Routing DICOM Documents in Productions

Version 2024.1
2024-05-02

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

Routing DICOM Documents in Productions
InterSystems IRIS Data Platform Version 2024.1 2024-05-02
Copyright © 2024 InterSystems Corporation
All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble® InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

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

1 Introduction to DICOM 1
1.1 InterSystems Support for DICOM 1
1.2 Sample Productions 2

2 Creating DICOM Associations 3
2.1 Introduction to DICOM Associations 3
2.2 Importing DICOM Associations 4
2.2.1 Import Association Example 4
2.3 Creating DICOM Associations 5
2.3.1 Transfer Syntax Considerations 6
2.4 Maintaining DICOM Associations 6
2.4.1 Creating and Editing DICOM Associations 7
2.4.2 Viewing Abstract Syntax 8
2.4.3 Viewing the DICOM Data Dictionary 8
2.5 Testing DICOM Associations 9

3 Tasks for DICOM Productions 11
3.1 Adding a DICOM Duplex Business Service 11
3.2 Adding a DICOM File Business Service 11
3.3 Creating a DICOM Business Process 12
3.3.1 Developing a DICOM Business Process Class 12
3.3.2 Adding a DICOM Business Process 13
3.3.3 Integrating a DICOM Business Process 13
3.4 Adding a DICOM Duplex Business Operation 13
3.5 Configuring a DICOM Duplex Business Host 14
3.6 Setting the DICOM User ID Class and Version 16
3.7 Configuring a DICOM Production to Control the Storage Location 16

4 Sample DICOM Modality Worklist Production 17
4.1 Configuring the DICOM Worklist Business Service 18
4.2 Creating a Business Process Class for a DICOM Worklist Production 19
4.2.1 DICOM Worklist Process Properties 19
4.2.2 DICOM Worklist Process Methods 19
4.3 Testing the DICOM Worklist Production 20

5 Sample DICOM Routing to Storage Production 23
5.1 Configuring the DICOM Routing to Storage Business Service 24
5.2 Creating a Business Process Class for a DICOM Storage Production 24
5.3 Testing the DICOM Routing to Storage Production 25

6 Sample DICOM File Storage Production 27
6.1 Configuring the DICOM Routing to Storage Business Service 28
6.2 Testing the DICOM File Storage Production 28

7 Sample DICOM Router Productions 29
7.1 Synchronous DICOM Router 29
7.2 Asynchronous DICOM Router 29

Appendix A: DICOM Transfer Syntax 31

DICOM Glossary of Terms 33

1

Introduction to DICOM

Digital Imaging and Communications in Medicine (DICOM) is a global information technology standard used in most hospitals worldwide. It is designed to ensure the interoperability of systems used to produce, store, display, process, send, retrieve, query, or print medical images and derived structured documents as well as to manage related workflow.

The DICOM standard is a product of the DICOM Standards Committee and its many international working groups. It is managed by the Medical Imaging & Technology Alliance, a division of the National Electrical Manufacturers Association (NEMA). For more information visit the organization's web site at https://www.dicomstandard.org/.

1.1 InterSystems Support for DICOM

InterSystems supports DICOM documents as virtual documents. A *virtual document* is a kind of message that InterSystems IRIS for Health™ and HealthShare® Health Connect parses only partially. This kind of message has the standard production message header and the standard message properties such as ID, Priority, and SessionId. The data in the message, however, is not available as message properties; instead it is stored directly in an internal-use global, for greater processing speed. For background information, see Using Virtual Documents in Productions.

InterSystems products provides specialized classes for use with DICOM communications. You can use these classes to add the following elements to a production:

Element	Options
Business services	InterSystems provides a couple of business service classes for DICOM, each with a specialized adapter.
Business processes	InterSystems provides a couple of sample business processes, shown in the sample productions.
Business operations	InterSystems provides one business operation class for DICOM, with a specialized adapter.
Message class	InterSystems provides a specialized message class for carrying DICOM documents as virtual documents. DICOM messages differ from other production messages in that the DICOM adapter is bidirectional and operates asynchronously.

InterSystems provides a bidirectional DICOM adapter to allow your production to send and receive DICOM messages and route them to and from devices that understand the DICOM protocol. InterSystems also provides a maintenance facility for defining the mechanism (DICOM associations) critical to communicating with DICOM devices; see Creating DICOM Associations.

The InterSystems implementation includes the ability to receive a message containing a DICOM-tagged image over TCP transport, to dissect that document and retrieve the tagged data items, and then to send that message to a DICOM recipient, again via TCP.

A DICOM message is a well defined entity which includes a metadata command set as well as the physical data set (which may or may not contain image data). To handle this type of messaging, InterSystems includes a specialized business host that has characteristics of both a business service and a business operation; it can send and receive messages in both directions.

The DICOM adapter is a gateway into the InterSystems production, allowing DICOM data to be on an equal footing with HL7 and other supported standard data formats. You can route and edit messages using **Get()**, **Set()**, and **GetNext()** methods on values positionally in a message or by name. You can use ObjectScript in a business process to route messages. You can route, modify, and build DICOM messages using classes. There are, however, some significant differences with DICOM messages because of the following limitations:

- InterSystems does not support a graphical detail editor like the one for HL7.
- InterSystems IRIS for Health and HealthShare Health Connect are*not* DICOM viewers; it contains no means of seeing images.
- The bidirectional nature of DICOM means you cannot write business processes using BPL; you must create a custom business process using ObjectScript.

Important: Actual DICOM data is not stored in an InterSystems database, but is stored as an external file stream to reduce journaling overhead and prevent inflating the database with many large files.

The DICOM adapter uses TCP to talk to modalities and systems (a PACS, for example), but InterSystems products can also open DICOM format files and treat them as DICOM messages using the EnsLib.DICOM.File class. InterSystems provides a file service in a demonstration production expressly for that purpose. The file service does not require an association because there is no communication with an originating entity via the DICOM protocol.

1.2 Sample Productions

Sample productions will be provided on github. InterSystems is testing these samples and this document will have links to the github repository. The following articles describe each of these productions:

- DICOM Modality Worklist Production — preparing the information to preload patient and scheduling data into a modality, an x-ray machine for example.
- DICOM Routing to Storage Production — routing messages containing images to the proper end point, a PACS system for example.
- DICOM File Storage Production — routing files containing instances of EnsLib.DICOM.Document to a DICOM storage system.

2

Creating DICOM Associations

This page describes how to create DICOM associations, which are required before the production can process DICOM input or output.

2.1 Introduction to DICOM Associations

Before you can develop a production that processes DICOM input or output, you require access to the information contained in the manufacturer's conformance statement for each DICOM device (*modality*). You use this information to create DICOM associations that the production uses to facilitate communication with a DICOM modality. A DICOM *association* is the metadata that describes the characteristics of the conversation of such communication.

The unique and most important aspect of productions interfacing with DICOM modalities is creating and testing associations. Before you can create these associations you require the information contained in the *Conformance Statement* from the manufacturer of the modality. A manufacturer claiming that their equipment or software conforms to the DICOM standard must be able to provide a conformance statement that describes exactly how that device or software conforms to the standard.

An association defines the characteristics of a conversation between modalities. To create an association, use the following information from the Conformance Statement:

- *Information objects recognized by this implementation* — In DICOM terms, Application Entity (AE) definitions supported by the modality. Application entities define the end points of an association. One DICOM device can have many AEs.
- *Service classes this implementation supports* — Service Object Pair (SOP) classes (or abstract syntax) with their accompanying binary transfer syntax. The list of SOP classes supported is one of the key aspects of the conformance statement. This list describes which service classes and information objects the application provides and accepts.
- *Presentation contexts* — Consist of the different options for transfer syntax for each abstract syntax. A DICOM conformance statement lists both presentation contexts that an application proposes during negotiation and those that it accepts.

During negotiation of the association, the production picks one transfer syntax to use. The order in which they are listed is important. The transfer syntaxes at the front of the list are given preference over the transfer syntaxes at the end of the list. You achieve better performance if you know the exact syntax, which is important for compressed images. The production does not convert from one syntax to another.

