends the FHIR STU3 data to HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process. In this case, you must also create a CSP app for the endpoint and add an entry in FHIR CSP Config to tie the new CSP app to the new business service.

8.5.1.1 About HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process

HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process is the built-in business process that converts FHIR STU3 data into SDA format. It:

- Accepts a FHIR STU3 resource or bundle as input.
- Converts the FHIR STU3 content to an SDA container.
- Forwards the container to the business host specified by the TargetConfigName setting.
- Receives the response from the business host.

- Returns a FHIR STU3 response (based on what it received) to the business host that originally called it.

Important: Initially, the TargetConfigName setting of HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process does not have a value. Set it to specify where the SDA output should be sent.

Messages

The request message from a business host to HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process is HS.Message.FHIR.Request.

The response message from HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process is HS.Message.FHIR.Response.

Settings

Settings of HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process that influence FHIR to SDA conversions include:

- TargetConfigName — Specifies the business host where the XMLMessage that includes the SDA3 stream is sent after it is transformed from FHIR STU3 by the DTL transformation. This setting is located in the **Basic Settings** section of the **Settings** tab in the Production Configuration window.
- CallbackClass — Specifies the class that is used for the callback object that is invoked during the DTL transformation process. If left blank, the class HS.FHIR.DTL.Util.API.HC.Callback.Default.<FHIRVersion>ToSDA3 is used at run time, where <*FHIRVersion*> is derived from the FHIRVersion property of the incoming FHIR request message. This setting is located in the **Additional Settings** section of the **Settings** tab in the Production Configuration window.
- OutputToQuickStream — By default, the output of HS.FHIR.DTL.Util.HC.FHIR.SDA3.Process is an HS.Message.XMLMessage object that contains the SDA3 stream produced by the DTL transformation. If this setting is checked, the SDA3 stream is placed in a separate HS.SDA3.QuickStream object, and the QuickStreamID of the QuickStream object is placed in the AdditionalInfoItem property of the XMLMessage. If this setting is not selected, the SDA3 stream is placed in the ContentStream property of the XMLMessage. This setting is located in the **Additional Settings** section of the **Settings** tab in the Production Configuration window.

8.5.2 SDA to FHIR STU3 Conversions

To configure your existing FHIR production to handle SDA to FHIR STU3 conversions, do the following:

1. Make sure you have already installed a FHIR production. For instructions on installing a new FHIR production, see Installing and Configuring a FHIR Production.
2. Log into the Terminal with a role that has the **`%Admin_Manage:Use`** privilege.
3. Go to any Foundation namespace, for example, the namespace that contains your FHIR production:

   ```
   zn "FHIRNAMESPACE"
   ```

4. Enter:

   ```
   do ##class(HS.HC.Util.Installer.Kit.FHIR.SDA3ToFHIR).Add(,"FHIRNAMESPACE","STU3")
   ```

   Where *FHIRNAMESPACE* is the namespace where you installed your FHIR production.

As part of the setup procedure, InterSystems IRIS for Health:

- Adds a new business service, EnsLib.File.PassthroughService. This business service accepts a file containing SDA content in XML format and sends it to the built-in business process. For more information, see About EnsLib.File.PassthroughService.
- Adds a new business process, HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process. This business host transforms SDA input to FHIR STU3 output. For more information, see About HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process.

8.5.2.1 About EnsLib.File.PassthroughService

This business service is a typical file service that accepts SDA file input and sends it to HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process so it can be transformed into FHIR STU3. Relevant settings include:

- File Path — The directory on disk from which it retrieves its file input. You must specify a value for this property, as it does not have one by default. This setting is located in the **Basic Settings** section of the **Settings** tab in the Production Configuration window.
- TargetConfigNames — The business host to which it sends its output. The setup method specifies HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process as the value of this property. This setting is located in the **Basic Settings** section of the **Settings** tab in the Production Configuration window.

The output of this business service is content in Ens.StreamContainer format.

8.5.2.2 About HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process

The HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process business process:

- Accepts an SDA container as input and loops through each contained object.
- Converts the SDA container to FHIR STU3 content, in the form of a FHIR `Bundle` resource.
- Forwards the FHIR STU3 content to the business host specified by the TargetConfigName setting.
- Receives a response from the business host.
- Returns a response (based on what it received) to the business host that originally called it.

Important: Set the value of the TargetConfigName setting of HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process. to specify where it should send its FHIR output.

