



Medical Imaging in IDL

IDL Version 7.1
May 2009 Edition
Copyright © ITT Visual Information Solutions
All Rights Reserved

Restricted Rights Notice

The IDL®, IDL Analyst™, ENVI®, and ENVI Zoom™ software programs and the accompanying procedures, functions, and documentation described herein are sold under license agreement. Their use, duplication, and disclosure are subject to the restrictions stated in the license agreement. ITT Visual Information Solutions reserves the right to make changes to this document at any time and without notice.

Limitation of Warranty

ITT Visual Information Solutions makes no warranties, either express or implied, as to any matter not expressly set forth in the license agreement, including without limitation the condition of the software, merchantability, or fitness for any particular purpose.

ITT Visual Information Solutions shall not be liable for any direct, consequential, or other damages suffered by the Licensee or any others resulting from use of the software packages or their documentation.

Permission to Reproduce this Manual

If you are a licensed user of these products, ITT Visual Information Solutions grants you a limited, nontransferable license to reproduce this particular document provided such copies are for your use only and are not sold or distributed to third parties. All such copies must contain the title page and this notice page in their entirety.

Export Control Information

The software and associated documentation are subject to U.S. export controls including the United States Export Administration Regulations. The recipient is responsible for ensuring compliance with all applicable U.S. export control laws and regulations. These laws include restrictions on destinations, end users, and end use.

Acknowledgments

ENVI® and IDL® are registered trademarks of ITT Corporation, registered in the United States Patent and Trademark Office. ENVI Zoom™ is a trademark of ITT Visual Information Solutions.

Numerical Recipes™ is a trademark of Numerical Recipes Software. Numerical Recipes routines are used by permission.

GRG2™ is a trademark of Windward Technologies, Inc. The GRG2 software for nonlinear optimization is used by permission.

NCSA Hierarchical Data Format (HDF) Software Library and Utilities. Copyright © 1988–2001, The Board of Trustees of the University of Illinois. All rights reserved.

NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities. Copyright © 1998–2002, by the Board of Trustees of the University of Illinois. All rights reserved.

CDF Library. Copyright © 2002, National Space Science Data Center, NASA/Goddard Space Flight Center.

NetCDF Library. Copyright © 1993–1999, University Corporation for Atmospheric Research/Unidata.

HDF EOS Library. Copyright © 1996, Hughes and Applied Research Corporation.

SMACC. Copyright © 2000–2004, Spectral Sciences, Inc. and ITT Visual Information Solutions. All rights reserved.

This software is based in part on the work of the Independent JPEG Group.

Portions of this software are copyrighted by DataDirect Technologies, © 1991–2003.

BandMax®. Copyright © 2003, The Galileo Group Inc.

Portions of this computer program are copyright © 1995–1999, LizardTech, Inc. All rights reserved. MrSID is protected by U.S. Patent No. 5,710,835. Foreign Patents Pending.

Portions of this software were developed using Unisearch's Kakadu software, for which ITT has a commercial license. Kakadu Software. Copyright © 2001. The University of New South Wales, UNSW, Sydney NSW 2052, Australia, and Unisearch Ltd, Australia.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

MODTRAN is licensed from the United States of America under U.S. Patent No. 5,315,513 and U.S. Patent No. 5,884,226.

FLAASH is licensed from Spectral Sciences, Inc. under a U.S. Patent Pending.

Portions of this software are copyrighted by Merge Technologies Incorporated.

Support Vector Machine (SVM) is based on the LIBSVM library written by Chih-Chung Chang and Chih-Jen Lin (<http://www.csie.ntu.edu.tw/~cjlin/libsvm>), adapted by ITT Visual Information Solutions for remote sensing image supervised classification purposes.

IDL Wavelet Toolkit Copyright © 2002, Christopher Torrence.

IMSL is a trademark of Visual Numerics, Inc. Copyright © 1970–2006 by Visual Numerics, Inc. All Rights Reserved.

Other trademarks and registered trademarks are the property of the respective trademark holders.

Portions of this software are copyrighted by Merge Technologies Incorporated.





Contents

Chapter 1	
Expanded DICOM Support in IDL	9
Expanded DICOM Functionality in IDL	10
Using DICOM Network Services	10
Reading and Writing DICOM Files	10
Platform Support and Licensing	11
Supported Platforms	11
Licensing Requirements	11
Compliance with the DICOM Standard	12
Chapter 2	
Using IDL DICOM Network Services	13
Overview of DICOM Network Services	14
Network Performance and Virus Protection Software	15
Determining Configuration Parameters	16
Determining the IP Address	16

Determining TPC/IP Port Numbers	16
Starting the Network Services Utility	18
Local Versus System Configuration	18
Starting the DICOM Network Services Utility in Local Mode	19
Starting the DICOM Network Services Utility in System Mode	20
Completing Required Setup Tasks	21
Configuring Your System to Receive Files	22
Configuring an Application Entity	25
Default Application Entities	25
Defining a New Application Entity	26
Modifying an Application Entity	28
Deleting an Application Entity	29
Defining a Machine to Be Queried	30
Querying a Remote Machine	33
Configure Query SCU Service Properties	33
Query the Remote Node	34
Build or Modify a Custom Query	36
Retrieve Files from a Remote Node	38
Troubleshooting a Retrieval Operation	40
Sending Files to a Remote Destination	43
Defining the Remote Storage SCP Node	44
Sending Files to a Remote Machine	44
Troubleshooting a Send Operation	47
Returning Connection Status with Echo	48
Troubleshooting Echo Errors	48
About the Storage SCP Service	49
Storage SCP Service Permissions	49
Storage SCP Service Log Files	50
Managing the Storage SCP Service	51
Starting and Stopping the Service Outside the Utility	52
DICOM Network Services User Interface	53
System Mode Interface	53
Local Mode Interface	54

Chapter 3	
IDL DICOM Reference	57
DICOMEX_GETCONFIGFILEPATH	58
DICOMEX_GETSTORSCPDIR	60
DICOMEX_NET	62
IDLffDicomEx	64
File Compression and Transfer Syntax Support	68
IDLffDicomEx Properties	70
IDLffDicomEx::AddGroup	86
IDLffDicomEx::AddPrivateGroup	93
IDLffDicomEx::AddPrivateSequence	100
IDLffDicomEx::AddSequence	104
IDLffDicomEx::ChangeTransferSyntax	108
IDLffDicomEx::Cleanup	114
IDLffDicomEx::Commit	115
IDLffDicomEx::CopyTags	116
IDLffDicomEx::EnumerateTags	121
IDLffDicomEx::GetDescription	127
IDLffDicomEx::GetPixelData	129
IDLffDicomEx::GetPrivateValue	136
IDLffDicomEx::GetPrivateValueCount	140
IDLffDicomEx::GetPrivateValueLength	143
IDLffDicomEx::GetPrivateVR	148
IDLffDicomEx::GetProperty	151
IDLffDicomEx::GetValue	152
IDLffDicomEx::GetValueCount	155
IDLffDicomEx::GetValueLength	157
IDLffDicomEx::GetVR	161
IDLffDicomEx::Init	163
IDLffDicomEx::QueryPrivateValue	176
IDLffDicomEx::QueryValue	179
IDLffDicomEx::SetPixelData	182
IDLffDicomEx::SetPrivateValue	191
IDLffDicomEx::SetProperty	199
IDLffDicomEx::SetValue	201
IDLffDicomExCfg	212

IDLffDicomExCfg Properties	215
IDLffDicomExCfg::Cleanup	216
IDLffDicomExCfg::Commit	217
IDLffDicomExCfg::Echo	218
IDLffDicomExCfg::GetApplicationEntities	220
IDLffDicomExCfg::GetApplicationEntity	222
IDLffDicomExCfg::GetServiceLists	223
IDLffDicomExCfg::GetServiceTypes	224
IDLffDicomExCfg::GetValue	225
IDLffDicomExCfg::Init	227
IDLffDicomExCfg::IsDirty	230
IDLffDicomExCfg::RemoveApplicationEntity	231
IDLffDicomExCfg::SetApplicationEntity	232
IDLffDicomExCfg::SetValue	234
IDLffDicomExCfg::StorageScpService	236
IDLffDicomExQuery	239
IDLffDicomExQuery Properties	242
Patient Name and Name Sub-field Properties	257
IDLffDicomExQuery::Cleanup	260
IDLffDicomExQuery::ClearProperties	261
IDLffDicomExQuery::GetProperty	262
IDLffDicomExQuery::Init	263
IDLffDicomExQuery::Query	265
IDLffDicomExQuery::QueryModelsSupported	270
IDLffDicomExQuery::Retrieve	273
IDLffDicomExQuery::SetProperty	282
IDLffDicomExStorScu	283
IDLffDicomExStorScu Properties	285
IDLffDicomExStorScu::Cleanup	289
IDLffDicomExStorScu::ClearProperties	290
IDLffDicomExStorScu::GetProperty	291
IDLffDicomExStorScu::Init	292
IDLffDicomExStorScu::Send	294
IDLffDicomExStorScu::SetProperty	298

Appendix A	
DICOM Resources	299
DICOM Attributes	300
Value Representations	372
Appendix B	
IDL DICOM Quick Reference	387
Alphabetical Listing	388
Index	393



Chapter 1

Expanded DICOM Support in IDL

This chapter provides introductory information about DICOM support in IDL including general information and licensing requirements.

Expanded DICOM Functionality in IDL . . .	10	Compliance with the DICOM Standard . . .	12
Platform Support and Licensing	11		

Expanded DICOM Functionality in IDL

Using DICOM Network Services

IDL DICOM Network Services supports SCU (service class user) and SCP (service class provider) network functionality including Echo SCU, Query/Retrieve SCU, Storage SCU, and Storage SCP.

Using the **DICOM Network Services** utility, you can:

- Query a remote machine and retrieve selected DICOM files
- Build custom queries
- Send files to a remote destination
- Troubleshoot network connections using echo

See [Chapter 2, “Using IDL DICOM Network Services”](#) for complete details on using the **DICOM Network Services** utility. The section, [“Overview of DICOM Network Services”](#) on page 14, provides more detailed introductory information.

Reading and Writing DICOM Files

The IDLffDicomEx object, described in [“IDLffDicomEx”](#) on page 64, greatly expands IDL’s DICOM file access capabilities. Previously, read-only DICOM support was provided through the IDLffDICOM object. The IDLffDicomEx object offers the following enhancements over the IDLffDICOM object:

- Ability to read from and write to DICOM files. Using the IDLffDicomEx object, you can read, clone, or create a new DICOM file. The IDLffDICOM object only supports reading DICOM files.
- Ability to read and write both public and private attributes including sequences and sets of repeating tags within sequences (groups).
- Ability to read and write compressed DICOM files on Windows and UNIX platforms. See [“File Compression and Transfer Syntax Support”](#) on page 68 for more information.
- Additional SOP class support. See [“Compliance with the DICOM Standard”](#) on page 12 for information about the conformance statement, which contains the most current list of supported SOP classes.
- Ability to copy DICOM attributes from one file to another, and view all the DICOM attributes contained within a file.

Platform Support and Licensing

Supported Platforms

See [“Feature Support by Operating System”](#) (Chapter 2, *What’s New in IDL 7.1*) for a current list of operating systems that support the expanded DICOM support in IDL.

Licensing Requirements

For more information about the following items, contact your ITT Visual Information Solutions sales representative or technical support. To redistribute applications that contain the **DICOM Network Services** utility or that use the IDLffDicomEx object functionality, an embedded license or a runtime license can be purchased for end users, or users can be instructed to purchase their own licenses. Please refer to [Chapter 23, “Distributing Runtime Mode Applications”](#) (*Application Programming*) for more details about the different licensing options.

DICOM Network Services

An additional-cost license is required to access IDL’s DICOM Network Services. Your license purchase for DICOM Network Services *includes* the license required for DICOM Read/Write support.

DICOM Read/Write

The IDLffDicomEx object greatly expands IDL’s DICOM capabilities and requires an additional-cost license key to access the functionality.

Compliance with the DICOM Standard

Information on the compliance of IDL's DICOM Network Services and the IDLffDicomEx object to the DICOM standard are provided in a conformance statement. See www.itvis.com/idl/dicom for a current DICOM Conformance Statement.

Note

IDL, the IDL DICOM toolkit, and the contents of the IDL documentation are not marketed as products with FDA Section 510(k) premarket notification nor have they been reviewed by any other regulatory agency from any other country. As is, IDL, the IDL DICOM toolkit, and the contents of the IDL documentation, are intended for educational, research, and third party application developers. If IDL and the DICOM toolkit, or information derived from them, are to be used as part of an application for clinical purposes, such as for treating or diagnosing human subjects, it is the responsibility of the LICENSEE to determine the need for and to obtain the appropriate regulatory reviews before such usage. ITT Visual Information Solutions maintains engineering and quality control records suitable for review by a regulatory agency. We reserve the right to recover costs incurred during a regulatory review from the LICENSEE. We shall not be liable for any damages arising out of the use of IDL or the DICOM toolkit by any party for any purpose.

Chapter 2

Using IDL DICOM Network Services

This chapter provides information on how to use the **DICOM Network Services** utility to configure Application Entities, and how to perform query, retrieve, storage, and send operations.

Overview of DICOM Network Services . . .	14	Querying a Remote Machine	33
Determining Configuration Parameters	16	Sending Files to a Remote Destination . . .	43
Starting the Network Services Utility	18	Returning Connection Status with Echo . . .	48
Completing Required Setup Tasks	21	About the Storage SCP Service	49
Configuring Your System to Receive Files . .	22	Managing the Storage SCP Service	51
Configuring an Application Entity	25	DICOM Network Services User Interface . .	53
Defining a Machine to Be Queried	30		

Overview of DICOM Network Services

IDL DICOM Network Services supports SCU (service class user) and SCP (service class provider) network functionality including:

- Echo SCU
- Query/Retrieve SCU
- Storage SCU
- Storage SCP

Note

This feature requires an additional-cost license key to access the functionality. For more information, contact your ITT Visual Information Solutions sales representative or technical support.