The conformance statement describes how an activity handles associations (that is, whether the activity initiates associations and accepts multiple associations) for each activity in the model. Some devices, such as the archives in a picture archiving and communication system (PACS), must support multiple associations for acceptable performance. Otherwise, only a single activity (for example, DICOM storage) could be handled at any given time.

The association information you enter is namespace-specific. You must create and run your production in the same namespace where you define and maintain these associations.

See EnsLib.DICOM.Util.AssociationContext in the Class Reference for more information.

2.2 Importing DICOM Associations

You can add associations to your interoperability-enabled namespace by importing acceptable and proposed presentation contexts from a file containing SOP / transfer syntax pairs by executing the **ImportAssociation()** method of EnsLib.DICOM.Util.AssociationContext.

The **ImportAssociation()** method has the following signature:

```
classmethod ImportAssociation(pFileName As %String,
                              pCallingAET As %String,
                              pCalledAET As %String,
                              pOverWriteExisting As %Boolean = 0)
```

The **ImportAssociation()** method has the following arguments:

pFileName

The name of a file containing a list of presentation contexts in the form of SOP class UID/transfer syntax UID. The method creates an association between the calling application entity title (AET) and the called AET with a presentation context for each pair in the file.

List the SOP classes (abstract syntaxes) one per line. Specify the transfer syntax by appending it to the line using a \ as the delimiter. If you do not specify any transfer syntaxes, then the import adds the default DICOM transfer syntax (`Implicit VR - Little Endian`). Use the numeric representation (UID) in the import file.

pCallingAET

Identifies the Application Entity (AE) that contains the requestor of the service; it is based on the source DICOM Application Name. The calling Application Entity Title (AET) is usually the Service Class User (SCU).

pCalledAET

Identifies the Application Entity that contains the intended acceptor of the service. It is based on the destination DICOM Application Name. The called Application Entity Title is usually the Service Class Provider (SCP).

pOverWriteExisting

If an association between the same AET peers exists in the namespace, this argument controls what happens. If True (1), overwrite the association. If False (0), display an error stating the association exists.

See DICOM Transfer Syntax for detailed information. For DICOM abstract syntax, use the Management Portal. Select **Interoperability** > **Interoperate** > **DICOM** > **DICOM Abstract Syntax**, and then click **Go**.

2.2.1 Import Association Example

For example, you can add an association to a work list modality using the information from the following sample *Presentation Context Table* taken from a conformance statement.

Presentation Context Table					
Abstract Syntax		Transfer Syntax		Role	Extended Negotiation
Name	UID	Name List	UID List		
Verification	1.2.840.10008.1.1	ILE	1.2.840.10008.1.2	SCP	None
Modality Worklist Information Model – FIND	1.2.840.10008.5.1.4.31	ILE ELE EBE	1.2.840.10008.1.2 1.2.840.10008.1.2.1 1.2.840.10008.1.2.2	SCP	None

Create a file for the association containing a line for each abstract syntax / transfer syntax combination. Use the UIDs for each item. For example, the contents of text file, MyAssociations.txt:

```
1.2.840.10008.1.1
1.2.840.10008.5.1.4.31\1.2.840.10008.1.2
1.2.840.10008.5.1.4.31\1.2.840.10008.1.2.1
1.2.840.10008.5.1.4.31\1.2.840.10008.1.2.2
```

If the only acceptable transfer syntax for a SOP class is the default `Implicit VR - Little Endian` (UID: 1.2.840.10008.1.2), you do not have to include the transfer syntax for that SOP class, as seen in line 1 of the file.

Once you create a file of abstract / transfer syntax pairs, you invoke the following method in the namespace where you are developing your production. For example:

```
 set $namespace = "mynamespace"

 Do ##class(EnsLib.DICOM.Util.AssociationContext).ImportAssociation
           ("c:\InterSystems\DICOM\MyAssociations.txt",
            "ENS-SCU",
            "MWL-SCP",
             0)
```

You can view the resulting association information in the Management Portal, as follows:

1. Click **Interoperability** > **Interoperate** > **DICOM** > **DICOM Settings**.

   The system displays a table that list the association information. Each row displays a **Calling AET** item and the associated **Called AET** item.

2. For more information on the association, select a row for the association and then click **Edit**.

   The system then displays the **Association** and **Presentation Context** tabs.

2.3 Creating DICOM Associations

You can also create a superset of associations between the calling and called AET for every known SOP class and a provided list of transfer syntaxes by executing the **CreateAssociation()** method of EnsLib.DICOM.Util.AssociationContext.

The **CreateAssociation()** method has the following signature:

```
classmethod CreateAssociation(pCallingAET As %String,
                              pCalledAET As %String,
                              pTransferSyntaxes As %List)
```

The **CreateAssociation()** method has the following arguments:

pCallingAET

Identifies the Application Entity (AE) that contains the requestor of the service; it is based on the source DICOM application name. The calling Application Entity Title (AET) is usually the Service Class User (SCU).

pCalledAET

Identifies the Application Entity that contains the intended acceptor of the service. It is based on the destination DICOM application name. The calling Application Entity Title is usually the Service Class Provider (SCP).

pTransferSyntaxes

A list of transfer syntaxes to pair with every known SOP class to create the presentation context for the association.

(The default list contains the required default `Implicit VR Little Endian` transfer syntax.)

This method creates an association context instance for the specified calling and called AET that supports *all* known SOPs.

Note: This association simply defines all SOP classes that are known in the data dictionary. There is a small performance penalty for this and it may be preferable for you to define just the SOP classes you need either using the Management Portal as described in the next section or by using the **ImportAssociation()** method described in the previous section. A handshake may take longer between the modality and the production if you include unnecessary classes in the association.

For example, a DICOM productions that generates default associations that include all known SOP classes could use the following code.

```
#; We will be accepting Storage and Query requests from JD-SCU
  Do ##class(EnsLib.DICOM.Util.AssociationContext).CreateAssociation
            ("JD-SCU","ENS-SCP",$ListBuild($$$IMPLICITVRLETRANSFERSYNTAX))

#; We will be sending storage requests to the JD-SCP
  Do ##class(EnsLib.DICOM.Util.AssociationContext).CreateAssociation
            ("ENS-SCU","JD-SCP",$ListBuild($$$IMPLICITVRLETRANSFERSYNTAX))
```

2.3.1 Transfer Syntax Considerations

Sometimes explicit works better because in many cases, it preserves original type names and provides additional backward compatibility.

Because explicit and implicit value representation (VR) encoding cannot be mixed, the decision regarding which technique to use must be made at the very beginning of any data transfer; DICOM applications negotiate and agree on encoding types before they exchange any data.

2.4 Maintaining DICOM Associations

To maintain your associations, use the **Interoperability** > **Interoperate** > **DICOM** > **DICOM Settings** page of the Management Portal.

This page lists any existing associations for the selected namespace and provides the following functions:

- **Create, edit, or delete associations** — InterSystems provides a form to enter or edit association information. You can also delete any existing association from the list. See Creating and Editing DICOM Associations.
- **View Abstract Syntax** — InterSystems provides a list of DICOM abstract syntaxes that includes all known SOP class information from the DICOM standard. See Viewing Abstract Syntax.
- **View Dictionary** — InterSystems provides a DICOM data dictionary that includes all known data element information from the DICOM standard. See Viewing DICOM Data Dictionary.

2.4.1 Creating and Editing DICOM Associations

The **Interoperability** > **Interoperate** > **DICOM** > **DICOM Settings** page lists all associations defined in the selected namespace. From this list you can **Edit** or **Delete** an existing association or you can choose to **Create a New Association**. To delete an association, click **Delete** in the appropriate row and click OK to confirm.

This page has two tabs:

- The **Association** tab enables you to create or edit associations. Enter or edit the following information on this tab:
  - **Calling AET** — Identifies the Application Entity (AE) that contains the requestor of the service; it is based on the source DICOM application name. The calling AET is usually the Service Class User (SCU).
  - **Called AET** — Identifies the Application Entity (AE) that contains the acceptor of the service; it is based on the destination DICOM application name. The called AET is usually the Service Class Provider (SCP).
  - **Name** — Optional name of the association up to 64 characters long.
  - **Description** — Optional description of the association up to 254 characters long.