Messages

The request message to HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process is either Ens.Container or HS.Message.XMLMessage.

There is no response message from HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process. It returns a success or failure status instead.

Settings

Settings of HS.FHIR.DTL.Util.HC.SDA3.FHIR.Process that influence SDA to FHIR conversions include:

- TargetConfigName — Specifies the business host to which HS.FHIR.FromSDA.DTL.Transaction.Process sends its output. This setting is located in the **Basic Settings** section of the **Settings** tab in the Production Configuration window.
- FHIRVersion — Specifies the version of the FHIR output. This setting is located in the **Additional Settings** section of the **Settings** tab in the Production Configuration window.
- TransmissionMode — Specifies how the business process transmits the FHIR STU3 bundle for further processing:
  - transaction — The business process sends the bundle of resources in a single interaction and the processing succeeds or fails for the whole of the bundle; if processing any single resource fails, processing for the other resources (and the entire bundle) stops. This is the default.
  - batch — The business process sends the bundle of resources in a single interaction and each resource is processed independently; if processing any single resource fails, processing for the other resources still proceeds. (This is analogous to the manner in which InterSystems IRIS for Health processes an SDA container.)
  - individual — The business process sends each resource from the bundle separately as its own interaction.

  This setting is located in the **Additional Settings** section of the **Settings** tab in the Production Configuration window.

- FullTransactionResponse — If selected, the FHIR request message that this process sends is created with a "PREFER" header value set to "return=representation". Per the FHIR spec, this header indicates to a FHIR server that every created or updated resource should be returned in its entirety as it is saved (i.e., with any modifications applied by the server). Whether the server actually does this depends on the server. In general, this setting should be left unchecked except during debugging or if the FHIR client has a specific need to receive back the created/updated resources, as requesting this information is likely to increase response time from the FHIR server. This setting is located in the **Additional Settings** section of the **Settings** tab in the Production Configuration window.
- FHIRFormat — Specifies whether the content is in XML or JSON format. This setting is located in the **Additional Settings** section of the **Settings** tab in the Production Configuration window.
- FormatFHIROutput — Specifies whether or not content is formatted for readability. If selected, this setting has a performance impact, and as such should be enabled only during development and testing. This setting is located in the **Additional Settings** section of the **Settings** tab in the Production Configuration window.
- CallbackClass — Specifies the class that is used for the callback object that is invoked during the DTL transformation process. If left blank, the class HS.FHIR.DTL.Util.API.HC.Callback.Default.SDA3To<FHIRVersion> is used at run time, where <*FHIRVersion*> is derived from the FHIRVersion property of the incoming FHIR request message. This setting is located in the **Additional Settings** section of the **Settings** tab in the Production Configuration window.
- ValidResourceRequired — If selected, the FHIR that is generated by the SDA3–to-FHIR DTL transformation must be valid FHIR. If it fails validation, then it will not be forwarded. Regardless of the value of this setting, validation failures will be logged in the Event Log as alert messages. This setting is located in the **Additional Settings** section of the **Settings** tab in the Production Configuration window.
- OutputToQuickStream — If selected, the FHIR payload sent by this business process is placed in an HS.SDA3.QuickStream object, and the id of the QuickStream object is place in the QuickStreamId property of the request message. If left unselected, the FHIR output from the transformation is placed in the Payload property of the request message. This setting is located in the **Additional Settings** section of the **Settings** tab in the Production Configuration window.

8.6 Using the SDA-FHIR STU3 Transformation API

To invoke the SDA-FHIR STU3 transformations directly without using the built-in business processes, use the method calls in the SDA-FHIR STU3 Transformation API. The method calls are available within HSLIB or your FHIR namespace.

8.6.1 Transforming SDA to FHIR STU3

To transform SDA to STU3, use the following class methods of HS.FHIR.DTL.Util.API.HC.Transform:

- `SDA3ToSTU3Stream(`*`container`*`, Output `*`bundle`*` As %Stream.Object, `*`bundleType`*` As %String, `*`format`*` As %String, `*`callbackObject`*` As HS.FHIR.DTL.Util.API.HC.Callback.Abstract.SDA3ToSTU3 = "", Output `*`validationErrors`*` As HS.Types.FHIR.Errors)`

  Parameters:

  - *`container`*: a stream with an XML representation of a HS.SDA3.Container object, or an instantiated HS.SDA3.Container object
  - *`bundle`*: Output in stream format
  - *`bundleType`*: either `transaction` or `batch`. For more information, see the definition of the batch/transaction interactions in the official FHIR specification.
  - *`format`*: either `json` or `xml`. Performance with JSON will always be better than that with XML.