Three broad areas of functionality in the **DICOM Network Services** utility that support these services are:

- **Configuration** — this tab allows you to configure properties of Application Entities (AEs), echo a remote node and manage the Storage SCP service. AEs can be configured in a local **Configuration** tab or a system **Configuration** tab. See [“Local Versus System Configuration”](#) on page 18 for important information on the distinction.
- **Query Retrieve SCU** — this tab allows you to query a remote database of files and retrieve selected files. You can also see the query/retrieve operation status, and optionally define custom query parameters. You will need to configure an Application Entity for a Query SCP node before performing a query operation. See [“Defining a Machine to Be Queried”](#) on page 30 for details.
- **Storage SCU** — this tab allows you to select a node to which files are sent and select the files to send. You will need to configure an Application Entity for the remote device that supports Storage SCP. See [“Sending Files to a Remote Destination”](#) on page 43 for details.

See [“DICOM Network Services User Interface”](#) on page 53 for screen shots of each of the available tabs.

Note

See “[Compliance with the DICOM Standard](#)” on page 12 for important information regarding the use of the **DICOM Network Services** utility for clinical research or diagnostic purposes.

Network Performance and Virus Protection Software

Transferring files over a network can be a time consuming process. To optimize file transfer speed, you may want to disable any virus scanning software when performing query, retrieve or send operations.

Determining Configuration Parameters

When defining Application Entities, you will need the host name (or IP address), and port number of your machine and any remote machines you will connect to, in addition to other information. This section provides information on how to retrieve this information.

Determining the IP Address

Determine the IP address of your machine by completing the following instructions for your platform.

Windows

1. Click **Start** → **Programs** → **Accessories** → **Command Prompt** to open a DOS window.
2. At the prompt, type `ipconfig`. This returns the IP address of the local machine.

UNIX

At the UNIX prompt, type `/sbin/ifconfig -a` to return the IP (ether) address or type `netstat -i` to return the hostname associated with the IP address.

Mac OS X

Open **Finder**. Under **Applications/Utilities** launch **Network Utility**. Select the **Info** tab and then select the network interface from the pull down menu.

Determining TPC/IP Port Numbers

Use the information in the following section to determine available TCP/IP ports.

Windows

1. Click **Start** → **Programs** → **Accessories** → **Command Prompt** to open a DOS window.
2. At the prompt, type `netstat -a`. View the results to determine port availability and status.

UNIX

Use the `netstat -a` command to return port information.

Mac OS X

Open **Finder**. Under **Applications/Utilities** launch **Network Utility**. Select the **Netstat** tab and select the information you want to return.

Starting the Network Services Utility

The **DICOM Network Services** utility allows you to create and store Application Entity definitions in a local configuration file or a system configuration file. This distinction is only required when you define an Application Entity associated with the Storage SCP service, or configure properties of this service. This information must be stored in the system configuration file, which requires starting the utility with a special **SYSTEM** keyword. Application Entities using other service types are typically defined in the local configuration file (available when the utility is started without any keywords). When started in this manner, the utility also provides access to query, retrieve and send operations. See the following section for details on local and system modes.

Local Versus System Configuration

There are two modes of configuration available; local configuration and system configuration. Underlying each set of user-defined settings is an `.xml` file. Changes made to Application Entities (AEs) defined in the system configuration and local configuration mode of the **DICOM Network Services** utility are reflected in these files.

Note

Always use the **DICOM Network Services** utility to make changes to the configuration files. The raw `.xml` files should never be modified directly.

The system configuration portion of the **DICOM Network Services** utility corresponds to a single system configuration file that is used by the Storage SCP service. The Storage SCP service listens for incoming files and is typically started at boot time, which means the service is not associated with a specific user. The properties of the Storage SCP service can only be set when the **DICOM Network Service** utility is started in system configuration mode.

When you share a single installation of IDL among a number of users, changes made in the system configuration file are visible to all users. In such a network, it is likely that only system-wide configuration information will be entered in system mode, and you will enter individual Application Entity definitions in your local configuration file by starting the utility in local mode. Each user has a local configuration file.

It is important to understand when changes made in system configuration mode are propagated to each user's local configuration file:

- The first time the **DICOM Network Services** utility is accessed in IDL, the local configuration file is populated with AEs defined in the system configuration file.
- After the first time the utility is started, the local file is only overwritten with system configuration information if the local directory is deleted. In such a case, the local configuration settings will be recreated from the system configuration settings.

If having locally configured AEs that are not duplicated in the system file is acceptable, then there is no need to modify the system file except to define Storage SCP service parameters. This must be done in the system configuration file. See [“Configuring Your System to Receive Files”](#) on page 22 for details.

Tip

You can use the `DICOMEX_GETCONFIGFILEPATH` function to return the location of either configuration file. See [“DICOMEX_GETCONFIGFILEPATH”](#) on page 58 for details.

Starting the DICOM Network Services Utility in Local Mode

Enter the following at the IDL command prompt to launch the **DICOM Network Services** utility in local configuration mode:

```
DICOMEX_NET
```

This provides access to the local **Configuration** tab, the **Query Retrieve SCU** tab and the **Storage SCU** tab.

Note

On Windows, you can also select **Start** → **Programs** → **IDL x.x** → **DICOM Network Services** and click the **Local** button to start the utility in local configuration mode. On Macintosh, double-click `DicomExNetStartup` in the root IDL folder.

Starting the DICOM Network Services Utility in System Mode

Use the **SYSTEM** keyword to launch the **DICOM Network Services** utility in system configuration mode:

```
DICOMEX_NET, /SYSTEM
```

This provides access to the system **Configuration** tab.

Note

On Windows, you can also select **Start** → **Programs** → **IDL x.x** → **DICOM Network Services** and click the **System** button to start the utility in system configuration mode. On Macintosh, double-click `DicomExNetStartup` in the root IDL folder.

Completing Required Setup Tasks

Before using the **DICOM Network Services** utility to query, retrieve or transfer files across a network, you need to define Application Entities (AEs) for the needed DICOM services. AEs define the machine characteristics where a local or remote DICOM network service exists.

The exact steps depend on the type of services you will be using. See the following for more information:

- [“Configuring Your System to Receive Files”](#) on page 22 — describes how to define an Application Entity associated with the DICOM Storage SCP service which includes setting a directory location where returned files are stored. This change requires restarting the Storage SCP Service.
- [“Defining a Machine to Be Queried”](#) on page 30 — describes how to configure and define a Query SCP node, a machine from which you will request files.
- [“Sending Files to a Remote Destination”](#) on page 43 — describes how send files from your system (one that supports Storage SCU) to a remote machine (one that supports Storage SCP).

After defining the necessary Application Entities, you can select among the defined AEs, choosing one for each service. You can also test connections to remote nodes using echo, create custom queries, retrieve and send files, and manage the Storage SCP Service state.

Configuring Your System to Receive Files

Storage is the process by which DICOM files are transferred from one DICOM device to another. In order to receive files, you need to specify a Storage SCP Application Entity for the machine that will store the files that are returned. This entity uses the Storage SCP Service, which listens at a TCP/IP port for incoming DICOM files and writes them to a defined disk directory.

Note

If you are running Windows XP service pack 2, you may need to modify your firewall settings in order to send and receive DICOM files. See “[Troubleshooting a Retrieval Operation](#)” on page 40 for details.

To configure the Storage SCP Service parameters and an Application Entity, complete the following steps:

1. Open the **DICOM Network Services** utility in system configuration mode:

```
DICOMEX_NET, /SYSTEM
```

Note

The required **Storage SCP Application Entity** area is grayed out on the **Configuration** tab unless you start the **DICOM Network Services** utility using the SYSTEM keyword.

2. In the **Storage SCP Application Entity** area, configure Storage SCP settings for the machine on which the files will be stored.

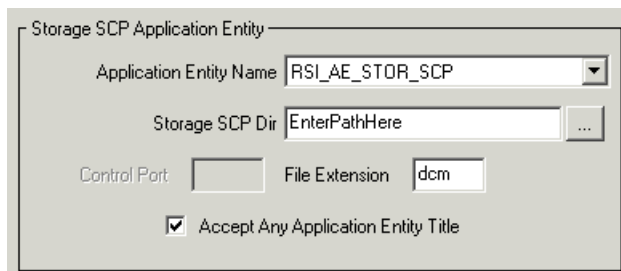


Figure 2-1: System Configuration of Storage SCP Application Entity

Note

You must define a valid directory for **Storage SCP Dir** in order for the Storage SCP service to function correctly.

Application Entity Name	Accept the default value, IDL_AE_STOR_SCP or see “ Defining a New Application Entity ” on page 26 to configure a new Application Entity.
Storage SCP Dir	Enter the full path to the directory location where the retrieved files will be located, or click the ... button to select or create the directory. Note - If multiple clients access a single installation of IDL, this file storage directory will be shared among all users.
Control Port (UNIX only)	Enter the port number the Storage SCP service listens at for control messages. Typically this value is one greater than the port number used by the Storage SCP Application Entity. The default for the control port value is 2511 (this is one greater than the default port number, 2510, for the default IDL_AE_STOR_SCP Application Entity). To change this value, you must: <ol style="list-style-type: none"> 1. Stop the Storage SCP service by clicking Stop Service. 2. Change the Control Port value. 3. Click Save to save the change. 4. Restart the Storage SCP Service by clicking Start Service. Warning - If you fail to follow these steps to change the control port value, you may not be able to stop the service from the DICOM Network Services utility because the user interface and the service will not be using the same control port number.

File Extension	Accept the default <code>dcm</code> extension or enter the extension that will be appended to files.
Accept Any Application Entity	<p>Check this box to allow the Storage SCP Service to accept files from any remote machine. Uncheck this box to accept files only from Application Entities defined in this dialog and stored in the system configuration file.</p> <p>When you uncheck this box, enter Application Entity information for each remote source machine that will be queried and will send files to the directory defined by Storage SCP Dir. This information must exist in the system configuration file. See “Configuring an Application Entity” on page 25 for information on configuring AEs.</p>

3. Click **Save** to save the Application Entity and Storage SCP directory information.
4. Stop and restart the Storage SCP Service. This step is required any time changes are made to the system configuration file. Click **Stop Service** and then click **Start Service**. You can then verify the service is running by clicking **Update Service Status**.

Note

Making changes in this dialog only affects characteristics of the local Storage SCP service Application Entity. It is not possible to change characteristics of remote SCP service nodes.

Make a note of the Storage SCP Application Entity information you have just configured. You will need to add this information on the device that will be queried for files. See [“Defining a Machine to Be Queried”](#) on page 30 for instructions.

Tip

You can use the `DICOMEX_GETSTORSCPDIR` function to return the location of the directory associated with the DICOM Store SCP service. See [“DICOMEX_GETSTORSCPDIR”](#) on page 60 for details.

Configuring an Application Entity

An Application Entity (AE) is a local or remote DICOM service. In an image archival system, common DICOM services include storage and query/retrieve. Devices (local or remote imaging equipment or computers) may support one or more services, and one or more service roles. Roles define a device as a service class user (SCU) or a service class provider (SCP). A SCU role is typically associated with a client action. A SCP role is typically associated with a server action.

Default Application Entities

The **DICOM Network Services** utility defines four default Application Entities:

- **IDL_AE_QUERY_SCU** — uses the Query SCU service, which lets you ask a device (a file source) about DICOM files contained in its database. This service works in conjunction with the Query SCP service, which allows you retrieve DICOM files from the remote node. On the file source device (the server in this instance), the Query SCP service listens for and responds to queries. It also listens for requests and sends the specified DICOM files.
- **IDL_AE_STOR_SCP** — uses the Storage SCP service, which listens at the defined TCP/IP port for incoming DICOM files and writes them to a local disk directory.
- **IDL_AE_STOR_SCU** — uses the Storage SCU service, which lets you send local DICOM files to a remote node.
- **IDL_AE_ECHO_SCU** — uses the Echo SCU service, which lets you determine if a remote SCP node is accessible.

The network characteristics of the default entities can be modified, but they cannot be deleted. You can also define new AEs. See the following section for details:

- [“Defining a New Application Entity”](#) on page 26
- [“Modifying an Application Entity”](#) on page 28
- [“Deleting an Application Entity”](#) on page 29

Replacing a Default Entity

You can use the default **IDL_AE_ECHO_SCU**, **IDL_AE_QUERY_SCU**, and **IDL_AE_STOR_SCU** Application Entities for echo, query and send operations without making any modifications. If you decide to define new entities to replace these, you will need to select the entity in the **Configuration** tab.

After defining a new entity as described in “[Defining a New Application Entity](#)” in the following section, complete the following steps:

1. Open the **DICOM Network Services** utility by entering:

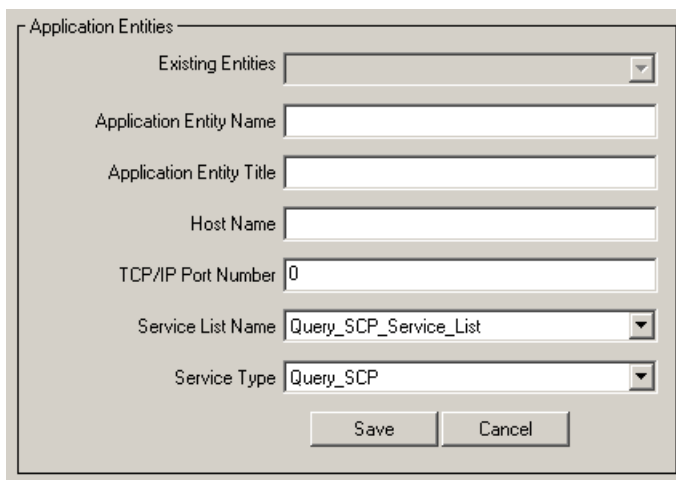
```
DICOMEX_NET
```

2. In the upper right quadrant of the **Configuration** tab, select the name of the new Application Entity in the **Application Entity Name** droplist to indicate the entity to be associated with the Echo SCU, Query Retrieve SCU or Storage SCU services.
3. Click **Save** to record the changes.

Defining a New Application Entity

To define a new Application Entity for a service, complete the following steps:

1. Select the **Configuration** tab of the **DICOM Network Services** utility that has been started in local or system configuration mode. See “[Starting the Network Services Utility](#)” on page 18 for details.
2. In the **Application Entities** area, click the **New** button to invoke Application Entity definition mode. This desensitizes other utility areas and clears fields within the **Application Entities** area, shown in the following figure.



The screenshot shows a dialog box titled "Application Entities". It contains several input fields and dropdown menus. The fields are: "Existing Entities" (a dropdown menu), "Application Entity Name" (a text box), "Application Entity Title" (a text box), "Host Name" (a text box), "TCP/IP Port Number" (a text box containing "0"), "Service List Name" (a dropdown menu containing "Query_SCP_Service_List"), and "Service Type" (a dropdown menu containing "Query_SCP"). At the bottom of the dialog are two buttons: "Save" and "Cancel".