  Click **Save** to update or create the association. If a required field is missing, its required description appears in red. Click **Close** to discard your updates and return to the list of existing associations on the **DICOM Settings** page.

  The calling and called application entity titles together create a unique ID for the association. Once you save the association, you cannot modify these fields; you can delete the association and then add a new one.

- The **Presentation Context** tab enables you to maintain the related presentation context for an association. You can perform the following actions on this tab:
  - **Return** — Click to go back to the list of associations on the **Interoperability** > **Interoperate** > **DICOM** > **DICOM Settings** page.
  - **Add** — Click to add an abstract syntax entry to the list and maintain the list of acceptable or proposed transfer syntax combinations. See Adding Presentation Context.
  - **Delete** — Highlight an abstract syntax and click **Delete**. To delete only a specific transfer syntax, click **Edit** and remove it from the selected list.
  - **Edit** — Click to add or remove transfer syntax for the selected abstract syntax.

  Once you have entered all the pertinent information for the association to a DICOM source, you can use the association information in the configuration of your production.

2.4.1.1 Adding Presentation Context

Each network transfer involving DICOM-specific messages begins with establishing an association handshake, when the two connecting applications exchange information about each other. This information is called the *presentation context*. The presentation context is the set of DICOM network services used over an association, as negotiated between application entities; it includes *abstract syntaxes* and *transfer syntaxes*. the production uses the EnsLib.DICOM.Util.PresentationContext class to represent this information. If the two applications can match their contexts, they can connect and start SCU-SCP processing.

To add a presentation context entry:

1. Click **Add** on the **Presentation Context** tab.
2. Click an abstract syntax supported by the DICOM device. The tab lists the UID and SOP class name for all known abstract syntax in the DICOM standard. You can only select one **Abstract Syntax** from the list at a time for which to add transfer syntax.

3. Once you select an abstract syntax, use the **>** arrow to move the types valid for your device from the **Available Transfer Syntax** list to the **Selected Transfer Syntax** list.
4. When you are finished selecting the appropriate transfer syntax for the abstract syntax, click **Save** to add the new context to the tree. Click Cancel to ignore your edits and return to the existing presentation context tree.

For example, the common SOP Verification class (1.2.840.10008.1.1) appears as an abstract syntax in many conformance statements. A conformance statement usually lists the acceptable presentation context combinations when the modality acts as a provider (SCP) and the proposed presentation context combinations when it acts as a user (SCU). From the previous example Presentation Context Table this class accepts the Implicit VR Little Endian transfer syntax (1.2.840.10008.1.2).

DICOM Transfer Syntax lists the information displayed on this page.

2.4.2 Viewing Abstract Syntax

The **Interoperability** > **Interoperate** > **DICOM** > **DICOM Abstract Syntax** page displays a table of SOP classes and their unique identifiers from the DICOM PS 3.6 standard. The term *abstract syntax* is used in part because it is defined in one of the international standards that DICOM references.

To use the **Filter** feature on the **View Abstract Syntax** page, enter a partial string and hit the **Tab** key to reload the table with the filter applied. The search is not case-sensitive and the filter returns results if your entry matches data in any column.

2.4.3 Viewing the DICOM Data Dictionary

The DICOM data dictionary represents the centralized registry which defines the collection of all DICOM data elements available to represent information, along with elements used for media encoding and a list of uniquely identified items that DICOM assigns. The **Interoperability** > **Interoperate** > **DICOM** > **DICOM Dictionary** page displays a table of elements from the DICOM PS 3.6 standard.

To use the **Filter** feature on the **View Dictionary** page, enter a partial string and press the **Tab** key to reload the table with the filter applied. The search is not case-sensitive and the filter returns results if your entry matches data in any column.

The following table describes the page display.

Column name	Description
Tag	A unique identifier for an element of information composed of an ordered pair of numbers (a group number followed by an element number), which is used to identify attributes and corresponding data elements.
Name	Name of the DICOM data dictionary element.
VR	Value representation — specifies the data type and format of the value(s) contained in the value field of a data element. (For example, the type of integer or string.)
VM	Value multiplicity — specifies the number of values that can be encoded in the value field of that data element.
(**Code**)	Displays a link if the item has code defined. When you click **Code**, a dialog box displays a list of values valid for this element with the corresponding internal numeric code. You may also filter this table by entering a partial string in the **Filter**.

The information is intended for reference only. The production does not enforce the use of these enumerated values as variations may appear in the field. You may use these reference tables when trying to decide what value to put in a particular field. The definitive reference is always the official NEMA DICOM PS 3.6 standard.

2.5 Testing DICOM Associations

Once you define your associations, you can test them in a production using the following procedure:

1. Obtain a TCP trace of a working modality before you configure the production.
2. Start either of the demonstration productions.
3. Add a business service.
4. Send an echo message.
5. Debug the connection.

The following is a sample log from a third-party DICOM testing tool showing the establishment of an association, a request and response of an ECHO message and a release of the association connection.

```
test: #5:ENS-SCP << A-ASSOCIATE-RQ PDU
test: #5:ENS-SCP >> A-ASSOCIATE-AC PDU
test: #5:ENS-SCP << C-ECHO-RQ Verification SOP Class
test: #5:ENS-SCP >> C-ECHO-RSP Verification SOP Class, status #0000H[Success]
test: #5:ENS-SCP << A-RELEASE-RQ PDU
test: #5:ENS-SCP >> A-RELEASE-RP PDU
test: #5:ENS-SCP closing socket
```

3

Tasks for DICOM Productions

This page describes how to perform typical production configuration specifically for a DICOM production:

3.1 Adding a DICOM Duplex Business Service

InterSystems provides a built-in duplex business host that accepts DICOM documents from external sources over TCP connections.

To add a DICOM duplex business service to a production:

1. Display the production in the **Interoperability** > **Configure** > **Production** page of the Management Portal.
2. In the **Services** column, click the Add button (a plus sign).
3. In the dialog box, select the EnsLib.DICOM.Service.TCP class from the **Service Class** list.
4. For **Service Name**, type the name of this business service. The name should be unique among the business services. Do not use periods or spaces.

   The default is the name of the class on which this service is based.
5. Click **OK**.

3.2 Adding a DICOM File Business Service

InterSystems provides a built-in file business service that accepts DICOM documents from external files.

To add a DICOM file business service to a production:

1. Display the production in the **Production Configuration** page of the Management Portal.
2. In the **Services** column, click the Add button (a plus sign).
3. In the dialog box, select the EnsLib.DICOM.Service.File class from the **Service Class** list.
4. For **Service Name**, type the name of this business service. The name should be unique among the business services. Do not use periods or spaces.

   The default is the name of the class on which this service is based.
5. Click **OK**.

6. Optionally enable the **UseStorageLocation** setting of the new business host (EnsLib.DICOM.Service.File).

   When a EnsLib.DICOM.Service.File reads a file, it creates a stream that it stores in the stream directory for the namespace (for example, mgr/name/stream). If **UseStorageLocation** is enabled, these streams will be stored in the StorageLocation, which is where the rest of the DICOM data is stored.

3.3 Creating a DICOM Business Process

To build a DICOM business process for use in a DICOM production, you must develop a custom business process class that performs the appropriate logic, create a business process that uses your custom class, configure it, and integrate it into the production. This topic explains each step:

- Developing a DICOM Business Process Class
- Adding a DICOM Business Process
- Integrating a DICOM Business Process

3.3.1 Developing a DICOM Business Process Class

The bidirectional nature of DICOM means you cannot write business processes using BPL; you must create a custom business process using ObjectScript.

When you create your custom class, extend EnsLib.DICOM.Process which is the superclass for all user-defined DICOM business processes. See its entry in the *Class Reference* for detailed information. For information on creating a class, see Developing Custom Business Processes.

Because of the duplex nature of DICOM communication, the business process must keep track of what is happening outside of the process. You can accomplish this using a context variable for the *state* of the process and model your code after a event driven finite state machine.