  - *callbackObject*: an instance of a class that extends HS.FHIR.DTL.Util.API.HC.Callback.Abstract.SDA3ToSTU3 to override aspects of transformation behavior as detailed below. If you leave this parameter empty, the default implementation will be used.
  - *validationErrors*: Errors encountered when validating the Bundle as an object of type HS.Types.FHIR.Errors

- `SDA3ToSTU3Object(`*container*`, Output` *bundle* `As HS.FHIR.DTL.vSTU3.Model.Resource.Bundle,` *bundleType* `As %String,` *format* `As %String,` *callbackObject* `As HS.FHIR.DTL.Util.API.HC.Callback.Abstract.SDA3ToSTU3 = "", Output` *validationErrors* `As HS.Types.FHIR.Errors)`

  Parameters:

  - *container*: a stream with an XML representation of a HS.SDA3.Container object, or an instantiated HS.SDA3.Container object
  - *bundle*: Output as an object of type HS.FHIR.DTL.vSTU3.Model.Resource.Bundle
  - *bundleType*: either `transaction` or `batch`. For more information, see the definition of the batch/transaction interactions in the official FHIR specification.
  - *format*: either `json` or `xml`. Performance with JSON will always be better than that with XML.
  - *callbackObject*: an instance of a class that extends HS.FHIR.DTL.Util.API.HC.Callback.Abstract.SDA3ToSTU3 to override aspects of transformation behavior as detailed below. If you leave this parameter empty, the default implementation will be used.
  - *validationErrors*: Errors encountered when validating the Bundle as an object of type HS.Types.FHIR.Errors

The default behaviors for both methods are as follows:

- The incoming stream or object is broken down into individual streamlets, which are in turn transformed into STU3 resources.
- UUIDs are generated and assigned as resource IDs, and each resource ID added to a resourceIds object property for later lookups. This behavior can be overridden: see the class documentation for **HS.FHIR.DTL.Util.API.HC.Callback.Abstract.SDA3ToSTU3:AssignResourceID**(). For example, you could call **HS.FHIR.vSTU3.Repository.Storage:GenerateResourceID()** in lieu of generating UUIDs.
- If you override **AssignResourceID()**, you may also have to override **GetIdByIdentifer()** and **GetPatientId()**, both of which use the resourceIds property.
- The methods do not modify incoming URLs at all by default. This behavior can be overridden with the **GetURLPrefix()** method: for example, if you are posting to a specific repository, you can provide the URL prefix for the repository.
- Resources will contain references to other resources regardless of the mechanism used to assign IDs.
- Patient and Encounter references will be added to all available resources using the Patient and Encounter streamlets. Encounter references can be made successfully only if the EncounterNumber fields in the SDA streamlets are used. If they are empty, no references will be generated.
- Anything that does not have a true destination in FHIR will be mapped to a FHIR extension.
- In the case of shared resources such as Organization, Practitioner, or Medication, a hash of the first 32 kilobytes of each resource is added to a hash table. Each subsequent shared resource is checked for duplication by searching the hash table for a direct match. If a match is found, the resource will be marked as a duplicate. This behavior can be overridden: see the class documentation for **HS.FHIR.DTL.Util.API.HC.Callback.Abstract.SDA3ToSTU3:IsDuplicate**().

- The Bundle will be validated at the end of the transformation and any errors saved in the `validationErrors` output. The Bundle will be returned whether or not there are errors: it is up to your implementation to check for and handle errors before the Bundle is used.

8.6.2 Transforming FHIR STU3 to SDA

To transform STU3 to SDA, use the following class methods of HS.FHIR.DTL.Util.API.HC.Transform:

- `STU3ToSDA3Stream(`*`bundle`*`, Output `*`container`*` As %Stream.Object, `*`callbackObject`*` As HS.FHIR.DTL.Util.API.HC.Callback.Abstract.STU3ToSDA3 = "")`

  Parameters:

  – *`bundle`*: an instance of HS.FHIR.DTL.vSTU3.Model.Resource.Bundle or a stream with a JSON or XML serialized representation of a Bundle object. Performance with JSON will always be better than that with XML.