Figure 2-2: Defining a New Application Entity

- Configure the properties of the Application Entity as follows:

Application Entity Name	A locally unique, descriptive identifier for the Application Entity consisting of a maximum of 30 characters.
Application Entity Title	A non-unique identifier for the Application Entity consisting of a maximum of 15 characters. An Application Entity with the same title may support more than one service (<i>e.g.</i> support Query SCP and Storage SCP).
Host Name	The machine host name or IP address consisting of a maximum of 30 characters. The term <code>localhost</code> is equivalent to the host name of the machine containing the IDL installation, typically the local machine. Note - See “Determining the IP Address” on page 16 for instructions on accessing a machine’s IP address.
TCP/IP Port Number	The valid port number of the local or remote SCP service consisting of a maximum of 5 digits. Set to 9999 for a SCU service type. The default ITT Visual Information Solutions-defined SCP service value is 2510. Note - See “Determining TPC/IP Port Numbers” on page 16 for information on returning port status.

Service List Name	Select the service list type from the droplist. Options are: <ul style="list-style-type: none"> • Query_SCP_Service_List • Query_SCU_Service_List • Storage_SCP_Service_List • Storage_SCU_Service_List • Echo_SCU_Service_List
Service Type	Select the service type from the droplist. This value should correspond to the service list type selected in the previous field. Options are: <ul style="list-style-type: none"> • Query_SCP • Query_SCU • Storage_SCP • Storage_SCU • Echo_SCU

4. Complete the Application Entity configuration. Click **Save** to save the Application Entity definition in the local or system configuration file. Click **Cancel** to dismiss the changes and exit Application Entity definition mode.
5. Stop and restart the Storage SCP Service any time you modify Application Entities that use the Storage_SCP service type while in system configuration mode (when you have started the **DICOM Network Services** utility with the SYSTEM keyword). See [“Managing the Storage SCP Service”](#) on page 51 for details.

Modifying an Application Entity

On the **Configuration** tab of the **DICOM Network Services** utility, you can modify the characteristics of default or custom application entities. For the default, Application Entities (those listed in [“Default Application Entities”](#) on page 25), you can modify the following fields:

- Application Entity Title
- Host Name
- TCP/IP Port Number

For a custom Application Entity, you can modify any configurable characteristic. Complete the following steps to modify an Application Entity definition:

1. Launch the **DICOM Network Services** utility in local or system configuration mode if needed. See “[Starting the Network Services Utility](#)” on page 18 for details.
2. Select the Application Entity to be modified from the **Existing Entities** droplist on the **Configuration** tab.
3. Click in any available field and make the desired changes. Other areas of the dialog are desensitized while in Application Entity definition mode.
4. Complete the Application Entity modification. Click **Save** to save the changes to the Application Entity definition in the local or system configuration file. Click **Cancel** to dismiss the changes and exit Application Entity definition mode.

Deleting an Application Entity

You can delete any custom Application Entity definitions that you have defined. You cannot delete the default Application Entities (those listed in “[Default Application Entities](#)” on page 25). To delete an Application Entity, complete the following steps:

1. Select the Application Entity to be deleted from the **Existing Entities** droplist.
2. Click **Delete**.
3. Confirm the deletion by clicking **Yes**.
4. Click **Save** to record the changes or **Cancel** to undo the changes.

Defining a Machine to Be Queried

Query/Retrieve is the process by which DICOM devices request information from a database and retrieve data and images through those requests. To form an association, the local Query Retrieve SCU connects with a remote Query Retrieve SCP. The client at the Query Retrieve SCU machine sends a request to the remote Query Retrieve SCP machine. The Query Retrieve SCP machine responds with a list of patients. From this list the client requests files (identified by patient ID) be sent to a destination (identified by Application Entity title). The remote Query Retrieve SCP machine responds to the request and sends files to the file storage directory that you defined in association with the Storage SCP Application Entity. The Storage SCP service writes the incoming files to disk.

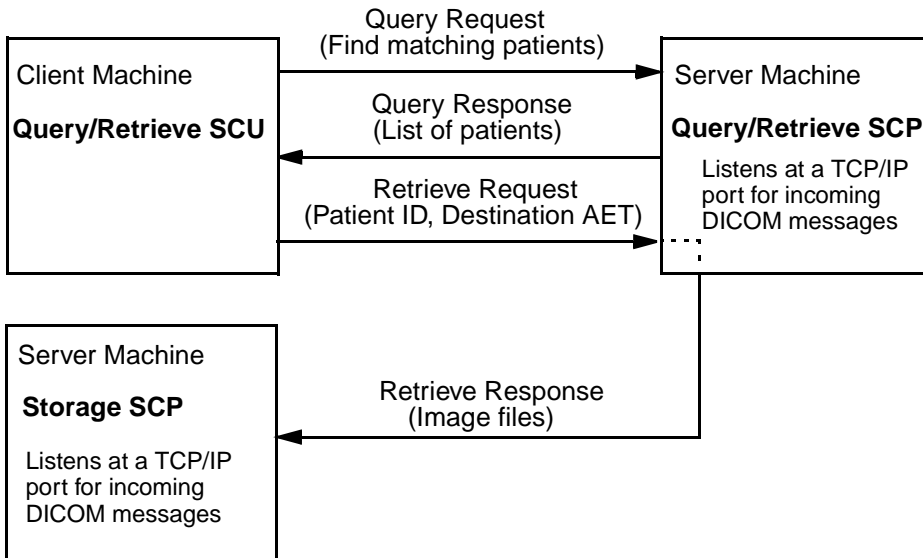


Figure 2-3: Flow of Data in a Query Retrieve Operation

Note

The default Query Retrieve SCU Application Entity is `IDL_AE_QUERY_SCU`. There is no need to make any changes to this entity definition, but you can define a new Application Entity if desired. You would then select the new entity in the **Query Retrieve SCU Application Entity** area of the **Configuration** tab in the **DICOM Network Services** utility and click **Save**.

Before a query/retrieve operation, you need to:

- Tell the Query Retrieve SCP device (the *file source*) where files should be sent (the Storage SCP Application Entity information that indicates the *file storage* location)
- Configure a Query SCP Application Entity (for the file source) on your machine (the machine running the query)

Complete the following steps:

1. On the file source (likely a remote machine or device) that is running the Query Retrieve SCP service, enter the Storage SCP Application Entity information. This Application Entity is the one defined in “[Configuring Your System to Receive Files](#)” on page 22. You can look up the following information in the **DICOM Network Services** utility by launching it with the SYSTEM keyword. (See “[Starting the DICOM Network Services Utility in System Mode](#)” on page 20 if needed).
 - **Application Entity Title** — the default is IDL_STORE_SCP.
 - **Host Name** — if the current value is *localhost*, you will instead need to provide the machine host name or IP address. See “[Determining the IP Address](#)” on page 16 if you do not know the host name.
 - **Port Number** — the default value is 2510.
2. Record the Application Entity title, host name or IP address, and port number of the Query Retrieve SCP service on the file source machine.
3. On your local machine, the machine from which the queries will be made, open the **DICOM Network Services** utility by entering:

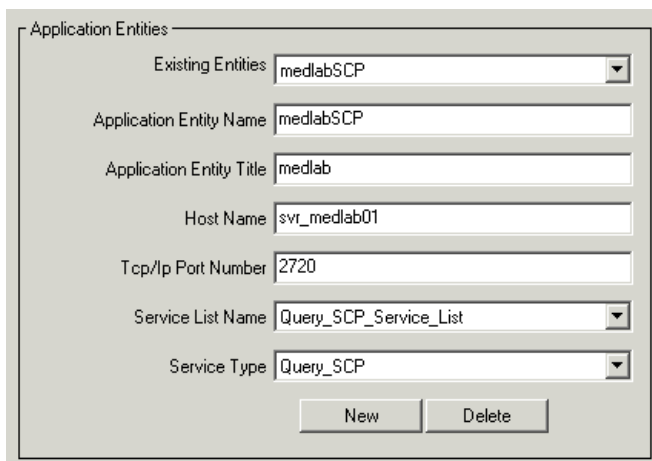

```
DICOMEX_NET
```
4. Define an Application Entity that describes the network characteristics of the device from which files are requested, the file source. Use the information collected in step 2 to configure a local Query SCP Application Entity. For instructions on defining a new Application Entity, see “[Defining a New Application Entity](#)” on page 26.

This entity should be assigned the following:

- **Service List Name** — Query_SCP_Service_List
- **Service Type** — Query_SCP

Selecting the Query_SCP service type for this Application Entity indicates the remote device listens for and responds to queries. This service also listens for

requests and sends the specified DICOM files. The following figure shows a hypothetical Application Entity configured as a Query SCP service for the file source device.



The screenshot shows a configuration window titled "Application Entities". It contains the following fields and values:

- Existing Entities: medlabSCP (dropdown)
- Application Entity Name: medlabSCP (text box)
- Application Entity Title: medlab (text box)
- Host Name: svr_medlab01 (text box)
- Tcp/Ip Port Number: 2720 (text box)
- Service List Name: Query_SCP_Service_List (dropdown)
- Service Type: Query_SCP (dropdown)

At the bottom of the window are two buttons: "New" and "Delete".

Figure 2-4: Application Entity for Remote Machine to be Queried

5. Also on your local machine, make sure the Storage SCP Application Entity referenced in step 1 is listed in the **Existing Entities** droplist on the **Configuration** tab. (This is the entity defined in [“Configuring Your System to Receive Files”](#) on page 22.) If you modified the default characteristics of the IDL_AE_STOR_SCP Application Entity or created a new Application Entity that uses the Storage SCP service while in system configuration mode, you must duplicate those settings in the **DICOM Network Services** utility while in local configuration mode.

See the following section, [“Querying a Remote Machine”](#) on page 33, for information on how to use the **DICOM Network Services** utility to query and retrieve files.

Querying a Remote Machine

The Query/Retrieve functionality of the **DICOM Network Services** utility is located on the **Query Retrieve SCU** tab. Once you have defined a Query SCP Application Entity for one or more file source devices (as described in “[Defining a Machine to Be Queried](#)” on page 30), you can query the remote machine. If you have also defined a Storage SCP service (defined in “[Configuring Your System to Receive Files](#)” on page 22), you can return requested files to a specified directory. This section covers topics related to configuring queries and retrieving files:

- “[Configure Query SCU Service Properties](#)” below
- “[Query the Remote Node](#)” on page 34
- “[Build or Modify a Custom Query](#)” on page 36
- “[Retrieve Files from a Remote Node](#)” on page 38

Note

The **Query Retrieve SCU** tab is not available if you start the **DICOM Network Services** utility with the **SYSTEM** keyword.

Configure Query SCU Service Properties

Before querying a remote machine, select an Application Entity that supports the Query SCU service type and define the number of responses to be returned.

1. Open the **DICOM Network Services** utility by entering the following at the IDL command prompt:

```
DICOMEX_NET
```

2. On the **Configuration** tab, modify settings in the **Query Retrieve SCU Application Entity** area:
 - **Application Entity Name** — select an Application Entity from the droplist. The default is `IDL_AE_QUERY_SCU`. If you have defined other Application Entities that use the Service Type of Query SCU, they will appear here.
 - **Max Query Responses** — enter the maximum number matches to return for a query. The default value is 100.

Query the Remote Node

To query a remote machine, complete the following steps:

1. Open the **DICOM Network Services** utility if needed by entering:

```
DICOMEX_NET
```

2. Click on the **Query Retrieve SCU** tab.
3. In the **Query Node** droplist, select the Application Entity name associated with the remote machine to be queried. If this droplist is blank, you need to define an Application Entity for the remote machine as described in [“Defining a Machine to Be Queried”](#) on page 30.
4. Select the query to be performed from the droplist:
 - **All patients** — the default patient-level query
 - **Use current query** — the last configured custom query. If you have not built a custom query, the default is a patient-level query.

Note

See [“Build or Modify a Custom Query”](#) on page 36 if you want to change the default query.

5. Click the **Query** button. All patient files located on the remote machine (up to the number of **Max Query Responses** defined on the **Configuration** tab) that match the query parameters will be returned.
6. View the query results in the **Results** section, described in [“Query Results”](#) on page 35. Status information appears in the **Status** window.

Note

If you have a query error, you can use the Echo SCU functionality to test the connection. See [“Returning Connection Status with Echo”](#) on page 48.

Cancelling a Query Operation

To stop a query, click the **Cancel** button. This sends a cancel request to the remote machine and halts the return of file information.

Query Results

The results of any query are shown in the **Results** section of the **Query Retrieve SCU** tab. This area is divided into a navigation tree in the left pane and a table display in the right pane. A subset of the **Results** area is shown in the following figure.

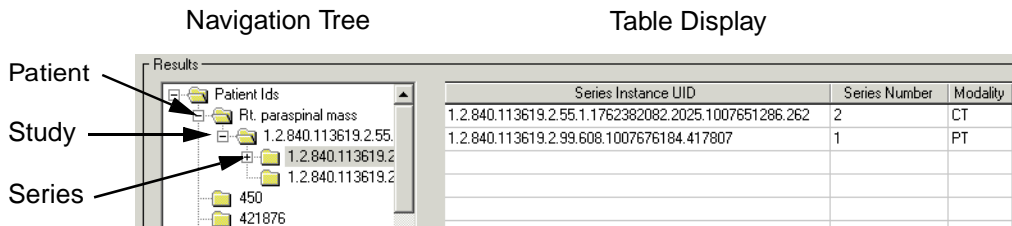


Figure 2-5: Results Area of the Query Retrieve SCU Tab

The Navigation Tree area displays the patient-study-series-image hierarchy. The Table Display area displays selected patient, study, series or image information.

When you click on an item in the Navigation Tree, a sub-query returns child items. For example, clicking on a series will perform a sub-query to return all images associated with that series, up to the **Maximum Query Responses** value. Click the + symbol next to any previously selected item (or double-click the item) to see sub-items.

Note

For optimum performance, the initial query does not populate the Navigation Tree with all child items. Because these items are returned on request, there may be a slight delay in the display of sub-query results.

Note

Some remote machines do not return a patient ID with query results. When you perform a query by clicking the **Query** button, all results returned without a patient ID appear under “Unknown Patient ID” in the Navigation Tree window.