3.3.1.1 About Sessions

The base class for DICOM business processes includes a `StopPrivateSession` method that can be called to ensure that each message gets its own session. For example, the `OnMessage` method of a business process could include the following code:

```
#; Send the reply back to the service ( don't want a response )
Set tSC=..SendRequestAsync(..ServiceDuplexName,tOutput,0)
f $$$ISERR(tSC) Quit

#; Stop the private session to ensure each message gets its own session
Set tSC=..StopPrivateSession(..ServiceDuplexName)
If $$$ISERR(tSC) Quit
```

However this approach can cause an asynch timing issue if the business service in the production receives a new document before it processes the request to stop the private session. If this scenario is possible, and each DICOM incoming message requires its own private session, the solution is to place the `StopPrivateSession()` call in a custom business service

rather than the business process. For example, you could subclass `EnsLib.DICOM.Service.TCP` and override the `SendDocument` method with the following code:

```
Method SendDocument(pRequest As EnsLib.DICOM.Document, Output pResponse As %Library.Persistent) As
%Status
{
Set tSC = ##super(pRequest)
If $$$ISOK(tSC)

{ If ..%InPrivateSession Set tSC=..StopPrivateSession() }
Quit tSC
}
```

When deciding whether to stop a private session, it is important to consider whether a single request can have multiple replies. For example, if there is a request that expects multiple replies that can come in fast and cause an issue, then it makes sense to wait until everything has been processed before stopping the session. If, on the other hand, there is a simple 1-to-1 operation like a single C-STORE and its response, then it is appropriate to have each request/response in a separate private session.

3.3.2 Adding a DICOM Business Process

To add a DICOM business process to a production:

1. Display the production in the **Production Configuration** page of the Management Portal.
2. In the **Processes** column, click the Add button (a plus sign).
3. In the dialog box, select your custom class from the **Process Class** list.
4. For **Process Name**, type the name of this business process. The name should be unique among the business processes. Do not use periods or spaces.

   The default is the name of the class on which this process is based.
5. Click **OK**.

3.3.3 Integrating a DICOM Business Process

To integrate a new DICOM business process into a production, you must associate it with the business service that receives its incoming documents, and with the routing rule set that determines its actions based on those documents. To do this:

Select the DICOM business service in the configuration diagram. In the **Duplex Target Config Names** field enter the configured **Name** of the DICOM business process.

3.4 Adding a DICOM Duplex Business Operation

InterSystems provides built-in duplex business host that accepts DICOM document messages from external sources over TCP connections.

To add a DICOM business operation to a production:

1. Display the production in the **Production Configuration** page of the Management Portal.
2. In the **Operations** column, click the Add button (a plus sign).
3. In the dialog box, select the EnsLib.DICOM.Operation.TCP class from the **Operation Class** list.
4. For **Operation Name**, type the name of this business operation. The name should be unique among the business operations. Do not use periods or spaces.

The default is the name of the class on which this operation is based.

5. Click **OK**.

3.5 Configuring a DICOM Duplex Business Host

To configure a business host, click it in the diagram on the **Production Configuration** page, edit details on the **Settings** tab, and click **Apply**.

For a DICOM business host, you can configure the following settings:

Duplex Target Config Names

Specify the configuration item within the production to which the business host sends any DICOM documents that it receives.

LocalAET

The called Application Entity Title (AET) that the remote DICOM peer uses to communicate with the production. This corresponds to the **Called AET** you use when defining an association in an interoperability-enabled namespace. For DICOM business services, you can set this property and the `RemoteAET` property to the `*` wildcard. If both of these properties are set to the `*` wildcard, the business service bypasses the usual checks on AET names, instead does a direct lookup on association contexts defined within the database. If the association context is defined then the connection is opened and the association is negotiated as usual, if not then the connection is rejected.

RemoteAET

The calling Application Entity Title(s) of a remote DICOM peer.

When the adapter is in the role of Service Class Provider (SCP, server) it contains a comma-delimited list of *names* of the DICOM peers which are allowed to connect. A *name* can either be a literal string or a pattern/substitution specification of the form:

?*Pattern/Substitution*

For the pattern/substitution form, the production matches the calling AET against the pattern. If there is a match then it uses the substitution for association validation. For example, the following setting value:

```
?1"B".E/JD-SCU
```

Indicates that the pattern must match exactly one letter `B` followed by any number of any other character(s), so this entry matches any calling AET that starts with `B` and substitutes `JD-SCU`. To accept *any* AET use `?.E` for a pattern.

When the adapter is in the role of Service Class User (SCU, client), it must contain the AET of the DICOM peer to which the prodcution is connecting or be set to the `*` wildcard (see description of `LocalAET` property).

TraceVerbosity

Flag value for how much debug information to provide:

```
0 - No information
1 - Terse debug information
2 - Verbose debug information
```

ARTIM

The association request/reject/release timeout (in seconds). Adjust the value upwards if you find that a peer takes a long time to respond to association requests.

TXTIM

The data transfer timeout (in seconds). Adjust the value upwards if you find that a peer takes a long time to respond during data transfer.

Job Per Connection

True or False — Indicates whether or not to spawn a new process to handle each incoming TCP connection. A value of True allows simultaneous handling of multiple connections.

IP Port

IP port of the DICOM peer.

IP Address

IP address to connect to the DICOM peer. This property is null if it is in listening mode.

OS Accept Connection Queue Size

If in Listening mode, how many incoming connections should the OS hold open on our behalf until they can be processed. Set to 0 if only one connection at a time is expected; the operating system will refuse additional connection attempts. Set to a large number if many clients can connect rapidly (1000 is the maximum).

Local Interface

In a multi-homed system, specify which network interface the TCP connection should go through. An empty value means to use any interface. To be able to bind to IPv6 interfaces you may need to enable IPv6 in your instance. This is done in the System Management Portal under **System Administration** > **Configuration** > **Additional Settings** > **Startup**, by editing the IPv6 setting.

SSL Configuration

The name of an existing SSL/TLS configuration to use to authenticate this connection. Choose a client SSL/TLS configuration, because the adapter initiates the communication.

To create and manage SSL/TLS configurations, use the Management Portal. See Create or Edit a TLS Configuration. The first field on the **Edit SSL/TLS Configuration** form is **Configuration Name**. Use this string as the value for the **SSLConfig** setting.

The remaining settings are either common to all business hosts or are determined by the type of adapter. See

- Settings in All Productions
- File Adapter Settings
- TCP Adapter Settings

3.6 Setting the DICOM User ID Class and Version

The InterSystems DICOM implementation is identified by two items of information: the UIC and the VER. You can configure your own UIC or VER by setting the *^Ens.Config("DICOM","UIC")* node and the *^Ens.Config("DICOM","VER")* node respectively.

3.7 Configuring a DICOM Production to Control the Storage Location

Since the DICOM data is not stored in an InterSystems database, but is instead stored as an external file stream, you must configure DICOM productions to specify the location to use for these external files. You do this by defining a property named StorageLocation in your production. The DICOM business services, processes, and operations depend on the presence of this property˙. If the property specifies a directory, it is used as the storage location for the DICOM data. If the property has an empty string value, then the instance uses its default temporary directory to store the DICOM data.

Important: If you add a DICOM component to a production, you must ensure that the StorageLocation property is defined in the production. If the StorageLocation property is not present, then the DICOM components can encounter an error when attempting to create DICOM data. We recommend that you specify a directory to use for DICOM data and not specify an empty string, which uses the default temporary directory. Restarting the InterSystems instance may delete data from the default temporary directory.

Also, by default, when a EnsLib.DICOM.Service.File reads a file, it creates a stream that it stores in the stream directory for the namespace (for example mgr/name/stream). To instead store these streams in the StorageLocation, enable the **UseStorageLocation** setting of the EnsLib.DICOM.Service.File.

The StorageLocation property is defined in the production definition. To add this property to a new production, follow this procedure:

1. Edit the production class in Studio.
2. Insert the following lines in the production class definition after the XDATA that defines the production settings:

   ```
   Parameter SETTINGS = "ShutdownTimeout,UpdateTimeout,StorageLocation";

   /// This is the storage location for the DICOM streams to be stored
   Property StorageLocation As %String [ InitialExpression = "directory-location" ];
   ```

   Replace directory-location with the directory where DICOM data is to be stored on the system, such as c:\InterSystems\DICOM.
3. Compile the production class.
4. Open the production in the Management Portal. The **StorageLocation** property appears in the **Additional Settings** section. Set the value of **StorageLocation** as needed (or keep the value specified by the InitialExpression as shown previously).