  – *`container`*: A stream or a container as an object of type %Stream.Object

  – *`callbackObject`*: an instance of a class that extends HS.FHIR.DTL.Util.API.HC.Callback.Abstract.STU3ToSDA3 to customize the aspects of the transformation as detailed below. If you leave this parameter empty, the default implementation will be used.

- `STU3ToSDA3Object(`*`bundle`*`, Output `*`container`*` As HS.SDA3.Container, `*`callbackObject`*` As HS.FHIR.DTL.Util.API.HC.Callback.Abstract.STU3ToSDA3 = "")`

  Parameters:

  – *`bundle`*: an instance of HS.FHIR.DTL.vSTU3.Model.Resource.Bundle or a stream with a JSON or XML serialized representation of a Bundle object. Performance with JSON will always be better than that with XML.

  – *`container`*: A container stored in an HS.SDA3.Container object

  – *`callbackObject`*: an instance of a class that extends HS.FHIR.DTL.Util.API.HC.Callback.Abstract.STU3ToSDA3 to customize the aspects of the transformation as detailed below. If you leave this parameter empty, the default implementation will be used.

The default behaviors for both methods are as follows:

- An incoming FHIR STU3 Bundle is broken down into individual resources, and those resources transformed into SDA3 streamlets.
- When a reference is encountered in a resource and the referenced field has a mapping, the Bundle will be searched for that resource. If the resource cannot be found, an alert will be logged in the event log. If no existing transform handles the referenced resource type and the SDA3 target destination, it will be dropped. It may well be a valid reference, just not one that SDA3 supports.
- When a mapping is transforming a reference to an object and:

  – A subtransformation exists but the referenced resource has no values for any of the elements with mappings, or

  – There is no subtransformation from the referenced resource type to the datatype in the SDA3 object

  then no object will be created in the SDA streamlet.

- The EncounterNumber field on an Encounter streamlet will be populated starting at 1 and incremented for each encounter that is processed. Any subsequent resources that reference that Encounter resource, when transformed to SDA3, will perform a lookup based on the resource ID and will find the encounter number it should use. The assignment of encounter numbers can be overridden with **HS.FHIR.DTL.Util.API.HC.Callback.Abstract.STU3ToSDA3:AssignEncounterNumber()**.

- Similar to encounter numbers, ExternalID values for HealthConcern and Goal resources are populated starting at 1 by default. This behavior can be overridden with **HS.FHIR.DTL.Util.API.HC.Callback.Abstract.STU3ToSDA3:AssignExternalId()**.
- The value of Container:SendingFacility is set as follows: if a reference to the patient's managingOrganization exists and the organization is in the Bundle, it is used. Otherwise, the patient identifiers are searched for an MRN with an assigning authority, and that assigning authority is used. If neither of these items is found, the string `FHIR` is used.
- SDA3 extensions are not used. If a field does not exist in SDA3, the content will be dropped.
- If a Bundle comes in without a Patient resource, an error will be logged and nothing will be returned. Other than that, no validation will be performed on the container. It will simply be returned as is.

8.7 Customizing an SDA-FHIR STU3 Transformation

Each SDA to FHIR STU3 transformation and FHIR STU3 to SDA transformation corresponds to a Data Transformation Language (DTL) class that performs the transformation. These DTL classes can be customized using the DTL Editor or by editing the code directly.

There are two strategies you can follow when customizing a transformation DTL class. First, you can replace the entire standard DTL, in which case future changes to the standard DTL will need to be replicated in the customized version. Alternatively, you can create a custom DTL that calls the standard DTL through a method in the custom code. This approach is useful for pre-processors and post-processors. If you are calling the standard DTL from custom DTL code, use the **HS.FHIR.DTL.Util.Execute.ExecuteStandardClass()** method.

You customize an SDA-FHIR STU3 transformation within a specific namespace, not for the entire instance, so you can have different customizations in each namespace. If you want multiple namespaces to have the same customized transformations, you must repeat the customization process for each namespace.