Build or Modify a Custom Query

You can create a custom query to return patient or study files that match specific patient-, study-, or series-level characteristics. To create a custom query, complete the following steps:

1. Click the **Build Query** button on the **Query Retrieve SCU** tab to open the **Query Fields** dialog.
2. In the **Query Model** area, select **Patient Root** or **Study Root**. This indicates the highest level of information that can be retrieved. For instance, if you select **Study Root**, you cannot retrieve the study by selecting the associated patient. You must select a study, series or image to be retrieved.

Note

To check what query models are supported by the remote node, use the **Echo SCU** service on the **Configuration** tab. Select the Query SCP entity in the droplist and click **Echo**. If the echo is successful, query model information is printed following echo results in the **Status** window.

3. In the **Query Level** area, select **Patient Level**, **Study Level** or **Series Level** to define patient, study or series characteristics against which files are matched.

Note

Changing the query model or query level makes some query fields active and others insensitive. Any values shown in insensitive fields are not applied to the current query.

4. Click **Apply** to store the query or **Ok** to store the query and close the dialog. The specified query fields are applied to subsequent queries. If you select **Clear**, the query will be performed at the **Query Model** level. Select **Cancel** to clear changes and revert to the last used query. Click the **Help** button for information on using wildcards in attribute matching, and information on DICOM date formats.

Attribute Matching Using Wildcards

The implementation of the Query/Retrieve SCP service on the remote node determines the quality of attribute matching. All attribute matching is performed by the remote service. In addition to case sensitivity, matching can include one or more of the following:

Single Value Matching	Matches single values.
Wildcard Matching	Use * to match zero or more characters. Use ? to match any single character. For example, to return patients with a name beginning with HA, use HA*. To return a list of patients whose names vary by one or more instances of a single value, use the ? character to indicate the wildcard value as in M?NROE.
Range Matching	Use the - character in between date or time values to return any matches within that range. For example, to match any date in the first six months of the year, use 20050101-20050630.

The following table indicates the value representation (VR) and what wildcards are supported in each query field. A "•" in the column indicates the wildcard is supported.

Field	VR	*	?	-
Patient Name	PN	•	•	
Family Name	PN	•	•	
Given Name	PN	•	•	
Middle Name	PN	•	•	
Prefix	PN	•	•	
Suffix	PN	•	•	
Patient ID	LO	•	•	
Study Date	DA			•

Table 2-1: Wildcard Support for Query Fields

Field	VR	*	?	-
Study Time	TM			•
Accession Number	SH	•	•	
Study ID	SH	•	•	
Modality	CS			

Table 2-1: Wildcard Support for Query Fields

Retrieve Files from a Remote Node

The **DICOM Network Services** utility lets you retrieve selected DICOM data from a remote machine, and store the files in the directory you associated with the Storage SCP service. To retrieve data, complete the following steps:

1. Click on the **Query Retrieve SCU** tab of the **DICOM Network Services** utility if it is not already selected.
2. Perform a query against a remote database. See [“Query the Remote Node”](#) on page 34 for instructions.
3. In the **Destination Node** droplist, select the Application Entity name associated with the current Storage SCP service. If there is more than one Application Entity that supports Storage SCP, you can locate the name of the current entity on the **Configuration** tab in the grayed out **Storage SCP Application Entity** area.
4. In the Navigation Tree portion of the **Results** area, select the patient, study, series or image that you want to retrieve.

Note

If you have selected **Study Root** in the **Query Model** area of the **Query Fields** dialog (described in [“Build or Modify a Custom Query”](#) on page 36), you cannot retrieve data at a patient level. You will need to select a study, series or image in the Navigation Tree.

5. Click **Retrieve** to return the selected data. This retrieves all images associated with a selected patient, study or series, or retrieves the selected image. Files are stored in the directory you associated with the Storage SCP Application Entity. The directory location is shown in the grayed out **Storage SCP Application Entity** area on the **Configuration** tab. Retrieve status information is available in the **Status** window.

Note

There is a `Depot` subdirectory located in the directory you specified when configuring the Storage SCP Application Entity. This directory is used in the process of file retrieval and can be ignored. See [“About the Depot Directory”](#) on page 50 for details.

Warning

Existing files with duplicate names are overwritten. See [“About the Storage SCP Service”](#) on page 49 for details.

Canceling a Retrieval Operation

To halt the retrieval of data, click the **Cancel** button. This sends a cancel request to the remote node and halts file transfers.

Troubleshooting a Retrieval Operation

There are common configuration errors that can lead to retrieval problems. If one of the following errors appear in the **Status** window of the **Query Retrieve SCU** tab, consider the following possible resolutions:

Error	Resolution
Move destination unknown.	<p>The remote machine does not recognize the retrieve Destination Node. Try these steps:</p> <ol style="list-style-type: none"> 1. Make sure that your current Storage SCP Application Entity is selected in this droplist. This is described in “Query the Remote Node” on page 34. 2. Echo the Storage SCP Application Entity to verify valid network connection settings. See “Returning Connection Status with Echo” on page 48. 3. Try to send a file to the Storage SCP directory by selecting a local directory containing any DICOM file and sending it to yourself using the Storage SCU functionality. See “Sending Files to a Remote Destination” on page 43 for details.

Table 2-2: Troubleshooting Retrieval Errors

Error	Resolution
Unable to process error.	<p>If all of the resolutions in the previous steps work correctly, the problem is likely with the configuration of the your Storage SCP Application Entity information on the remote machine. Check these settings on the machine you are attempting to query. See “Defining a Machine to Be Queried” on page 30 for more information.</p> <p>If you are running Windows XP service pack 2, the firewall that is automatically started is likely blocking incoming DICOM packets. See “Allowing File Transfer with a Windows XP Firewall” on page 41 for ways to modify your firewall settings.</p>
Files do not appear in the specified directory.	<p>Restart the Storage SCP service. After making any changes to the configuration properties of a Storage SCP Application Entity while in system configuration mode (when the DICOM Network Services utility is started with <code>DICOMEX_NET, /SYSTEM</code>), you must stop and restart the Storage SCP Service. Use the Stop Service and the Start Service buttons on the Configuration tab. See “Managing the Storage SCP Service” on page 51 for details.</p>

Table 2-2: Troubleshooting Retrieval Errors

Note

If you are unable to retrieve files, verify that virus scanning software is not blocking the transfer. If file transfers are extremely slow, you may want to disable any virus scanning software when performing query, retrieve or send operations.

Allowing File Transfer with a Windows XP Firewall

By default, Windows XP service pack 2 automatically starts a firewall. This blocks DIOCM file packet transfer until you either modify your firewall settings or disable the firewall.

To modify your firewall settings to include the port number associated with your IDL_AE_STOR_SCP Application Entity (as defined in [“Configuring Your System to Receive Files”](#) on page 22), complete the following steps.

1. Select **Start** → **Control Panel** → **Windows Firewall**.
2. Click on the **Exceptions** tab and add the port number of entity associated with your Store SCP application entity. Select **Add Port** and enter a name identifying the port and the port number. If you accepted the default settings for the IDL_AE_STOR_SCP Application Entity, the port number is 2510.
3. Click **OK** to save the changes and **OK** to exit the dialog.

You also have the option to completely disable the firewall, although this should be done only if there is low risk of infection from viruses or other external attacks. To disable your firewall, select **Start** → **Control Panel** → **Windows Firewall** and choose **Off**. Click **OK** to save the changes and exit the dialog.

Sending Files to a Remote Destination

The Storage SCU functionality of the **DICOM Network Services** utility transmits DICOM files from the local DICOM Storage SCU (Service Class User) to a remote destination that is identified as a DICOM Storage SCP (Service Class Provider). This sends a copy of the images to the file storage machine, leaving the original image data intact.

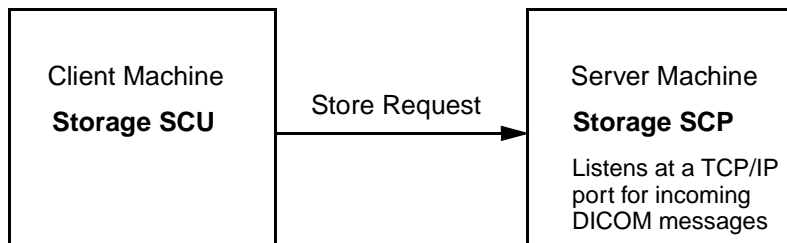


Figure 2-6: Flow of Data in a Storage Operation

The IDL Storage SCU functionality typically sends a DICOM file using the file's current transfer syntax. IDL detects, during the association negotiation, whether or not the remote Storage SCP server can accept the file's current transfer syntax. If the remote Storage SCP cannot accept the current transfer syntax the file's transfer syntax is changed to Implicit VR or Explicit VR and then sent to the remote node. The original file on disk remains unchanged. JPEG Lossy files are not converted to Implicit VR or Explicit VR files because the user at the remote end could be misled into thinking that the file contains lossless data.

Note

The required **Storage SCU** tab is not available if you start the **DICOM Network Services** utility with the **SYSTEM** keyword.

To send files to a remote machine, see the following topics:

- [“Defining the Remote Storage SCP Node”](#) below — describes how to configure an Application Entity for the machine to which you will send files
- [“Sending Files to a Remote Machine”](#) on page 44 — describes using the **DICOM Network Services** utility to browse for, select and transmit files to the remote machine

Defining the Remote Storage SCP Node

You need to configure an Application Entity for the remote machine to which you will send DICOM files. (Once an Application Entity has been defined, you do not need to re-execute the following steps.) Record the Application Entity name, host name or IP address, and port number of the remote machine's Storage SCP service. You need this information to configure the network properties of the Application Entity as follows:

1. Open the **DICOM Network Services** utility by entering the following at the IDL command prompt:

```
DICOMEX_NET
```

2. Click on the **Configuration** tab and create a new Application Entity. Enter the network parameters for the remote machine. Additionally, this Application Entity must support:
 - **Service List Name** — Storage_SCP_Service_List
 - **Service Type** — Storage_SCP

See [“Defining a New Application Entity”](#) on page 26 for further instructions.

Note

You can verify a connection to the remote machine using the Echo SCU functionality. See [“Returning Connection Status with Echo”](#) on page 48.

Once the Application Entity has been defined, you can send files to the remote node, described in the following section.

Sending Files to a Remote Machine

You can use the Storage SCU functionality of the **DICOM Network Services** utility to browse for and send DICOM files to a remote machine. To do so, complete the following steps:

1. Open the **DICOM Network Services** utility by entering the following at the IDL command prompt if needed:

```
DICOMEX_NET
```

2. Click on the **Storage SCU** tab.

Note

This tab is not available if the **DICOM Network Services** utility is started with the **SYSTEM** keyword.

3. Select the Application Entity associated with the machine to which you want to send files in the **Destination Node** droplist. This was configured as described in “[Defining the Remote Storage SCP Node](#)” on page 44.
4. Select the files to send. You can either send files associated with the DICOM patient hierarchy (patient, study, series, image), or directly select files to send as follows:
 - **Send data related to the DICOM patient hierarchy.** In the **Send Patient Data** area, click the **Select Directory** button and select a directory containing DICOM files. Click in the Tree Navigation area and select a patient, study, or series to send all related images, or select a single image. Details about the selected item is displayed to the right of the Tree Navigation area. Click the **Send Patient Files** button to send the image(s) to the remote machine. Operation status is displayed in the **Status** area. The following figure shows an image selected in the Tree Navigation area.

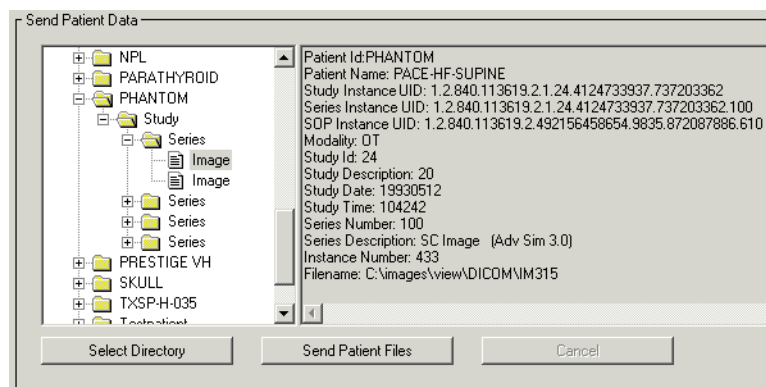


Figure 2-7: Selecting an Image to Send

- **Send a selected file.** In the **Send File(s)** area, click the **Browse/Send Files** button. Locate the file or files you want to send. Select multiple files by **Ctrl-** or **Shift-**clicking. Click **Open** to send the file(s). Operation status is reported in the **Status** area, an excerpt of which is shown here.

```
Status
-----
Send Started. Pushing Dicom files to SHOWCASE_SCP
File Sent: C:\images\view\1.2.840.113619.2.25.1.1762306543.872804962.239.dcm
File Sent: C:\images\view\1.2.840.113704.3.1.111611.115401127.202720.56.dcm
File Sent: C:\images\view\1.2.840.113704.3.1.111611.115401127.202720.56.dcm
Send Completed: 3 files successfully sent
```

Figure 2-8: Status Report of a Send Operation

Troubleshooting a Send Operation

There are common configuration errors that can lead to problems when attempting to send files. If the following error appears in the **Status** window of the **Storage SCU** tab, consider the following possible resolutions:

Error	Resolution
Failed to open an association.	<p>The remote machine you are attempting to send images to is not recognized. Try these steps:</p> <ol style="list-style-type: none"> 1. Make sure that you can connect to the remote machine. See “Returning Connection Status with Echo” on page 48. If echo fails, see the following item. 2. Try sending files to yourself. Select the current Storage SCP entity (which appears in the grayed out Storage SCP Application Entity section of the Configuration tab) from the Destination Node droplist. (This was configured in “Configuring Your System to Receive Files” on page 22). Select and send yourself a file. 3. If you can send yourself a file, double-check the network settings of the Application Entity related to the machine you are attempting to send files to. The Application Entity Title, Host Name (or IP Address), and Port Number must all match the values of the remote machine. See “Defining the Remote Storage SCP Node” on page 44.

Table 2-3: Troubleshooting Send Operation Errors

Note

If your files are not arriving at the destination and you are running Windows XP service pack 2, the firewall that is automatically started is likely blocking the transfer of DICOM packets. See [“Allowing File Transfer with a Windows XP Firewall”](#) on page 41 for ways to modify your firewall settings.