Note: The EnsLib DICOM components check for the storage location with the following call:

```
Set tStorageLocation=..GetProductionSettingValue("StorageLocation",.tSC)
```

4

Sample DICOM Modality Worklist Production

A common DICOM SOP class is the Modality Worklist (MWL), which enables primary imaging equipment (modalities) to query for patient demographics and study details from the MWL Service Class Provider (SCP), normally part of a radiology information system (RIS). The modality asks for a list of patients with chosen criteria via a standard C-FIND operation and the Modality Worklist service responds accordingly.

The sample production named Demo.DICOM.Production.WorkList demonstrates how to use a production to process DICOM worklists for modalities that support them. This production contains the business service EnsLib.DICOM.Service.TCP, the business process Demo.DICOM.Process.Storage, and the business operation EnsLib.DICOM.Operation.TCP.

The scenario for this production is such that at the beginning of the day a modality needs a list of work to process. The production does the following:

1. The modality sends a C-FIND request message to the production.
2. The C-FIND request specifies the need for a worklist.
3. The production finds the information.
4. The production creates a C-FIND response message for each patient in the list.
5. The production sends the response message back to the modality.

This production includes business service EnsLib.DICOM.Service.TCP and the business process Demo.DICOM.Process.WorkList.

The business service, EnsLib.DICOM.Service.TCP, connects to a DICOM imaging device. The Demo.DICOM.Process.WorkList business process class demonstrates how to handle DICOM C-FIND-RQ messages and respond with worklist entries.

This is the only way that the modality worklist SOP class (MWL) works, namely the modality querying data from the service provider (SCP); there is no means for the SCP to broadcast the data out to modality equipment. Likewise, the MWL protocol itself provides no means for the modality to inform the RIS of which patient has been chosen from the list of matches returned, though this can be communicated using the Modality Performed Procedure Step (MPPS) service when it is supported by both pieces of equipment.

The following steps outline the procedure to add this type of interface to a production:

1. Create a generic production by clicking **Create New Production** on the **Production Configuration** page. See "Creating and Configuring a Production".
2. In Studio, modify the production definition to add both the StorageLocation property and the SETTINGS parameter. See Configuring a DICOM Production to Control the Storage Location for details on defining StorageLocation and

SETTINGS. The SETTINGS parameter is required for the StorageLocation to work correctly. After defining StorageLocation and SETTINGS, compile the production.

3. Add a DICOM duplex business service to the production using the EnsLib.DICOM.Service.TCP class.
4. Configure the DICOM business service settings specifically for a modality worklist.
5. Create a business process class that creates worklist entries from incoming DICOM C-FIND-RQ message documents.
6. Add a DICOM business process using the custom class from the previous step.
7. Test the production to verify that it receives a request message for a worklist and sends the appropriate response message documents back.

You can view the class code of Demo.DICOM.Production.WorkList.cls using Studio to see the production details.

4.1 Configuring the DICOM Worklist Business Service

You can configure a DICOM business service by clicking it in the diagram on the **Production Configuration** page. Configuring a DICOM Duplex Business Host describes the details. This section describes the settings specific to the business service in the demonstration worklist production.

Duplex Target Config Names

Specify the configuration item within the production to which the business service should send any DICOM documents that it receives.

The Demo.DICOM.ProductionStorage production uses the business process based on the class, Demo.DICOM.Process.WorkList, which is described in Creating a Business Process for a DICOM Worklist Production. This is a custom process that contains the logic for processing DICOM messages from the modality.

LocalAET

The called Application Entity Title (AET) that the remote DICOM peer uses to communicate with the production. This corresponds to the **Called AET** you use when defining an association in an interoperability-enabled namespace.

When you run the Demo.DICOM.Production.WorkList production for the first time, it creates the DICOM association necessary for the production to connect to a test DICOM application for this demonstration production. You can view this association in the production's namespace from the **Interoperability** > **Interoperate** > **DICOM** > **DICOM Settings** page.

RemoteAET

The calling Application Entity Title(s) of a remote DICOM peer.

When the adapter is in the role of Service Class Provider (SCP, server) it contains a comma-delimited list of names of the DICOM peers which are allowed to connect. A name can either be a literal string or a pattern/substitution.

The EnsLib.DICOM.Service.TCP service of the demo production uses JD-SCU for the value of **RemoteAET**.

4.2 Creating a Business Process Class for a DICOM Worklist Production

The Demo.DICOM.Production.WorkList production uses a sample custom business process class, Demo.DICOM.Process.WorkList, which demonstrates how to handle DICOM C-FIND request messages and respond with worklist entries. This demonstration class mimics retrieving patient information from a RIS system.

This custom class extends EnsLib.DICOM.Process, which is the superclass for all user-defined DICOM business processes. See its entry in the *Class Reference* for detailed information.

For instructions, see Developing Custom Business Processes in *Developing Productions*.

The Demo.DICOM.Process.WorkList class uses the parameter *SETTINGS* to expose a new property, NumberOfWorkListEntries, for configuration. This setting influences the number of worklist entries returned as a result of a C-FIND request.

See the following sections for an overview of the contents of the custom class:

- DICOM Worklist Process Properties
- DICOM Worklist Process Methods

You can view the class code for Demo.DICOM.Process.WorkList.cls using Studio to see the processing details.

Because of the duplex nature of DICOM communication, the business process must keep track of what is happening outside of the process. You can accomplish this using a context variable for the *state* of the process.

4.2.1 DICOM Worklist Process Properties

The custom class adds the following properties it needs to read in the requests and format responses:

DocumentFromService

The incoming message from the business service (a EnsLib.DICOM.Document object).

NumberOfWorkListEntries

Configurable number of worklist entries; defaults to 1.

OriginatingMessageID

Keeps track of the ID of the originating message.

ReplyCounter

Keeps track of the number of replies sent to the worklist request.

4.2.2 DICOM Worklist Process Methods

The custom class adds the following methods it needs to process the requests and create responses:

OnMessage()

This method handles the C-FIND request and C-CANCEL request messages. The find request asks for a query to be performed using the criteria specified in the dataset of the request. The query *may* take significant time and produce many results so individual matches are reported in one or more messages. This demonstration method does not expect any other types of DICOM messages.

The **OnMessage()** method has the following signature:

```
Method OnMessage(pSourceConfigName As %String,
                 pInput As %Library.Persistent)
                 As %Status
```

OnError()

This method is called when any error occurs. Returning the same error causes the business process to set its status to error and close down.

The **OnError()** method has the following signature:

```
Method OnError(request As %Library.Persistent, ByRef response As %Library.Persistent,
               callrequest As %Library.Persistent,
               pErrorStatus As %Status,
               pCompletionKey As %String)
               As %Status
```

CreateIntermediateFindResponse()

This method creates an intermediate instance of a message for the C-FIND response, filling in the mandatory DICOM fields. The protocol requires that all messages but the last one have their status set to *Pending*. This indicates to the client that there is more data coming. The last message has a status of *Success*, which means that the query has finished. This example ignores the selection criteria and returns dummy patient records. Your production implementation will be necessarily more complex.

The **CreateIntermediateFindResponse()** method has the following signature:

```
Method CreateIntermediateFindResponse(pDocIn As EnsLib.DICOM.Document,
                                      Output pDocOut As EnsLib.DICOM.Document)
                                      As %Status
```

CreateFinalFindResponse()

This method creates an instance of a message to indicate that the final process is complete, setting the command field to C-FIND-RSP to indicate that this is a find response message.

The **CreateFinalFindResponse()** method has the following signature:

```
Method CreateFinalFindResponse(pDocIn As EnsLib.DICOM.Document,
                               Output pDocOut As EnsLib.DICOM.Document)
                               As %Status
```

4.3 Testing the DICOM Worklist Production

Once you have working associations and created a production, you can attempt to process valid DICOM message documents through the worklist production. The demonstration DICOM production was developed using third-party software specifically developed for testing DICOM processing. You can use any of the many available software products, or test with your actual DICOM modality data.