8.7.1 Defining Package for Customizations

Before customizing SDA-FHIR STU3 transformations, you need to specify a single package for all customized classes. InterSystems recommends naming the class package: `HS.Local.FHIR.DTL`. To define the package for customized classes:

1. Open the InterSystems Terminal.
2. Change to the namespace of the FHIR production. For example, enter:

   ```
   zn "fhirNamepace"
   ```

   Where `fhirNamespace` is the FHIR production's namespace.
3. To check is a custom DTL class package already exists, enter:

   ```
   Write ##class(HS.FHIR.DTL.Util.API.ExecDefinition).GetCustomDTLPackage()
   ```

4. If you need to define a new custom DTL package, enter:

   ```
   Set status = ##class(HS.FHIR.DTL.Util.API.ExecDefinition).SetCustomDTLPackage("HS.Local.FHIR.DTL")
   ```

   Where `HS.Local.FHIR.DTL` is the recommended name for the new custom DTL package.
5. To verify that defining the DTL package was successful, enter:

   ```
   Write status
   ```

   The response should be: `1`.

8.7.2 Naming the Customized Transformation

Before customizing the transformation, you must call a method that returns the required name of the customized DTL class. After obtaining the name, you can do a Save As in the DTL Editor, or you can give the custom class this name in your code. To determine the required name for your customized class:

1. Open the Management Portal and navigate to the FHIR namespace.
2. Select **Interoperability** > **List** > **Data Transformations** from the Home page.
3. Find the name of the transformation that you want to customize. Transformations from SDA to FHIR STU3 are prefixed with `HS.FHIR.DTL.SDA3.vSTU3`. Transformations from FHIR STU3 to SDA are prefixed with `HS.FHIR.DTL.vSTU3.SDA3`.
4. Open the InterSystems Terminal.
5. To obtain the required name for the customized DTL class, enter the following in the Terminal:

   ```
   Write ##class(HS.FHIR.DTL.Util.API.ExecDefinition).PreviewDTLCustomClass("<standard_class_name>")
   ```

   Where *<standard_class_name>* is the full name of the transformation that you are customizing, including packages.
6. Be sure to make note of the response in the Terminal. You need to give your customized DTL class this name.

8.7.3 Creating the Customized DTL in the DTL Editor

In some cases, you might replace the entire standard DTL with a customized version by using a Save As in the DTL Editor. If you are using the DTL Editor, you can:

1. Open the standard DTL, and click **Save As**.
2. In the **Package** field, enter the package from the name of the customized DTL class that appeared in the Terminal. For example, if the customized class name in the Terminal was `HS.Local.FHIR.DTL.SDA3.vSTU3.Address.Address`, then enter `HS.Local.FHIR.DTL.SDA3.vSTU3.Address` (without the actual class name).
3. In the **Name** field, enter the name of the customized class. For example, if the customized class name in the Terminal was `HS.Local.FHIR.DTL.SDA3.vSTU3.Address.Address`, then enter `Address`.
4. Enter a description and click **OK**.

8.7.4 Editing the Customized Transformation in the DTL Editor

Once you have saved the customized transformation with a new name, you can use the DTL Editor to edit it. The following resources can help you get started with the DTL Editor:

- For examples of modifying an SDA to FHIR DSTU2 transformation, see Customizing a Transformation from SDA to FHIR DSTU2. Though the FHIR standard is different in these examples, the concepts are the same.
- For a simple, hands-on tutorial of the DTL Editor in a non-healthcare scenario, see *First Look: Data Transformations*.
- For complete details about creating DTL transformations, see *Developing DTL Transformations*.

8.7.5 Copying Custom Class to Mirror Members

If your environment uses mirroring and the package of your customizations resides in a non-mirrored database, you must copy the customized DTL class to the custom package on each mirror member. For example, if you defined the package

for customized classes as `HS.Local.FHIR.DTL`, then you must copy the customized DTL class to `HS.Local.FHIR.DTL` on each mirror member because `HS.Local` resides in the `HSCUSTOM` namespace, which is not mirrored. If your custom package resides in a mirrored database, no further action is required.

8.8 Customizing FHIR STU3 Transformation Lookup Tables

You can customize the lookup tables that are used by FHIR STU3 DTL transformations. These lookup tables are defined in a JSON file called Lookup.json. You can use a Terminal utility to modify a lookup table or you can edit a custom version of Lookup.json directly.

8.8.1 Using the Terminal Utility to Customize a Lookup Table

InterSystems IRIS for Health comes with a utility that leads you through the process of customizing a lookup table.