Returning Connection Status with Echo

The Echo SCU functionality of the **DICOM Network Services** utility lets you test the network connection to a remote machine that supports Query SCP or Storage SCP service types. To test the connection status of a remote node, do the following:

1. If needed, start the **DICOM Network Services** utility. See “[Starting the Network Services Utility](#)” on page 18 for details.
2. Select the **Configuration** tab.
3. In the **Echo SCU** area, select the Application Entity name from the **Remote Nodes** droplist. The results of the echo operation appear in the **Status** window.

Troubleshooting Echo Errors

If the following error appears in the **Status** window of the **Configuration** tab, consider the following possible resolutions:

Error	Resolution
Echo Failed: Open Association Failed	<p>The machine you are attempting to connect to is not available. Try these steps:</p> <ol style="list-style-type: none"> 1. Double-check the network settings of the Application Entity related to the machine you are attempting to connect to. The Application Entity Title, Host Name (or IP Address), and Port Number must all match the values of the remote machine. 2. Restart the Storage SCP service. After making any changes to the configuration properties of a Storage SCP Application Entity while in system configuration mode (when the DICOM Network Services utility is started with <code>DICOMEX_NET, /SYSTEM</code>), you must stop and restart the Storage SCP Service. Use the Stop Service and the Start Service buttons on the Configuration tab to restart the service.

Table 2-4: Troubleshooting Echo Operation Errors

About the Storage SCP Service

The Storage SCP Service runs as a service on Windows and as a daemon on UNIX. This service listens at a specified TCP/IP port for incoming DICOM files and writes them to the directory that is specified in [“Configuring Your System to Receive Files”](#) on page 22. Incoming files are named according to the file’s SOP Instance UID value and appended with the specified extension.

Warning

When a file with a duplicate name is retrieved, the original is overwritten. As the file name is based on the SOP Instance UID, this should only occur when retrieving the same file more than once.

For complete details regarding how to configure the characteristics of the Application Entity associated with the Storage SCP service, see [“Configuring Your System to Receive Files”](#) on page 22.

Storage SCP Service Permissions

As described in [“Configuring Your System to Receive Files”](#) on page 22, you must define a Storage SCP directory to which files are written. The Storage SCP Service requires permissions to:

- Create, read, write and delete files in the Storage SCP directory
- Create a `depot` subdirectory in the Storage SCP directory
- Create, read, write and delete files in the `depot` directory
- Create and write files in the `bin/bin.platform/dicomex` directory of the IDL installation directory where `platform` is the platform-specific `bin` directory.

When a user installs IDL with administrator or root privileges, the required permissions are set on the `dicomex` directory to allow access to all users. All users can change Storage SCP service configuration parameters. However, on Windows, only users with administrator privileges have the ability to start and stop the Storage SCP service, which is required for changes to take effect.

Note

The Storage SCP service issues a “Failed. Administrator privileges may be needed.” error if a user without administrator privileges tries to start or stop the Storage SCP service.

About the Depot Directory

The `depot` directory is used to ensure that the full DICOM file is successfully transferred from a remote machine before it is made available in the defined Storage SCP directory. The `depot` directory is created when you first retrieve files from a remote machine.

Storage SCP Service Log Files

Service events generated by the Storage SCP Service are recorded in two log files located in the `bin/bin.platform/dicomex` directory of the IDL installation directory where `platform` is the platform-specific `bin` directory. These files are called `storscp1.log` and `storscp2.log`. Events are logged to one file until the size limit (250 KB) is reached. The other file is then erased (if needed) and subsequent events are logged there.

These log files include information on four types of events:

- A start event when the Storage SCP Service is started
- An association event when an association is negotiated
- A write event when a file is written to the specified storage directory
- An error event when a service error occurs

Note

These log files do not contain information from the **DICOM Network Services** utility. No utility errors or other types of information appear in the Storage SCP Service log files.

Managing the Storage SCP Service

You can start and stop the Storage SCP service from within the **DICOM Network Services** utility. You will typically use the **Storage SCP Service Manager** after making changes to the Storage SCP Service configuration, or before performing a retrieve operation if you did not select to have the service automatically started at boot time.

Manage the service status from the **Storage SCP Service Manager** area of the **Configuration** tab of the **DICOM Network Services** utility. From this area, you can do the following:

- **Start Service** — click this button to start the service.
- **Stop Service** — click this button to stop the service.
- **Update Service Status** — click this button to verify the current status of the service. Inquiry results are printed in the **Status** area.

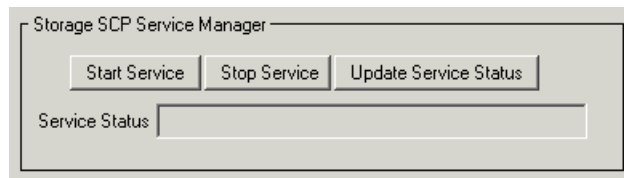


Figure 2-9: Storage SCP Service Manager

Note

To avoid potential problems, only start a single Storage SCP Service per machine.

Note

The current Storage SCP Service recognizes and uses the most recently installed version of IDL. Attempting to access the Storage SCP Service with a previous version of IDL will be problematic and is not recommended.

Starting and Stopping the Service Outside the Utility

On Windows, use the Windows Services Administrative tool to start or stop the IDL DicomEx Storage SCP service. To access the Service dialog, select **Start** → **[Settings]** → **Control Panel** → **Administrative Tools**, and double-click **Services** in the Administrative Tools list.

On UNIX and Mac OS X, start the service from the command prompt by using the startup script installed in the `bin/bin.platform` directory of the IDL installation directory where *platform* is the platform-specific bin directory. To stop the service, use the `kill` command with the process id of the service (`idl_dicom`). Use one of the following to return the process id:

- `ps -aef | grep idl_dicom` on Linux
- `ps -axu | grep idl_dicom` on Macintosh OS X
- `ps -aef | grep idl_dicom` on Solaris

DICOM Network Services User Interface

The user interface of the **DICOM Network Services** utility varies depending on whether it is started in system or local mode. See “[System Mode Interface](#)” and “[Local Mode Interface](#)” below for user interface details.

System Mode Interface

When you start the **DICOM Network Services** utility in system mode, you have access to the following **Configuration** tab:

The screenshot shows the 'Configuration' tab of the DICOM Network Services utility. The window is divided into several sections:

- Application Entities:** This section contains a list of existing entities (RSI_AE_QUERY_SCU) and a form to add a new one. Fields include Application Entity Name (RSI_AE_QUERY_SCU), Application Entity Title (RSI_QUERY_SCU), Host Name (localhost), TCP/IP Port Number (9999), Service List Name (Query_SCU_Service_List), and Service Type (Query_SCU). There are 'New' and 'Delete' buttons.
- Echo SCU:** This section has a 'Remote Nodes' dropdown (RSI_AE_STOR_SCP) and a 'Status' area with a scrollable list and an 'Echo' button.
- Echo SCU Application Entity:** A form with 'Application Entity Name' (RSI_AE_ECHO_SCU).
- Query Retrieve SCU Application Entity:** A form with 'Application Entity Name' (RSI_AE_QUERY_SCU) and 'Max Query Responses' (100).
- Storage SCU Application Entity:** A form with 'Application Entity Name' (RSI_AE_STOR_SCU).
- Storage SCP Application Entity:** A form with 'Application Entity Name' (RSI_AE_STOR_SCP), 'Storage SCP Dir' (EnterPathHere), 'Control Port' (empty), 'File Extension' (dcm), and a checked checkbox for 'Accept Any Application Entity Title'.
- Storage SCP Service Manager:** This section includes 'Start Service', 'Stop Service', and 'Update Service Status' buttons, along with a 'Service Status' field.

At the bottom of the window are 'Save', 'Cancel', 'Help', and 'Close' buttons.

Figure 2-10: Configuration Tab in System Mode

Local Mode Interface

When you start the **DICOM Network Services** utility in local mode, you have access to the following utility elements:

- “[Configuration Tab User Interface](#)”
- “[Query Retrieve Tab User Interface](#)” on page 55
- “[Storage SCU Tab User Interface](#)” on page 56

Configuration Tab User Interface

The **Configuration** tab is available in local and system modes, but the **Storage SCP Application Entity** area is not editable in local mode.

Figure 2-11: Configuration Tab in Local Mode

Query Retrieve Tab User Interface

The **Query Retrieve SCU** tab is available only in local mode.

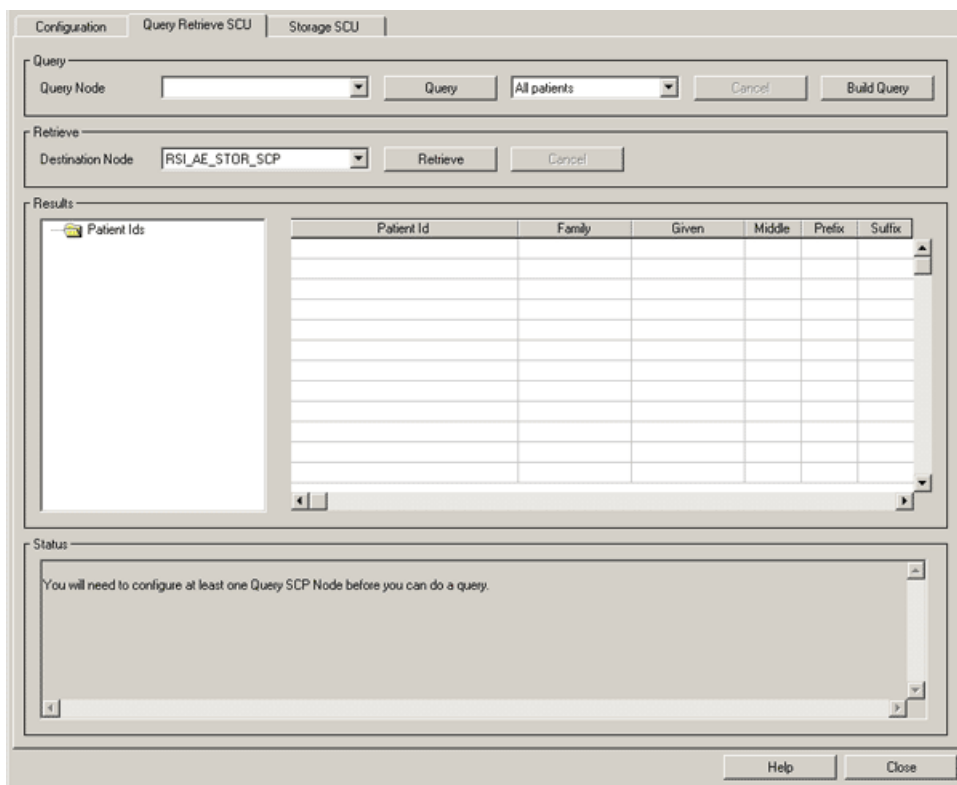


Figure 2-12: Query Retrieve SCU Tab in Local Mode

Storage SCU Tab User Interface

The **Storage SCU** tab is available only in local mode.

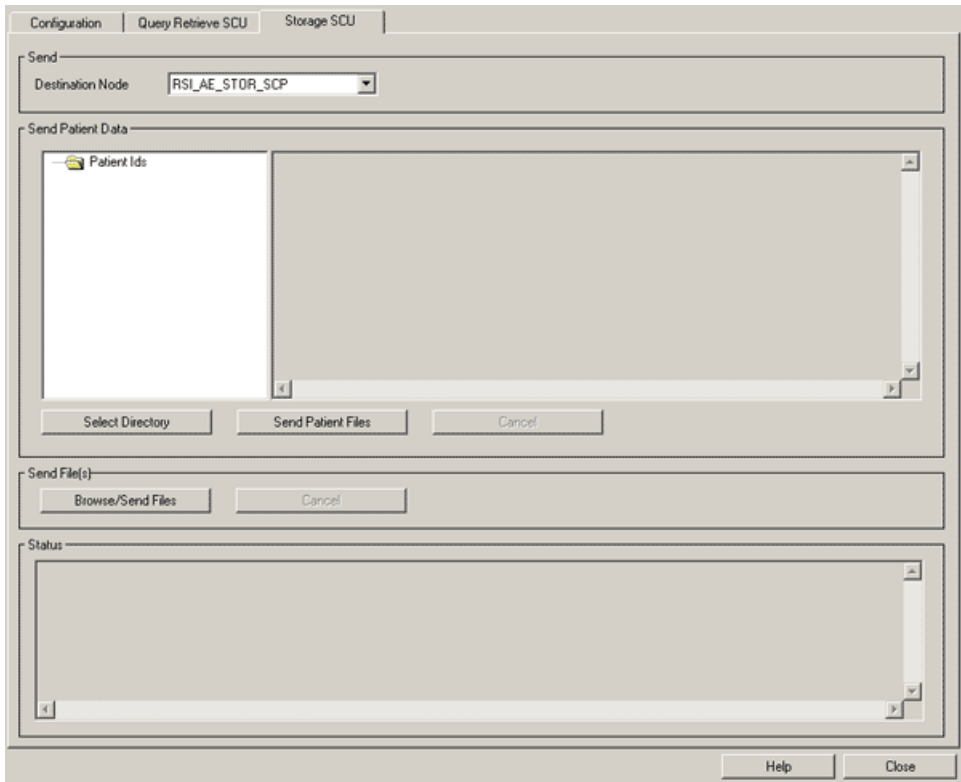


Figure 2-13: Storage SCU Tab in Local Mode

Chapter 3

IDL DICOM Reference

This chapter provides reference information for DICOM Network Services and the IDLffDicomEx object.

DICOMEX_GETCONFIGFILEPATH	58	IDLffDicomExCfg	212
DICOMEX_GETSTORSCPDIR	60	IDLffDicomExQuery	239
DICOMEX_NET	62	IDLffDicomExStorScu	283
IDLffDicomEx	64		

DICOMEX_GETCONFIGFILEPATH

The DICOMEX_GETCONFIGFILEPATH function returns the location of the local or system configuration file associated with Application Entities defined in the **DICOM Network Services** utility. See “[Local Versus System Configuration](#)” on page 18 for information on this distinction. See [Chapter 2, “Using IDL DICOM Network Services”](#) for information on using the **DICOM Network Services** utility.

Note

This function will fail if you have not installed and licensed IDL’s DICOM Network Services, which provides access to the **DICOM Network Services** utility. Use the [DICOMEX_NET](#) routine to start this utility.

Syntax

```
Result = DICOMEX_GETCONFIGFILEPATH([, /SYSTEM])
```

Return Value

Returns the location of the local or system configuration file, as specified using the SYSTEM keyword.

Keywords

SYSTEM

Set this keyword to return the full path of the system configuration file. The path of the local configuration file is returned if this keyword is not set.