This section shows portions of the Event Log, Message Browser, Message Details, and Message Contents pages in the Management Portal of the Demo.DICOM.Production.WorkList production running and accepting DICOM messages from a modality that supports worklists.

The following displays a DICOM message document within the Message Browser:

[Ensemble] > [Messages] Namespace: ENSDEMO

10	186	22:07:13.681	156	Completed	OK	Demo.DICOM.Process.WorkList	Details	Contents	Trace	Report

Command Set

Tag	Name	Contents
(0000,0000)	CommandGroupLength	70
(0000,0002)	AffectedSOPClassUID	1.2.840.10008.5.1.4.31
(0000,0100)	CommandField	32800
(0000,0120)	MessageIDBeingRespondedTo	1
(0000,0800)	CommandDataSetType	65278
(0000,0900)	Status	65281

Data Set

Tag	Name	Contents	
(0008,0050)	AccessionNumber	80372376	
(0010,0010)	PatientName	Patient90^Name	
(0010,0020)	PatientID	43171525	
(0020,000D)	StudyInstanceUID	1.2.40.0.13.0.192.168.1.3.13594894.1206712911593.32768	
(0032,1060)	RequestedProcedureDescription	RequestedProcedureDescription	
(0040,0100)	ScheduledProcedureStepSequence	(0008,0060) Modality	OT
		(0040,0001) ScheduledStationAETitle	StationAET
		(0040,0002) ScheduledProcedureStepStartDate	20080328
		(0040,0003) ScheduledProcedureStepStartTime	100151.000
		(0040,0007) ScheduledProcedureStepDescription	ScheduledProcedureStepDescription
		(0040,0009) ScheduledProcedureStepID	80871868
(0040,1001)	RequestedProcedureID	91249028	

You can see the structure of a DICOM message that contains metadata in the command set and the physical data in the data set.

The following portion of the Event Log page shows (from the bottom up) the events of running the production communicating with a modality and processing three worklist items as configured in the business process:

Method	Text
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Sending MSG type C-FIND-RSP
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Outbound Transfer Syntax : (1.2.840.10008.1.2) Implicit VR - Little Endian
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Outbound AbstractSyntax : (1.2.840.10008.5.1.4.31) Modality Worklist Information Model - FIND SOP Class
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Sending MSG type C-FIND-RSP
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Outbound Transfer Syntax : (1.2.840.10008.1.2) Implicit VR - Little Endian
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Outbound AbstractSyntax : (1.2.840.10008.5.1.4.31) Modality Worklist Information Model - FIND SOP Class
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Sending MSG type C-FIND-RSP
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Outbound Transfer Syntax : (1.2.840.10008.1.2) Implicit VR - Little Endian
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Outbound AbstractSyntax : (1.2.840.10008.5.1.4.31) Modality Worklist Information Model - FIND SOP Class
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Sending MSG type C-FIND-RSP
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Outbound Transfer Syntax : (1.2.840.10008.1.2) Implicit VR - Little Endian
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Outbound AbstractSyntax : (1.2.840.10008.5.1.4.31) Modality Worklist Information Model - FIND SOP Class
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Received MSG type C-FIND-RQ
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Inbound Transfer Syntax : (1.2.840.10008.1.2) Implicit VR - Little Endian
EnsLib.DICOM.Adapter.TCP :TraceMessage()	Inbound AbstractSyntax : (1.2.840.10008.5.1.4.31) Modality Worklist Information Model - FIND SOP Class

The following log from third-party DICOM testing software shows its communication with the production:

```
test: #3:ENS-SCP << A-ASSOCIATE-RQ PDU
test: #3:ENS-SCP >> A-ASSOCIATE-AC PDU
test: #3:ENS-SCP << C-FIND-RQ Modality Worklist Information Model - FIND SOP Class
test: #3:ENS-SCP << Dataset
test: #3:ENS-SCP >> C-FIND-RSP Modality Worklist Information Model-FIND SOP Class, status
#ff01H[StatusEntry.PENDING]
test: #3:ENS-SCP >> Dataset
test: #3:ENS-SCP >> C-FIND-RSP Modality Worklist Information Model-FIND SOP Class, status
#ff01H[StatusEntry.PENDING]
test: #3:ENS-SCP >> Dataset
test: #3:ENS-SCP >> C-FIND-RSP Modality Worklist Information Model-FIND SOP Class, status
#ff01H[StatusEntry.PENDING]
test: #3:ENS-SCP >> Dataset
test: #3:ENS-SCP >> C-FIND-RSP Modality Worklist Information Model-FIND SOP Class, status #0000H[Success]
test: #3:ENS-SCP << A-RELEASE-RQ PDU
test: #3:ENS-SCP >> A-RELEASE-RP PDU
test: #3:ENS-SCP closing socket
```

5

Sample DICOM Routing to Storage Production

The sample production named Demo.DICOM.Production.Storage demonstrates a basic DICOM function, sending an image from a modality to a RIS or PACS system for storage. This production contains the business service EnsLib.DICOM.Service.TCP, the business process Demo.DICOM.Process.Storage, and the business operation EnsLib.DICOM.Operation.TCP.

The scenario for this production is a DICOM modality sending a request message to send a document to a storage system. The production does the following:

1. The DICOM modality establishes an association and sends a C-STORE request message to the production through the duplex business service.
2. The business process checks the connection to the output storage system and establishes the association if needed.
3. The production recognizes the request, stores the originating message ID, and forwards the DICOM document message to the storage system through the duplex business operation.
4. The business operation returns the C-STORE response message from the storage system to the business process.
5. The production sends the response message back to the modality with the originating message ID.

The following steps outline the procedure to add this type of interface to a production:

1. Create a generic production by clicking **Create New Production** on the **Production Configuration** page. See Creating and Configuring a Production.
2. In Studio, modify the production definition to add the StorageLocation property and compile the production. See Configuring a DICOM Production to Control the Storage Location for details.
3. Add a DICOM duplex business service to the production using the EnsLib.DICOM.Service.TCP class.
4. Configure the DICOM business service settings specifically for a routing to storage production.
5. Create a business process class that routes the incoming DICOM document to a storage system from the incoming C-STORE-RQ message and passes the returned C-STORE-RSP message back to the originating system.
6. Add a DICOM business process using the custom class from the previous step.
7. Add a DICOM business operation.
8. Test the production to verify that it receives a request message for a worklist and sends the appropriate response message documents back.

You can view the class code of Demo.DICOM.Production.Storage.cls using Studio to see the production details.

5.1 Configuring the DICOM Routing to Storage Business Service

You can configure a DICOM business service by clicking it in the diagram on the **Production Configuration** page. Configuring a DICOM Duplex Business Host describes the details. This section describes the settings specific to the business service in the demonstration storage production.

Duplex Target Config Names

Specify the configuration item within the production to which the business service should send any DICOM documents that it receives.

The Demo.DICOM.Production.Storage production uses the business process based on the class, Demo.DICOM.Process.Storage, which is described in Creating a Business Process Class for a DICOM Storage Production. This is a custom process that contains the logic for processing DICOM messages from the input modality and routing them to a storage system.

LocalAET

The called Application Entity Title (AET) that the remote DICOM peer uses to communicate with the production. This corresponds to the **Called AET** you use when defining an association in an interoperability-enabled namespace.

When you run the Demo.DICOM.Production.Storage production for the first time, it creates the DICOM association necessary to connect to a test DICOM application for this demonstration production. You can view this association in the production's namespace from the **Interoperability** > **Interoperate** > **DICOM** > **DICOM Settings** page.

The EnsLib.DICOM.Service.TCP service of the demo production uses `ENS-SCP` for the value of **LocalAET**.

RemoteAET

The calling Application Entity Title(s) of a remote DICOM peer.

When the adapter is in the role of Service Class Provider (SCP, server) it contains a comma-delimited list of names of the DICOM peers which are allowed to connect. A name can either be a literal string or a pattern/substitution.

The EnsLib.DICOM.Service.TCP service of the demo production uses `JD-SCU` for the value of **RemoteAET**.