To run the customization utility:

1. Open the InterSystems Terminal.
2. To change to the FHIR namespace, enter:

   ```
   zn "fhirnamespace"
   ```

   where `fhirnamepace` is the FHIR STU3 namespace you have created.
3. To start the utility, enter:

   ```
   do ##class(HS.FHIR.DTL.Util.API.LookupTable).EditLookupTable()
   ```

4. Enter the Mapping Source for the lookup table you are customizing. For example, if you are customizing a lookup table that maps values from SDA3 to STU3, enter SDA3.
5. Enter the Mapping Target for the lookup table you are customizing. If you selected SDA3 for the Mapping Source in the previous step, STU3 is selected automatically and you can skip to the next step.
6. Enter the number that corresponds to Mapping Source Value Set in the lookup table you want to customize.
7. If only one lookup table with the Mapping Source Value Set exists, the Mapping Target Value Set is selected automatically and you can skip to the next step. If not, enter the number that corresponds to the Mapping Target Value Set you want to customize.
8. Select the code-to-code mapping you want to edit. If you want to add a new code-to-code mapping in the lookup table, enter `+`.
9. If you are editing the target value of a code-to-code mapping, enter the new target value for the mapping.

   If you want to edit the source value of the code-to-code mapping, you must enter `-` to delete the entire code-to-code mapping, then re-run the utility to add a new mapping with the correct source and target values.

8.8.2 Editing Lookup.json to Customize a Lookup Table

You can customize lookup tables by adding, deleting, or editing key/value pairs in a JSON file that contains all of the lookup tables used by transformations. Before beginning, you must make a custom copy of Lookup.json and put it into a special directory.

8.8.2.1 Creating Custom Lookup.json File

To create a custom JSON file that will be used by transformations when accessing lookup tables:

1. Navigate to <install-dir>\dev\fhir\lookup, where *<install-dir>* is the directory where you installed InterSystems IRIS for Health.
2. If it does not already exist, create a new directory called custom.
3. Navigate to the custom directory.
4. If it does not already exist, create a new directory that is the name of your FHIR namespace in all capital letters. For example, if the namespace that contains your FHIR STU3 production is called `fhirnamespace`, create a directory called FHIRNAMESPACE.
5. Copy Lookup.json from the <install-dir>\dev\fhir\lookup directory to the new <install-dir>\dev\fhir\lookup\custom\<NAMESPACE> directory.

   You can now begin to edit the lookup tables in the new copy of Lookup.json.

8.8.2.2 Editing Custom Lookup.json File

To begin customizing a lookup table, you must gather four pieces of information:

- Mapping Source
- Mapping Target
- Mapping Source Value Set
- Mapping Target Value Set

These values can be found in the FHIR Annotations in the Management Portal. To access these values:

1. Open the Management Portal and navigate to your FHIR namespace.
2. From the Home page, select **Health** > **Schema Documentation** > **FHIR Annotations**.
3. In the first drop-down list, select the type of transformation that contains the lookup table you are customizing. Choices are **FHIR3 —>FHIR2**, **FHIR3—>SDA3**, and **SDA3 —>FHIR3**.

   Make note of the Mapping Source and Mapping Target. The first interface format in the transformation pair is the Mapping Source. The second interface format is the Mapping Target. For example, if you select **FHIR3—>SDA3**, vSTU3 is the Mapping Source and SDA3 is the Mapping Target.
4. Click the **View <transformation> Lookup Tables** button, where the full name of the button depends on which transformation pair you selected.
5. Using the **View Lookup Tables** dialog, use the drop-down lists to note the Mapping Source Value Set and Mapping Target Value Set. The Mapping Source Value Set is the name in the left-hand drop-down list. The Mapping Target Value Set is the name in the right-hand drop-down list.

Now that you have the Mapping Source, Mapping Target, Mapping Source Value Set, and Mapping Target Value Set, you can edit a lookup table by adding, deleting, or editing the appropriate key/value pair in the custom Lookup.json file.

The top-level key/value pair in Lookup.json corresponds to the Mapping Source to Mapping Target relationship. For example, a lookup table used by SDA3 to FHIR STU3 transformations looks like:

```
"SDA3" : {
    "vSTU3" : {
```

The next level of key/value pairs corresponds to the Mapping Source Value Set to the Mapping Target Value Set. Search for the correct lookup table by finding the corresponding key/value pair. For example:

```
"HS.SDA3.Alert:Status" : {
    "event-status" : {
```

Once you have located the lookup table, you can add, delete, or edit the key/value pairs that correspond to the code-to-code mappings.

```
"A":"in-progress",
"C":"unknown",
"I":"aborted",
"INT":"completed"
```

A

Compliance with FHIR IHE Profile Specifications

InterSystems IRIS for Health adheres to the PIXm, PDQm, and MHD FHIR profile specifications in most respects. Differences are noted below.