Examples

Return the path to the local configuration file:

```
LocalFile = DICOMEX_GETCONFIGFILEPATH()
```

Return the path to the system configuration file:

```
SystemFile = DICOMEX_GETCONFIGFILEPATH(/SYSTEM)
```

Version History

6.2	Introduced
-----	------------

DICOMEX_GETSTORSCPDIR

Use the DICOMEX_GETSTORSCPDIR function to return the location of the directory associated with the Storage SCP Service. The location of this directory is configured using the **DICOM Network Services** utility as described in “[Configuring Your System to Receive Files](#)” on page 22. When performing a query/retrieve operation, this directory will contain the files returned by a request. Use this function to return the full path that was configured for that directory.

Note

This function will fail if you have not installed and licensed IDL’s DICOM Network Services, which provides access to the **DICOM Network Services** utility. Use the [DICOMEX_NET](#) routine to start this utility.

Syntax

Result = DICOMEX_GETSTORSCPDIR()

Return Value

Returns the location of the directory associated with the Storage SCP Service.

Keywords

None.

Examples

Return the path of the directory associated with the DICOM Store SCP service and allow the user to select a file from the directory using DIALOG_PICKFILE:

```
fileDir = DICOMEX_GETSTORSCPDIR()

; Allow the user to select a DICOM file.
sFile = DIALOG_PICKFILE(PATH=fileDir, $
    TITLE='Select DICOM Patient File', FILTER='*.dcm')
```

Version History

6.2	Introduced
-----	------------

DICOMEX_NET

Use the DICOMEX_NET procedure to launch the **DICOM Network Services** utility, which supports the following DICOM network services:

- Echo SCU
- Query/Retrieve SCU
- Storage SCU
- Storage SCP

See [“Overview of DICOM Network Services”](#) on page 14 for an introduction. See [“Using IDL DICOM Network Services”](#) on page 13 for complete details on how to use the **DICOM Network Services** utility.

Note

This feature requires an additional-cost license key to access the functionality. For more information, contact your ITT Visual Information Solutions sales representative or technical support.

Syntax

```
DICOMEX_NET [, /SYSTEM]
```

Keywords

SYSTEM

Set this keyword to start the **DICOM Network Services** utility in system mode. This mode allows you to define the directory and Application Entity associated with the Storage SCP Service.

If you do not set this keyword, the **DICOM Network Services** utility starts in local mode. In this mode you can configure Application Entities, and access Query Retrieve and Storage SCU functionality.

See [“Starting the Network Services Utility”](#) on page 18 for details.

Examples

Start the **DICOM Network Services** utility in local mode:

```
DICOMEX_NET
```

Start the **DICOM Network Services** utility in system mode:

```
DICOMEX_NET, /SYSTEM
```

Version History

6.2	Introduced
-----	------------

IDLffDicomEx

[Superclasses](#) | [Properties](#) | [Methods](#) | [Version History](#)

The IDLffDicomEx object allows you to access, read from, and write to DICOM files. Depending on how you initialize the IDLffDicomEx object, you can create a new DICOM file, clone and modify an existing DICOM file, or access elements from a file in read-only mode. See “[IDLffDicomEx::Init](#)” on page 163 for details.

Note

See “[IDLffDicomEx Overview](#)” on page 67 for specific details on the IDLffDicomEx object’s allowable transfer syntaxes, as well as introductory information on the structure of DICOM attributes. For conformance information, see www.ittvis.com/idl/dicom.

Note

This feature requires an additional-cost license key to access the functionality. For more information, contact your ITT Visual Information Solutions sales representative or technical support.

Superclasses

None

Creation

See “[IDLffDicomEx::Init](#)” on page 163.

Properties

The IDLffDicomEx object has the following properties.

- [BITS_ALLOCATED](#)
- [BITS_STORED](#)
- [COLUMNS](#)
- [FILENAME](#)
- [HIGH_BIT](#)
- [IMAGE_TYPE](#)
- [INSTANCE_NUMBER](#)
- [MODALITY](#)
- [NO_PIXEL_DATA](#)
- [NUMBER_OF_FRAMES](#)
- [PHOTOMETRIC_INTERPRETATION](#)
- [PIXEL_ASPECT_RATIO](#)

- `PIXEL_MAX`
- `PIXEL_REPRESENTATION`
- `PLANAR_CONFIGURATION`
- `SAMPLES_PER_PIXEL`
- `SOP_INSTANCE_UID`
- `PIXEL_MIN`
- `PIXEL_SPACING`
- `ROWS`
- `SOP_CLASS_UID`
- `TRANSFER_SYNTAX`

See “IDLffDicomEx Properties” on page 70 for details on individual properties.

Methods

This class has the following methods:

- `IDLffDicomEx::AddGroup`
- `IDLffDicomEx::AddPrivateGroup`
- `IDLffDicomEx::AddPrivateSequence`
- `IDLffDicomEx::AddSequence`
- `IDLffDicomEx::ChangeTransferSyntax`
- `IDLffDicomEx::Cleanup`
- `IDLffDicomEx::Commit`
- `IDLffDicomEx::CopyTags`
- `IDLffDicomEx::EnumerateTags`
- `IDLffDicomEx::GetDescription`
- `IDLffDicomEx::GetPixelData`
- `IDLffDicomEx::GetPrivateValue`
- `IDLffDicomEx::GetPrivateValueCount`
- `IDLffDicomEx::GetPrivateValueLength`
- `IDLffDicomEx::GetPrivateVR`
- `IDLffDicomEx::GetProperty`
- `IDLffDicomEx::GetValue`
- `IDLffDicomEx::GetValueCount`
- `IDLffDicomEx::GetValueLength`

- [IDLffDicomEx::GetVR](#)
- [IDLffDicomEx::Init](#)
- [IDLffDicomEx::QueryPrivateValue](#)
- [IDLffDicomEx::QueryValue](#)
- [IDLffDicomEx::SetPixelData](#)
- [IDLffDicomEx::SetPrivateValue](#)
- [IDLffDicomEx::SetProperty](#)
- [IDLffDicomEx::SetValue](#)

In addition, this class inherits the methods of its superclasses (if any).

Version History

6.1	Introduced
6.2	Added NO_PIXEL_DATA property

IDLffDicomEx Overview

The IDLffDicomEx object provides methods for reading and writing pixel data to a DICOM file, and for specifying the data compression. Other methods let you add, modify, or remove public and private DICOM attribute tags, public and private sequences, and sets of repeating tags within sequences. See “[DICOM Sequence Items](#)” below for details. Other methods allows you to output all tags in a DICOM file to an ASCII file or to an IDL structure. You can also copy subsets of tags from one file to another.

A DICOM file contains DICOM attributes or data elements, which are composed of:

- A *tag*, in the format of *group* number, *element* number (XXXX,XXXX) that identifies the attribute
- A *Value Representation* (VR) that describes the data type and format of the attribute’s value
- A *value length* that defines the length of the attribute’s value
- A *value field* that contains the attribute’s data

The basic attribute structure is shown in the following figure.



Figure 3-1: DICOM Attribute (Data Element) Structure

A DICOM attribute, identified by a (group number, element number) tag may be public or private. Attributes with an even group number are defined by the DICOM standard and are referred to as public tags. Attributes with an odd group number can be defined by users of the file format, but must conform to the same structure as standard elements. These are referred to as private tags.

DICOM Sequence Items

A DICOM attribute may be a *sequence*, which is a data element with a value representation of **SQ**. A sequence is an attribute that acts as a container for one or more items. A sequence can contain individual items, additional (nested) sequences of items, or sets of repeating items. These sets of repeating tags are identified as *groups* in this document.

File Compression and Transfer Syntax Support

The IDLffDicomEx object supports reading and writing to compressed files on Windows and UNIX platforms. There is no support for the JPEG compression algorithms on Macintosh. Using IDLffDicomEx properties and methods, you can return and modify the compression of a file. Use the [TRANSFER_SYNTAX](#) property to return the Transfer Syntax UID (0002,0010) associated with the DICOM file. Use the [IDLffDicomEx::ChangeTransferSyntax](#) method to change the transfer syntax of an object. The IDLffDicomEx object supports the following transfer syntaxes.

Compression Type	Transfer Syntax UID	Description
Implicit VR Little Endian	1.2.840.10008.1.2	Default transfer syntax for DICOM
Explicit VR Little Endian	1.2.840.10008.1.2.1	Little Endian data encoding
Explicit VR Big Endian	1.2.840.10008.1.2.2	Big Endian data encoding
JPEG Baseline	1.2.840.10008.1.2.4.50	Default Transfer Syntax for Lossy JPEG 8 Bit Image Compression
JPEG Extended (Process 2 & 4)	1.2.840.10008.1.2.4.51	Default Transfer Syntax for Lossy JPEG 12 Bit Image Compression (Process 4 only)
JPEG Lossless, Non-Hierarchical	1.2.840.10008.1.2.4.70	Lossless JPEG Image Compression. First-Order Prediction (Process 14 [Selection Value 1])

Table 3-1: Transfer Syntax Support

Compression Type	Transfer Syntax UID	Description
JPEG 2000, Lossless Only	1.2.840.10008.1.2.4.90	Lossless, reversible wavelet and color component transformation, and no quantization.
JPEG 2000, Lossy	1.2.840.10008.1.2.4.91	Lossy, irreversible wavelet transformation and color component transformation, and optional quantization.

Table 3-1: Transfer Syntax Support (Continued)

LffDicomEx Properties

The IDLffDicomEx object has the following properties.

- [BITS_ALLOCATED](#)
- [BITS_STORED](#)
- [COLUMNS](#)
- [FILENAME](#)
- [HIGH_BIT](#)
- [IMAGE_TYPE](#)
- [INSTANCE_NUMBER](#)
- [MODALITY](#)
- [NO_PIXEL_DATA](#)
- [NUMBER_OF_FRAMES](#)
- [PHOTOMETRIC_INTERPRETATION](#)
- [PIXEL_ASPECT_RATIO](#)
- [PIXEL_MAX](#)
- [PIXEL_MIN](#)
- [PIXEL_REPRESENTATION](#)
- [PIXEL_SPACING](#)
- [PLANAR_CONFIGURATION](#)
- [ROWS](#)
- [SAMPLES_PER_PIXEL](#)
- [SOP_CLASS_UID](#)
- [SOP_INSTANCE_UID](#)
- [TRANSFER_SYNTAX](#)

About Object Property Descriptions

Each property description includes a table similar to the following one.

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0100)	VR: US	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

The fields contain the following information:

- **Property Type** describes the property type associated with the property. If the property is *registered*, the property type will be one of the types described in [“Registered Property Data Types”](#) (*IDL Reference Guide*). If the property is not registered, this field will describe the generic IDL data type of the property value.
- **Name String** is the default value of the Name property attribute. If the property is registered, this is the value that appears in the left-hand column

when the property is displayed in a property sheet widget. If the property is not registered, this field will contain the words *not displayed*.

- **DICOM Attribute** is the (*group,element*) tag number of the attribute. See “[DICOM Attributes](#)” on page 300 for a list of attributes.
- **VR** is the Value Representation, which describes the data type of the attribute. See “[Value Representations](#)” on page 372 for definitions of the available VRs.
- **Multi-value** indicates whether the attribute has more than a single value in its value field.

Note

See “[IDLffDicomEx Overview](#)” on page 67 for more information about structure of a DICOM attribute.

- **Get, Set,** and **Init** describe whether the property can be specified as a keyword to the `GetProperty`, `SetProperty`, and `Init` methods, respectively.
- **Registered** describes whether the property is registered for display in a property sheet widget.

Properties with the word “Yes” in the “Get” column of the property table can be retrieved via `IDLffDicomEx::GetProperty`. Properties with the word “Yes” in the “Set” column in the property table can be set via `IDLffDicomEx::SetProperty`. Properties with the word “Yes” in the “Init” column of the property table can be specified during object initialization via `IDLffDicomEx::Init`.

Guidelines for Modifying IDLffDicomEx Properties

Several `IDLffDicomEx` object properties need to be set only when creating a new image object. There is no need to change these properties on an existing image, which can be a cloned image, or a read-only image. In fact, changing these properties on an existing image can result in the defined property values being inconsistent with the pixel data stored in the existing image, and the acceptable property values for the SOP Class of the image. Changing these properties does not change the characteristics of existing pixel data. To avoid propagating incorrect property values, set these properties only on a new image, and only prior to (or while) setting pixel data. See “[Specifying Pixel Data For a New Image](#)” on page 182 for a list of properties that must be set when assigning pixel data to a brand new image.

BITS_ALLOCATED

An integer that indicates the total number of bits allocated for each pixel sample. A pixel cell is made up of the pixel sample value as well as other pixel-related

information, such as overlay indications. The structure of each pixel sample value can be determined by the number of bits allocated ([BITS_ALLOCATED](#) property), the number of bits stored ([BITS_STORED](#) property), and the location of the most significant bit ([HIGH_BIT](#) property).

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0100)	VR: US	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See [“Guidelines for Modifying IDLffDicomEx Properties”](#) on page 71 for details.

BITS_STORED

An integer that indicates the number of bits stored for each pixel sample. The number of bits stored is less than or equal to the number of bits allocated, which determines the size of each pixel sample. A pixel cell is made up of the pixel sample value and other pixel-related information, such as overlay indications. The structure of each pixel sample value can be determined by the number of bits allocated ([BITS_ALLOCATED](#) property), the number of bits stored ([BITS_STORED](#) property), and the location of the most significant bit ([HIGH_BIT](#) property).

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0101)	VR: US	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See [“Guidelines for Modifying IDLffDicomEx Properties”](#) on page 71 for details.

COLUMNS

An integer that indicates the number of columns of pixels in an image.

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0011)	VR: US	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

FILENAME

A string that contains the filename associated with the object. The filename is defined when a new IDLffDicomEx object is created. The FILENAME property provides a convenient way to retrieve the filename passed in during object initialization.

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	n/a	VR: n/a	Multi-value: No
Get: Yes	Set: No	Init: No	Registered: No

HIGH_BIT

An integer that specifies the most significant bit within a zero-based pixel sample and determines the starting position of the bits used to store the pixel's value. A pixel cell is made up of the pixel sample value and other pixel-related information, such as overlay indications. The structure of each pixel sample value can be determined by the number of bits allocated ([BITS_ALLOCATED](#) property), the number of bits

stored (**BITS_STORED** property), and the location of the most significant bit (**HIGH_BIT** property).