5.2 Creating a Business Process Class for a DICOM Storage Production

In the sample Demo.DICOM.Production.Storage, the production uses a sample custom business process class, Demo.DICOM.Process.Storage, which demonstrates how to handle DICOM C-STORE request messages, send the DICOM document to storage and return a d C-STORE response message.

This custom class extends EnsLib.DICOM.Process, which is the superclass for all user-defined DICOM business processes. See its entry in the *Class Reference* for detailed information.

For information, see Developing Custom Business Processes in *Developing Productions*.

The Demo.DICOM.Process.Storage class uses the parameter *SETTINGS* to expose a new property, OperationDuplexName, for configuration.

Because of the duplex nature of DICOM communication, the business process must keep track of what is happening outside of the process. You can accomplish this using a context variable for the *state* of the process. The demonstration business process creates a CurrentState property for this purpose.

You can view the class code for Demo.DICOM.Process.Storage.cls using Studio to see the processing details.

5.3 Testing the DICOM Routing to Storage Production

Once you have working associations and created a production, you can attempt to process valid DICOM message documents through the production. The sample DICOM demonstration production was developed using third-party software specifically developed for testing DICOM processing. You can use any of the many available software products, or test with your actual DICOM modality data.

6

Sample DICOM File Storage Production

The sample production named Demo.DICOM.Production.StorageFile demonstrates a basic DICOM function, sending DICOM documents from a file to a DICOM compatible storage system. This production contains the business service EnsLib.DICOM.Service.File, the business process Demo.DICOM.Process.StorageFile, and the business operation EnsLib.DICOM.Operation.TCP.

In this scenario, the production receives a file containing DICOM-formatted messages and sends those messages to a storage system. The production does the following:

1. An business service receives a file with the .dcm extension as input and sends the messages to the DICOM business process. The DICOM message documents contain C-STORE requests.
2. The business process checks the connection to the output storage system and establishes the association to the storage system if needed.
3. The production recognizes the request, stores the originating message ID, and forwards the DICOM document message to the storage system through the duplex business operation.
4. The business operation returns the C-STORE response message from the storage system to the business process.

The following steps outline the procedure to add this type of interface to a production:

1. Create a generic production by clicking **Create New Production** on the **Production Configuration** page. See Creating and Configuring a Production.
2. In Studio, modify the production definition to add the StorageLocation property and compile the production. See Configuring a DICOM Production to Control the Storage Location for details.
3. Add a DICOM file business service to the production using the EnsLib.DICOM.Service.File class.
4. Configure the DICOM business service settings specifically for a file storage production.
5. Create a business process class that routes the incoming DICOM document to a storage system from the incoming C-STORE-RQ message and returns the C-STORE-RSP message.
6. Add a DICOM business process using the custom class from the previous step.
7. Add a DICOM business operation to the production using the EnsLib.DICOM.Operation.TCP class.
8. Test the production to verify that it receives a storage request message for a DICOM document and sends the document to the storage system.

You can view the class code of Demo.DICOM.Production.StorageFile.cls using Studio to see the production details.

6.1 Configuring the DICOM Routing to Storage Business Service

You can configure a DICOM business service by clicking it in the diagram on the **Production Configuration** page. Configuring a DICOM Duplex Business Host describes the details. This section describes the settings specific to the business service in the demonstration storage production.

TargetConfigName

Specify the configuration item within the production to which the business service should send any DICOM documents that it receives.

The Demo.DICOM.Production.StorageFile production uses the business process based on the class, Demo.DICOM.Process.StorageFile. This is a custom process that contains the logic for processing DICOM messages from the input file and routing them to a storage system.

File Path

Full pathname of the directory in which to look for files. This directory must exist, and it must be accessible through the file system on the local system.

File Spec

Filename or wildcard file specification for file(s) to retrieve. For the wildcard specification, use the convention that is appropriate for the operating system on the local system.

The Demo.DICOM.Production.StorageFile production uses `*.dcm` file extension.

6.2 Testing the DICOM File Storage Production

Once you have working associations and created a production, you can attempt to process valid DICOM message documents through the production. The sample DICOM demonstration production was developed using third-party software specifically developed for testing DICOM processing. You can use any of the many available software products, or test with your actual DICOM modality data.

This section shows portions of the Event Log, Message Browser, Message Details, and Message Contents pages in the Management Portal of the Demo.DICOM.Production.StorageFile production running and accepting DICOM messages from a file and passing them to a storage system that supports store requests.

7

Sample DICOM Router Productions

There are two sample productions that demonstrate a synchronous and an asynchronous router. These are named Demo.DICOM.Production.Router and Demo.DICOM.Production.AsyncRouter. These productions demonstrate how a user can route DICOM documents. Typically, DICOM uses synchronous messaging, but, in some cases, there are advantages to using an asynchronous router.

7.1 Synchronous DICOM Router

The synchronous DICOM router production, Demo.DICOM.Production.Router, completes sending a message to the targets before it can accept another message. The following describes the flow of messages in the Router production:

- The Remote Service Class User (SCU) connects to the Listener service and sends a message.
- When the Listener service receives a message, it sends a message to the Dispatcher.
- The Dispatcher sends a message to each filer. In the sample production, there are two filers.
- Each Filer sends the messages to an operation, which establishes an association with the appropriate PACS system and sends the document to it.
- When the Dispatcher receives a successful response from each of the operations, it sends a success response message to the Listener, which can then accept a new message.

7.2 Asynchronous DICOM Router

The asynchronous DICOM message Listener can continue to receive messages while the production is processing previous messages. The following describes the flow of messages in the AsyncRouter production:

- The Remote Service Class User (SCU) connects to the Listener service and starts sending messages.
- When the Listener service receives a message, it immediately replies with the C-STORE-RSP and accepts responsibility for that message. It sends a message to the AsyncDispatcher.
- The AsyncDispatcher process queues up an entry in the table Demo_DICOM.BatchedDocument for each Filer. In the sample production, there are two filers: AsyncFiler1 and AsyncFiler2. It uses the Rule Class SendToTargetFiler method to determine if the document should be sent to each filer.
  - If a document is intended for the filer, the AsyncDispatcher sets the DocumentStatus to "Queued".

  - If a document is not intended for the filer, the AsyncDispatcher sets the DocumentStatus to "Ignored".

- Once the association between the remote SCU and the Listener service is released, the AsyncDispatcher sends a message to each of the Filers for which there are documents queued up in that session.
- Each Filer sends the messages to an operation, which establishes an association with the appropriate PACS system and sends each of the documents to it.
- When the production receives the C-STORE-RSP response, it marks the entry in the Demo_DICOM.BatchedDocument table with a DocumentStatus of either "Delivered" if the status was a success or "Errored" if not, in which case the ErrorComment is stored as the entry's Remark.
- If the Remove Completed Entries setting for the Filer is checked, the filer removes all delivered entries in the current session once all the C-STORE-RSP messages have been received. However, if the remote SCP sends back an ABORT or if the association encounters an error before all the documents for the Filer in the current session have been delivered, then the filer resends all documents in the current session until all the documents are successfully delivered or until it reaches the specified Retry Count. If the filer reaches the specified Retry Count without successfully sending all the documents, then the filer requeues all the documents in the current session and issues an alert. The alert message is sent to Ens.Alert, which send an email alert.

A

DICOM Transfer Syntax

The following table shows the available Transfer Syntax values and their corresponding unique identifiers for DICOM standard 3.6.