A.1 Differences from PIXm Specification

Comments below apply to both DSTU2 and STU3.

A.1.1 Section 41.6.1: Proxy Model, Section 41.6.2 Manager group

PIXm server-side support is a proxy, not a pure FHIR model. This means that the PIXm Manager business process does not interact with the FHIR resource repository.

The PIXm Manager can be a proxy to our PIX, PIXv3 or PIXm client business operations by setting **`MPIOperations`** to point to any of those operations.

A.1.2 Section 3.83.4.2.2.2: Message Semantics

PIXm responses return only targetIdentifier elements, not targetId elements.

A.1.3 Section 3.83.5: Security Considerations

The PIXm implementation does not log ATNA audit messages.

A.2 Differences from PDQm Specification

A.2.1 DSTU2

A.2.1.1 Section 38.2.1 Pediatric Option

This option is supported with the exceptions below:

- Multiple Birth Indicator and Patient Birth Order cannot both appear in a given patient record.
- The Mother's Maiden Name extension is not supported.
- Last Update Date/Time and Last Update Facility are not supported.

A.2.1.2 Section 3.78.5 Security Considerations

The DSTU2 PDQm implementation does not log ATNA audit messages.

A.2.2 STU3

A.2.2.1 Section 38.2.1 Pediatric Option

This option is supported with the exceptions below:

- Multiple Birth Indicator and Patient Birth Order cannot both appear in a given patient record.
- Last Update Date/Time and Last Update Facility are not supported.

A.2.2.2 3.78.4.1.2.1 Query Search Parameters

In SDA3, there is no structure analogous to the FHIR STU3 Patient resource `active` parameter. Therefore the following provision is made:

- In HS.FHIR.vSTU3.PDQm.Consumer.Operations, `active` may be passed in only via a PatientSearchRequest AdditionalInfoItem named `FHIRactive`.
- In HS.FHIR.vSTU3.PDQm.Supplier.Process, active is handled as an invalid/unsupported search parameter and is reported as such.

A.2.2.3 Section 3.78.5 Security Considerations

The STU3 PDQm implementation does not log ATNA audit messages.

A.3 Differences from MHD Specification

A.3.1 DSTU2

A.3.1.1 3.65.4.1.3 Expected Actions (paragraph 600), 5.4.1.3 Folder Metadata Attributes

The List resource for XDSb Folders in DSTU2 is not supported.

A.3.1.2 33.2.2 XDS on FHIR Option, 33.6.1 MHD Actor Grouped with XDS Infrastructure

This option is supported. However, this is the only mode in which MHD is supported. The other mode would be as a pure FHIR implementation, which is unverified and untested.

A.3.1.3 33.6.2 MHD Actors Grouped with XCA Infrastructure

MHD support in conjunction with XCA has not been tested.

A.3.1.4 33.6.3 MHD Actor grouped with Retrieve Information for Display (RID) Profile

MHD support in conjunction with RID has not been tested.

A.3.1.5 3.66.4.1.2.1 Query Search Parameters (Find Document Manifests)

The patient search parameter for find document manifests is not supported.

A.3.1.6 3.67.4.1.2.1 Query Search Parameters (Find Document References)

The patient search and period search parameters for find document references are not supported.

A.3.2 STU3

A.3.2.1 33.2.2 XDS on FHIR Option, 33.6.1 MHD Actor Grouped with XDS Infrastructure

This option is supported. However, this is the only mode in which MHD is supported. The other mode would be as a pure FHIR implementation, which is unverified and untested.

A.3.2.2 33.6.2 MHD Actors Grouped with XCA Infrastructure

MHD support in conjunction with XCA has not been tested.

A.3.2.3 33.6.3 MHD Actor grouped with Retrieve Information for Display (RID) Profile

MHD support in conjunction with RID has not been tested.

A.3.2.4 3.66.4.1.2.1 Query Search Parameters (Find Document Manifests)

The patient search parameter for find document manifests is not supported.

A.3.2.5 3.67.4.1.2.1 Query Search Parameters (Find Document References)

The patient search and period search parameters for find document references are not supported.