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0102)	VR: US	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

IMAGE_TYPE

A string array that describes the type of image associated with a particular series. The string array can contain 2, 3, or 4 elements, the first two of which are required. The first and second elements describe the Pixel Data and Patient Examination characteristics. The third and fourth elements, which are optional, provide modality-specific and implementation-specific information.

Note

The value provided must be in all upper case letters. Lower or mixed case values will cause an error.

The following table shows allowable values for these elements.

Field	Possible String Values	Description
Pixel Data Characteristics	ORIGINAL	Pixel values are based on initial data.
	DERIVED	Pixel values have been generated from one or more other images.

Table 3-2: IMAGE_TYPE Values

Field	Possible String Values	Description
Patient Examination Characteristics	PRIMARY	Image created from direct patient examination.
	SECONDARY	Image created after patient examination.
Modality Specific Characteristics	Optional information object definitions related to the modality.	See the DICOM standard, <i>DICOM Part 3: Information Object Definitions</i> , for details.
Implementation Specific Characteristics	Other optional values.	This is a user-defined field.

Table 3-2: IMAGE_TYPE Values (Continued)

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	(0008,0008)	VR: CS	Multi-value: Yes (2 or more)
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

INSTANCE_NUMBER

A string that contains the identification (ID) number for an image.

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	(0020,0013)	VR: IS	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

MODALITY

A string that contains the type of equipment that acquired the data used to create the images in the series. See the DICOM standard, *DICOM Part 3: Information Object Definitions*, (PS 3.3-2003) C.7.3.1.1.1, for a list of the Modality Defined Terms.

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	(0008,0060)	VR: CS	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

NO_PIXEL_DATA

A scalar integer that determines whether pixel data is returned when you create an IDLffDicomEx object. Set this property to 1 to prevent pixel data from being loaded into memory and only return the tag information. Do not set this property or set the property value equal to zero return all DICOM file information including pixel data.

Setting this property provides a significant performance improvement when you only need attribute information. However, the pixel data is unavailable. If you attempt to access pixel data for an object that has this property set, IDL generates an error. You must create a new object in order access pixel data.

This property is not set by default.

Property Type	INTEGER		
Name String	<i>not displayed</i>		
Get: Yes	Set: No	Init: Yes	Registered: No

NUMBER_OF_FRAMES

An integer that specifies the number of frames contained within an image file. A DICOM image file can contain one image (frame) or multiple images (frames). When a DICOM image file contains more than one frame, the pixel data is concatenated into one array in the DICOM file. All related properties, such as [SAMPLES_PER_PIXEL](#) and [BITS_ALLOCATED](#), apply to all the frames. Some SOP Classes only support single frame images; consequently, this DICOM tag is not part of the set of tags that make up the class.

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0008)	VR: IS	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on a multi-frame image just created using the CREATE keyword of the [IDLffDicomEx::Init](#) method. There is no need to set this for a new single-frame image as the default value is 1. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

PHOTOMETRIC_INTERPRETATION

A string that contains the photometric interpretation of the pixel data. Photometric interpretation refers to how color (and/or intensity) is shown within an image. The following table provides a list of possible string values.

Note

The value provided must be in all upper case letters. Lower or mixed case values will cause an error.

String Value	Description
MONOCHROME1	Represents a monochrome (grayscale) image plane where the minimum pixel value is white. The SAMPLES_PER_PIXEL property equals 1.
MONOCHROME2	Represents a monochrome (grayscale) image plane where the minimum pixel value is black. The SAMPLES_PER_PIXEL property equals 1.
PALETTE COLOR	Represents a color (indexed) image plane. Each pixel value is mapped through a color look-up table (LUT). The SAMPLES_PER_PIXEL property equals 1.
RGB	Represents a color image containing red, green, and blue (RGB) planes. The SAMPLES_PER_PIXEL property equals 3.
HSV	Represents a color image containing hue, saturation, and value planes. The SAMPLES_PER_PIXEL property equals 3.
CMYK	Represents a color image described by cyan, magenta, yellow, and black planes. The SAMPLES_PER_PIXEL property equals 4.

Table 3-3: *PHOTOMETRIC_INTERPRETATION* Values

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0004)	VR: CS	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

PIXEL_ASPECT_RATIO

This is a multi-valued property that is stored as a one dimensional array. The first element is the vertical size of each pixel in millimeters. The second element is the horizontal size of each pixel in millimeters.

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0034)	VR: IS	Multi-value: Yes (2)
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

PIXEL_MAX

An integer that indicates the maximum pixel value within an image. This value (also known as Largest Image Pixel Value) is read from the attribute (0028,0107) in the DICOM file when an object is created. This value may not match the data’s maximum pixel value. The value of this tag reflects the value assigned to it, which may be a user-assigned value other than the data’s actual maximum pixel value. You can use IDL’s `MAX` function to return the largest value in the pixel data array. The pixel value of this property is either an unsigned or signed integer based on the value of the `PIXEL_REPRESENTATION` property.

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0107)	VR: US or SS	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

PIXEL_MIN

An integer that indicates the minimum pixel value within an image. This value (also known as Smallest Image Pixel Value) is read from the attribute (0028,0106) in the DICOM file when an object is created. This value may not match the data’s minimum pixel value. The value of this tag reflects the value assigned to it, which may be a user-assigned value other than the data’s actual minimum pixel value. You can use IDL’s `MIN` function to return the smallest value in the pixel data array. The pixel value of this property is either an unsigned or signed integer based on the value of the `PIXEL_REPRESENTATION` property.

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0106)	VR: US or SS	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

PIXEL_REPRESENTATION

An integer that indicates the data representation of the pixels within an image as follows:

- 0 = Unsigned Integer
- 1 = Signed Integer

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0103)	VR: US	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the [IDLffDicomEx::Init](#) method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

PIXEL_SPACING

A two-element array in which the first value is the physical distance within the patient between the center of each adjacent row pixel in millimeters (mm). The second value of the property is the physical distance within the patient between the center of each adjacent column pixel in millimeters (mm).

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0030)	VR: DS	Multi-value: Yes (2)
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the [IDLffDicomEx::Init](#) method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

PLANAR_CONFIGURATION

A value that indicates whether the pixel data of three- or four-plane images are sent color-by-pixel or color-by-plane. The pixel data has three or four planes if the value for the SAMPLES_PER_PIXEL property is greater than one.

Value	Description
0	Color-by-pixel or pixel interleaving – the value for the first pixel within the plane is sent, followed by the value for the first pixel in the second plane (R1, G1, B1, R2, G2, B2, ...).
1	Color-by-plane or image interleaving – values for all of the first plane's pixels are sent, followed by all of the pixel values in the next plane (R1, R2, R3, ..., G1, G2, G3, ... and B1, B2, B3, ...).

Table 3-4: PLANAR_CONFIGURATION Values

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0006)	VR: US	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See [“Guidelines for Modifying IDLffDicomEx Properties”](#) on page 71 for details.

ROWS

An integer that indicates the number of rows of pixels in an image.

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0010)	VR: US	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

SAMPLES_PER_PIXEL

The number of separate planes in the image. The image can contain either one, three, or four planes.

Value	Description
1	Contains a single plane representing a monochrome (grayscale) image or an indexed image with an associated color look-up table (LUT).
3	Contains three planes representing an RGB (red, green, blue), or HSV (hue, saturation, and value) image.
4	Contains four planes representing a CMYK (cyan, magenta, yellow, black) image.

Table 3-5: SAMPLES_PER_PIXEL Values

Property Type	Integer		
Name String	<i>not displayed</i>		
DICOM Attribute	(0028,0002)	VR: US	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property should be set only on an image just created using the CREATE keyword of the `IDLffDicomEx::Init` method. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

SOP_CLASS_UID

The unique identifier (UID) of the class of the service-object pair (SOP) associated with a source image. See “[SOP_CLASS](#)” on page 171 for available SOP class options.

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	(0008,0016)	VR: UI	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property value is automatically generated when you set the CREATE keyword of the `IDLffDicomEx::Init` method, and should not need to be further modified. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

SOP_INSTANCE_UID

The unique identifier (UID) of the image. This identifier is used when the image is transferred to or from a database and to identify the image within a hierarchical tree of information. An unique identifier is generated for each newly created or cloned image.

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	(0008,0018)	VR: UI	Multi-value: No
Get: Yes	Set: Yes	Init: No	Registered: No

Note

This property value is automatically generated when you set the CREATE keyword of the `IDLffDicomEx::Init` method, and should not need to be further modified. See “[Guidelines for Modifying IDLffDicomEx Properties](#)” on page 71 for details.

TRANSFER_SYNTAX

This property returns two values: the transfer syntax UID (unique identifier) and its description. Use the `IDLffDicomEx::ChangeTransferSyntax` method to modify the

file syntax. There is no support for the JPEG compression algorithms on Macintosh. See [“File Compression and Transfer Syntax Support”](#) on page 68 for information on supported transfer syntaxes.

Property Type	String		
Name String	<i>not displayed</i>		
DICOM Attribute	(0002,0010)	VR: UI	Multi-value: Yes (2)
Get: Yes	Set: No	Init: No	Registered: No

IDLffDicomEx::AddGroup

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Examples](#) | [Version History](#)

The `IDLffDicomEx::AddGroup` function method creates a series of repeating tags within a sequence. For example, the same set of 10 tags could be repeated 4 times in one sequence, which means the sequence would have 4 groups, each with the same 10 tags.

When calling `AddGroup` the *DicomTag* argument specifies a sequence (SQ) attribute. If the sequence does not exist, the `AddGroup` method creates the sequence before creating the group in this sequence. Once the group has been created, member items can be added via the [IDLffDicomEx::SetValue](#) method using the return value from this method as the SEQID value

Note

Changes are not written to the DICOM file until you call the [IDLffDicomEx::Commit](#) method. When you commit changes, all sequence identifiers are invalidated. You need to call [IDLffDicomEx::GetValue](#) to re-access the sequence identifiers. See “[Adding Groups to a Nested Sequence](#)” on page 89 for an example.

When adding a group to an existing sequence, existing sequence items are placed in a group, and the new group is then added to the sequence. When adding groups to a nested sequence (one sequence contained within another), specify the same PARENTSEQID for the group as was specified for the sequence. See “[Adding Groups to a Nested Sequence](#)” on page 89 for sample code.

Syntax

```
Result = Obj->[IDLffDicomEx::]AddGroup (DicomTag
    [, PARENTSEQID=integer] )
```

Return Value

Returns a long integer containing the group identifier for the newly created group. This identifier can be used by other methods that use the SEQID keyword such as [IDLffDicomEx::GetValue](#) and [IDLffDicomEx::SetValue](#) methods.

Arguments

DicomTag

A string that identifies the group and element of a DICOM sequence (SQ) attribute in the form 'XXXX, XXXX'. The *DicomTag* argument must reference a public tag that is part of the standard IOD for the image type and must be of the [SQ](#) VR type. See “[DICOM Attributes](#)” on page 300 for a list of tags.

Keywords

PARENTSEQID

Set this keyword only if adding the new group to an existing sequence. Use this keyword to specify a parent sequence identifier to add the group to as follows:

- If set to a non-zero value, then the group will be added as a member item to the specified nested sequence. This sequence identifier may have been returned via a previous call to the [IDLffDicomEx::AddSequence](#) method.
- If set to 0 or not specified, then the group is added to a sequence specified by the *DicomTag* argument, which exists at the root level of the DICOM file. This is the default.

Examples

Adding Groups to a Root Level Sequence

The following example adds a public sequence containing three groups to the root level of a selected file. This public sequence is typically associated with Ultrasound (US) files. Use the NON-CONFORMING keyword when creating a clone in order to avoid errors encountered when attempting to add non-standard attributes to the DICOM file. The new groups within the sequence are printed to the Output Log window.

Note

For an example that adds groups to a nested sequence, see “[Adding Groups to a Nested Sequence](#)” on page 89.

Note

This example does not write the cloned file to memory. To do so, simply use the [IDLffDicomEx::Commit](#) method.

```

PRO dicom_addgroup_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('',SUBDIRECTORY=['examples','data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
; Set the NON_CONFORMING keyword to be able to add a public SQ
; of repeating items typically associated with US files to
; the selected file.
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile, /NON_CONFORMING)

; Add a public group to the root level of the file. A
; sequence with the value specified for the DICOM
; attribute is automatically created if it does not exist.
; The AddGroup calls add a Sequence of Ultrasound Regions
; (0018,6011) with 3 sets of repeating tags.

; Add two tags to each group.
groupId_1 = oImg->AddGroup('0018,6011')
oImg->SetValue,'0018,6012', 'US', 1, SEQID = groupId_1
oImg->SetValue,'0018,6014', 'US', 2, SEQID = groupId_1

groupId_2 = oImg->AddGroup('0018,6011')
oImg->SetValue,'0018,6012', 'US', 3, SEQID = groupId_2
oImg->SetValue,'0018,6014', 'US', 4, SEQID = groupId_2

groupId_3 = oImg->AddGroup('0018,6011')
oImg->SetValue,'0018,6012', 'US', 5, SEQID = groupId_3
oImg->SetValue,'0018,6014', 'US', 6, SEQID = groupId_3

; Print a range including the new tags to
; the Output Log window.
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0018,6011', STOP_TAG='0020,0000')

; Format the output.
PRINT, FORMAT= $
    '(%" %3s, %2s, %-12s, %3s, %5s, %30s, %8s)", $
    'IDX', 'LVL', 'TAG', 'VR', 'SEQID', $
    'DESCRIPTION', 'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

    ; If the item is nested within another item, indicate the

```



```

; level using > symbol.
IF (vTags[xx].Level GT 0) THEN BEGIN
    vLvl = STRJOIN(REPLICATE('>',vTags[xx].Level))
    vtg = vLvl + vTags[xx].Tag
ENDIF ELSE BEGIN
    vtg = vTags[xx].Tag
ENDELSE

; If the tags are in a group, indicate this.
IF (vTags[xx].GroupNum GT 0) THEN BEGIN
    PRINT, FORMAT='(%"15s, %1d")', 'Group', vTags[xx].GroupNum
ENDIF

; Print the fields of the structure.
PRINT, FORMAT = $
    ('%"3d, %2d, %-12s, %3s, %5d, %30s, %8s")', $
    xx, vTags[xx].Level, vtg, vTags[xx].VR, vTags[xx].SeqId, $
    vTags[xx].Description, vTags[xx].Value

ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

END

```

This program generates the following output. A root-level sequence (SQ) attribute (0018,6011) and three sets of repeating tags are added to the file.