Name	UID
Implicit VR - Little Endian	1.2.840.10008.1.2
Explicit VR - Little Endian	1.2.840.10008.1.2.1
Deflated Explicit VR - Little Endian	1.2.840.10008.1.2.1.99
Explicit VR - Big Endian	1.2.840.10008.1.2.2
MPEG2 Main Profile/Main Level	1.2.840.10008.1.2.4.100
MPEG2 Main Profile/High Level	1.2.840.10008.1.2.4.101
MPEG-4 AVC/H.264 High Profile/Level 4.1	1.2.840.10008.1.2.4.102
MPEG-4 AVC/H.264 BD-compatible High Profile / Level 4.1	1.2.840.10008.1.2.4.103
MPEG-4 AVC/H.264 High Profile / Level 4.2 For 2D Video	1.2.840.10008.1.2.4.104
MPEG-4 AVC/H.264 High Profile / Level 4.2 For 3D Video	1.2.840.10008.1.2.4.105
MPEG-4 AVC/H.264 Stereo High Profile / Level 4.2	1.2.840.10008.1.2.4.106
HEVC/H.265 Main Profile / Level 5.1	1.2.840.10008.1.2.4.107
HEVC/H.265 Main 10 Profile / Level 5.1	1.2.840.10008.1.2.4.108
JPEG Baseline (Process 1)	1.2.840.10008.1.2.4.50
JPEG Extended (Process 2 & 4)	1.2.840.10008.1.2.4.51
JPEG Extended (Process 3 & 5)	1.2.840.10008.1.2.4.52
JPEG Spectral Selection, Non-Hierarchical (Process 6 & 8)	1.2.840.10008.1.2.4.53
JPEG Spectral Selection, Non-Hierarchical (Process 7 & 9)	1.2.840.10008.1.2.4.54
JPEG Full Progression, Non-Hierarchical (Process 10 & 12)	1.2.840.10008.1.2.4.55
JPEG Full Progression, Non-Hierarchical (Process 11 & 13)	1.2.840.10008.1.2.4.56
JPEG Lossless, Non-Hierarchical (Process 14)	1.2.840.10008.1.2.4.57
JPEG Lossless, Non-Hierarchical (Process 15)	1.2.840.10008.1.2.4.58

Name	UID
JPEG Extended, Hierarchical (Process 16 & 18)	1.2.840.10008.1.2.4.59
JPEG Extended, Hierarchical (Process 17 & 19)	1.2.840.10008.1.2.4.60
JPEG Spectral Selection, Hierarchical (Process 20 & 22)	1.2.840.10008.1.2.4.61
JPEG Spectral Selection, Hierarchical (Process 21 & 23)	1.2.840.10008.1.2.4.62
JPEG Full Progression, Hierarchical (Process 24 & 26)	1.2.840.10008.1.2.4.63
JPEG Full Progression, Hierarchical (Process 25 & 27)	1.2.840.10008.1.2.4.64
JPEG Lossless, Hierarchical (Process 28)	1.2.840.10008.1.2.4.65
JPEG Lossless, Hierarchical (Process 29)	1.2.840.10008.1.2.4.66
JPEG Lossless, Non-Hierarchical, First-Order Prediction (Process 14 [Selection Value 1])	1.2.840.10008.1.2.4.70
JPEG-LS Lossless	1.2.840.10008.1.2.4.80
JPEG-LS Near Lossless	1.2.840.10008.1.2.4.81
JPEG 2000 Lossless	1.2.840.10008.1.2.4.90
JPEG 2000 Either	1.2.840.10008.1.2.4.91
JPEG 2000 Lossless Multi-Component	1.2.840.10008.1.2.4.92
JPEG 2000 Either Multi-Component	1.2.840.10008.1.2.4.93
JPIP Referenced	1.2.840.10008.1.2.4.94
JPIP Referenced Deflate	1.2.840.10008.1.2.4.95
RLE Lossless	1.2.840.10008.1.2.5
RFC 2557 MIME Encapsulation	1.2.840.10008.1.2.6.1
XML Encoding	1.2.840.10008.1.2.6.2
Papyrus 3 Implicit VR Little Endian	1.2.840.10008.1.20

DICOM Glossary of Terms

abstract syntax

Describes the services that DICOM applications can render to each other; they encode SOPs supported on the communicating AEs.

Generally equivalent to a **SOP class**, the specification used to define the information to exchange in a message; does not represent a specific instance of the data object, but rather a class of similar data objects that have the same properties. Examples: MR image storage, CT image storage, MWL.

application entity (AE)

Uniquely identifies a DICOM device or program. Typically labeled with numbers and uppercase characters only.

An end point of a DICOM information exchange, including the DICOM network or media interface software; that is, the software that sends or receives DICOM information objects or messages. A single device may have multiple application entities.

application entity title (AET)

The externally known name of an *application entity*, used to identify a DICOM application to other DICOM applications on the network.

association

A network communication channel set up between *application entities*; the DICOM network data exchange between SCU and SCP peers. Each network transfer begins with an association establishment — a DICOM handshake.

DICOM association rules define the low-level protocols for DICOM network connectivity to ensure that two communicating DICOM applications (AEs) are compatible and transfer data in a well-defined format and order.

called AE title

This parameter identifies the application entity that shall contain the intended acceptor of the A-ASSOCIATE service. It is based on the destination DICOM application name. The called AE title may or may not be the same as the receiver address present in DICOM messages exchanged over the association. Note: It is the responsibility of the UL User that received the A-ASSOCIATE-RQ to verify whether the Called AE Title is its (or one of its) DICOM Application Name(s).

calling AE title

This parameter identifies the Application Entity (AE) that shall contain the requestor of the A-ASSOCIATE service. It is based on the Source DICOM Application Name. The relationship between DICOM Application Names and AE titles is specified in Annex C. The Calling AE title may or may not be the same as the Initiator Address present in DICOM Messages exchanged over the association. Note: It is the responsibility of the UL User that received the A-ASSOCIATE-RQ to verify whether the Calling AE Title is one of its known remote DICOM Application Names.

data dictionary element

A unit of information as defined by a single entry in the data dictionary. An encoded Information Object Definition (IOD) Attribute that is composed of, at a minimum, three fields: a Data Element Tag, a Value Length, and a Value Field. For some specific Transfer Syntaxes, a Data Element also contains a VR Field where the Value Representation of that Data Element is specified explicitly.

modality

A device that communicates using the DICOM standard.

picture archiving and communication system (PACS)

A system that acquires, transmits, stores, retrieves, and displays digital images and related patient information from a variety of imaging sources and communicates the information over a network.

presentation context

Set of DICOM network services used over an *association*, as negotiated between *application entities*; includes *abstract syntaxes* and *transfer syntaxes*.

Each network transfer begins with an association establishment (DICOM handshake), when the two connecting applications exchange information about each other. This information is called the presentation context; if the two applications can match their contexts, they can connect and start SCU-SCP processing.

radiology information system (RIS)

System used by radiology departments to store, manipulate, and distribute patient radiological data and imagery. The system generally comprises of patient tracking and scheduling, result reporting, and image tracking capabilities.

service class provider (SCP)

Role of an *application entity* that provides a DICOM network service; typically, a server that performs operations requested by another *application entity* (*Service Class User*). Examples: Picture Archiving and Communication System (image storage SCP, and image query/retrieve SCP), Radiology Information System (modality worklist SCP).

service class user (SCU)

Role of an *application entity* that uses a DICOM network service; typically, a client. Examples: imaging modality (image storage SCU, and modality worklist SCU), imaging workstation (image query/retrieve SCU).

service-object pair (SOP)

The pairing of compatible DICOM data and commands.

services and objects. IODs define DICOM data and DIMSE services define DICOM commands

service-object pair (SOP) class

The specification of the network or media transfer (service) of a particular type of data (object); the fundamental unit of DICOM interoperability specification. Examples: verification, CT image storage, print job.

service object pair (SOP) instance

An information object; a specific occurrence of information exchanged in a *SOP Class*. Example: a specific x-ray image.

transfer syntax (standard and private)

Encoding used for exchange of DICOM information objects and messages. Allows application entities to unambiguously negotiate the encoding techniques they are able to support (for example, data element structure, byte ordering, compression), thereby allowing these application entities to communicate.

See DICOM Transfer Syntax for a table of standard values. Manufacturers may also define private transfer syntax values for use among their own devices.

unique identifier (UID)

A globally unique dotted decimal string of characters that identifies a wide variety of items. Identifies object instances.

DICOM networking uses UIDs to encode various transactions types. Examples: SOP class and transfer syntax values.

value multiplicity (VM)

Specifies the number of values that can be encoded in the value field of that data element.

value representation (VR)

Format type of an individual DICOM data element, such as text, an integer, a person's name, or a code. DICOM information objects can be transmitted with either explicit identification of the type of each data element (Explicit VR), or without explicit identification (Implicit VR). With Implicit VR, the receiving application must use a DICOM data dictionary to look up the format of each data element.