IDX	LV	TAG	VR	SEQID	DESCRIPTION
0	0	0018,6011	SQ	184	Sequence of Ultrasound Regions.
		Group, 1			
1	1	>0018,6012	US	185	Region Spatial Format.
2	1	>0018,6014	US	185	Region Data Type.
		Group, 2			
3	1	>0018,6012	US	186	Region Spatial Format.
4	1	>0018,6014	US	186	Region Data Type.
		Group, 3			
5	1	>0018,6012	US	187	Region Spatial Format.
6	1	>0018,6014	US	187	Region Data Type.

Figure 3-2: Adding a Series of Repeating Tags to the Root Level

Adding Groups to a Nested Sequence

The following example adds two groups to a nested sequence by first adding a sequence ('0054,0016') to the root level of the cloned file. The code then adds an

attribute ('0018,1071') to the sequence. Another sequence ('0054,0300') is added to the root level sequence using the `IDLffDicomEx::AddGroup` method.

After adding the first set of repeating tags, call the `IDLffDicomEx::Commit` method to save the changes. Before you can add a second group of tags, you need to retrieve sequence identifiers. After returning the sequence identifier using `IDLffDicomEx::GetValue`, add the second group of tags. The root sequence is defined as the PARENTSEQID of the two groups.

Note

You do not have to call Commit after adding the first group of tags. This example calls Commit at this point simply to illustrate re-accessing the sequence identifier reference, which is lost when Commit is called.

Use the NON-CONFORMING keyword when creating a clone in order to avoid errors encountered when attempting to add non-standard attributes to the DICOM file. The new sequences and groups are printed to the Output Log window.

```
PRO dicom_addgrouptonestedseq_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
; Set the NON_CONFORMING keyword to be able to add a public SQ
; of radiopharmaceutical items to any file.
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile, /NON_CONFORMING)

; Add a sequence; Radiopharmaceutical Information Sequence.
vRootSeq = oImg->AddSequence('0054,0016')

; Add an attribute within the sequence.
oImg->SetValue, '0018,1071', 'DS', '0', SEQID=vRootSeq

; Add two hypothetical groups to the nested sequence,
; (0054,0300), the Radionuclide Code Sequence. Calling AddGroup
; with this sequence specified adds the sequence. Add two groups,
; each consisting of two tags. The parent sequence of the repeating
; tags (which are contained within a sequence) is the root
; sequence.
groupId_1 = oImg->AddGroup('0054,0300', PARENTSEQID=vRootSeq)
oImg->SetValue, '0008,0100', 'SH', 'Tc-99m', SEQID = groupId_1
oImg->SetValue, '0008,0102', 'SH', '99SDM', SEQID = groupId_1
```

```

; Commit the changes.
oImg->Commit

; After committing changes you must re-access any sequence
; identifiers. Failure to do so will cause an error.
vNewSeqid = oImg->GetValue('0054,0016')
groupId_2 = oImg->AddGroup('0054,0300', PARENTSEQID=vNewSeqid)
oImg->SetValue,'0008,0100', 'SH', 'Tc-99m', SEQID = groupId_2
oImg->SetValue,'0008,0102', 'SH', '99SDM', SEQID = groupId_2

; Print a range including the new tags to
; the Output Log window.
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0054,0016', STOP_TAG='0056,0000')

; Format the output.
PRINT, FORMAT= $
    '(%"3s, %2s, %-12s, %3s, %5s, %30s, %10s)", $
    'IDX', 'LVL', 'TAG', 'VR', 'SEQID', $
    'DESCRIPTION', 'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

    ; If the item is nested within another item, indicate the
    ; level using > symbol.
    IF (vTags[xx].Level GT 0) THEN BEGIN
        vLvl = STRJOIN(REPLICATE('>',vTags[xx].Level))
        vtg = vLvl + vTags[xx].Tag
    ENDIF ELSE BEGIN
        vtg = vTags[xx].Tag
    ENDELSE

    ; If the tags are in a group, indicate this.
    IF (vTags[xx].GroupNum GT 0) THEN BEGIN
        PRINT, FORMAT='(%"15s, %1d)", 'Group', vTags[xx].GroupNum
    ENDIF

    ; Print the fields of the structure.
    PRINT, FORMAT = $
        '(%"3d, %2d, %-12s, %3s, %5d, %30s, %10s)", $
        xx, vTags[xx].Level, vtg, vTags[xx].VR, vTags[xx].SeqId, $
        vTags[xx].Description, vTags[xx].Value

ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

```

```

; Delete the file to avoid duplicate file name errors when running
; example multiple times.
FILE_DELETE, path + 'aImgClone.dcm', /ALLOW_NONEXISTENT

END

```

This produces the following output. Notice that sequence (0054,0300) is nested inside sequence (0054,0016).

IDX.	LV.	TAG	VR.	SEQID.	DESCRIPTION.	VALUE
0.	0.	0054,0016	SQ.	120.	Radiopharmaceutical Informatio.	
1.	1.	>0018,1071	DS.	121.	Radiopharmaceutical Volume.	0
2.	1.	>0054,0300	SQ.	121.	Radionuclide Code Sequence.	
		Group, 1				
3.	2.	>>0008,0100	SH.	122.	Code Value.	Tc-99m
4.	2.	>>0008,0102	SH.	122.	Coding Scheme Designator.	99SDM
		Group, 2				
5.	2.	>>0008,0100	SH.	123.	Code Value.	Tc-99m
6.	2.	>>0008,0102	SH.	123.	Coding Scheme Designator.	99SDM

Figure 3-3: Adding Sets of Repeating Tags to a Nested Sequence

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::AddPrivateGroup

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Examples](#) | [Version History](#)

The `IDLffDicomEx::AddPrivateGroup` function method creates a group within a private sequence. A group is a set of repeating tags in a sequence. For example a sequence can have the same set of 10 tags repeat itself 4 times in one sequence, which means the sequence would have 4 groups, each with the same 10 tags.

When calling `AddPrivateGroup`, the *PrivateCode*, *Group*, and *Element* arguments identify the characteristics and placement of the group. The optional `PARENTSEQID` keyword can be used to specify the private sequence (SQ) attribute to which the group is to be added. If this value is non-zero then it identifies a sequence by sequence identifier, which may have been returned by a previous call to [IDLffDicomEx::AddPrivateSequence](#) or [IDLffDicomEx::GetPrivateValue](#). If the sequence does not exist, the `AddPrivateGroup` method creates the sequence before creating the group in this sequence. Once the group has been created, member items can be added via the [IDLffDicomEx::SetPrivateValue](#) method using the return value from this method, the identifier of the new group.

Note

The new group is not written to the DICOM file until you call the [IDLffDicomEx::Commit](#) method. When you commit changes, the sequence identifier is invalidated. You need to call [IDLffDicomEx::GetValue](#) to re-access the sequence identifier.

When adding a group to an existing private sequence that does not contain other groups, existing sequence items are placed in a group, and the new group is then added to the sequence.

Syntax

```
Result = Obj->[IDLffDicomEx::]AddPrivateGroup (PrivateCode, Group, Element  
[, PARENTSEQID=integer] )
```

Return Value

Returns a long integer containing the group identifier for the newly created group. This identifier can be used by other methods that use the `SEQID` keyword such as [IDLffDicomEx::GetPrivateValue](#) and [IDLffDicomEx::SetPrivateValue](#) methods.

Arguments

PrivateCode

A string identification code that identifies the private block of data. Within a given private group *PrivateCode* labels are stored sequentially in the element addresses ranging from '0010' to '00FF'. For example, the string value stored at DICOM tag address '0029,0010' is the *PrivateCode* for the block of data tagged at '0029,1000' - '0029,10FF'. The label stored at '0029,0011' would be the *PrivateCode* for the data in tags '0029,1100' - '0029,11 FF'.

Group

A string identifying the group tag number of the private attribute (the first four digits of a DICOM tag). This must be an odd number and in the form 'XXXX'. If this does not reference an existing sequence, then a new private sequence is created.

Element

A string identifying the last two digits of the element associated with the private attribute. This must be in the form 'XX'. Valid values are 10 - FF.

Note

The first two digits of the *Element* are implicit in the *PrivateCode* argument.

Keywords

PARENTSEQID

Set this keyword only if adding the new group (the new sequence and its repeating sets of tags) to an existing sequence. Use this keyword to specify a parent sequence identifier of a sequence to add the new group to as follows:

- If set to a non-zero value, then the group will be added as a member item to the private sequence associated with this parent sequence identifier. This sequence identifier may have been returned via a previous call to the [IDLffDicomEx::AddPrivateSequence](#) method.
- If set to 0 or not specified, then the group is added to a private sequence at the root level of the DICOM file. This is the default value.

Examples

This section features the following two examples:

- [“Adding Private Sets of Tags to a Root-level Sequence”](#) on page 95
- [“Adding Private Sets of Tags to a Nested Sequence”](#) on page 97

Adding Private Sets of Tags to a Root-level Sequence

The following example adds a two sets of repeating private attributes to a root level sequence in a DICOM file. There is no need to use the `NON_CONFORMING` keyword when creating the clone if you will only be adding private attributes, which are not regulated by the DICOM standard. For an example that adds private groups to a nested sequenced, see [“Adding Private Sets of Tags to a Nested Sequence”](#) on page 97.

Note

The cloned file containing these changes is not written to disk. To persist the file, call the `IDLffDicomEx::Commit` method.

```
PRO dicom_setprivaterootgroup_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH(' ', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile)

; Add two sets of repeating tags (groups) to a private sequence
; (0051, 0012), which is created by AddPrivateGroup. This sequence
; exists at the root-level of the DICOM file. Add two sets of
; repeating tags to the root sequence.
vGrp1 = oImg->AddPrivateGroup('Root Private SQ', '0051', '12')
oImg->SetPrivateValue, 'Root Private SQ', '0051', '14', 'ST', $
    'gr1Tag1', SEQID=vGrp1
oImg->SetPrivateValue, 'Root Private SQ', '0051', '15', 'ST', $
    'gr1Tag2', SEQID=vGrp1
vGrp2 = oImg->AddPrivateGroup('Root Private SQ', '0051', '12')
oImg->SetPrivateValue, 'Root Private SQ', '0051', '14', 'ST', $
    'gr2Tag1', SEQID=vGrp2
oImg->SetPrivateValue, 'Root Private SQ', '0051', '15', 'ST', $
    'gr2Tag2', SEQID=vGrp2
```

```

; Print a range including the new tags to
; the Output Log window.
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0051,0000', STOP_TAG='0057,0000')

; Format the output.
PRINT, FORMAT= $
    '(%"3s, %2s, %-12s, %3s, %7s, %3s, %5s, %12s, %15s)", $
    'IDX', 'LVL', 'TAG', 'VR', 'LEN', 'CNT', 'SEQID', $
    'DESCRIPTION', 'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

    ; If the item is nested within another item, indicate the
    ; level using > symbol.
    IF (vTags[xx].Level GT 0) THEN BEGIN
        vLvl = STRJOIN(REPLICATE('>',vTags[xx].Level))
        vtg = vLvl + vTags[xx].Tag
    ENDIF ELSE BEGIN
        vtg = vTags[xx].Tag
    ENDELSE

    ; If the tags are in a group, indicate this.
    IF (vTags[xx].GroupNum GT 0) THEN BEGIN
        PRINT, FORMAT='(%"15s, %1d)", 'Group', vTags[xx].GroupNum
    ENDIF

    ; Print the fields of the structure.
    PRINT, FORMAT = $
        '(%"3d, %2d, %-12s, %3s, %7d, %3d, %5d, %12s, %15s)", $
        xx, vTags[xx].Level, vtg, vTags[xx].VR, vTags[xx].Length, $
        vTags[xx].ValueCount, vTags[xx].SeqId, $
        vTags[xx].Description, vTags[xx].Value

ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

END

```


Running the previous example creates private sets of repeating tags as shown in the following figure.

IDX	LV	TAG	VR	LEN	CNT	SEQID	DESCRIPTION	VALUE
0	0	0051,0010	LO	15	1	291		Root Private SQ
1	0	0051,1012	SQ	2	2	291		
		Group, 1						
2	1	>0051,0010	LO	15	1	292		Root Private SQ
3	1	>0051,1014	ST	7	1	292		gr1Tag1
4	1	>0051,1015	ST	7	1	292		gr1Tag2
		Group, 2						
5	1	>0051,0010	LO	15	1	293		Root Private SQ
6	1	>0051,1014	ST	7	1	293		gr2Tag1
7	1	>0051,1015	ST	7	1	293		gr2Tag2

Figure 3-4: Adding Sets of Private Tags to a Root-level Sequence

Adding Private Sets of Tags to a Nested Sequence

The following example builds on the previous example and simply adds private sets of tags to a nested sequence.

Note

This example does not save the cloned file to disk. To do so, call the `IDLffDicomEx::Commit` method.

```
PRO dicom_setprivatenestedgroup_doc

; Select a DICOM file.
sfile = DIALOG_PICKFILE(PATH='examples\data', $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path+'\'+'aImgClone.dcm', $
    CLONE=sfile)

; Add a private sequence to the root level of the file.
vRootSeq = oImg->AddPrivateSequence('Root Private SQ', $
    '0051', '12')

; Add two sets of repeating tags (groups) to a private sequence
; (0017,0012, which is created by AddPrivateGroup) that is nested
; in the root sequence. The parent sequence identifier of each set
; of repeating tags is the root sequence.
vGrp1 = oImg->AddPrivateGroup('Nested Private SQ', '0017', '12', $
    PARENTSEQID=vRootSeq)
oImg->SetPrivateValue, 'Nested Private SQ', '0017', '14', 'ST', $
    'gr1Tag1', SEQID=vGrp1
```

```

oImg->SetPrivateValue, 'Nested Private SQ', '0017', '15', 'ST', $
    'gr1Tag2', SEQID=vGrp1
vGrp2 = oImg->AddPrivateGroup('Nested Private SQ', '0017', '12', $
    PARENTSEQID=vRootSeq)
oImg->SetPrivateValue, 'Nested Private SQ', '0017', '14', 'ST', $
    'gr2Tag1', SEQID=vGrp2
oImg->SetPrivateValue, 'Nested Private SQ', '0017', '15', 'ST', $
    'gr2Tag2', SEQID=vGrp2

; Print a range including the new tags to
; the Output Log window.
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0051,0000', STOP_TAG='0057,0000')

; Format the output.
PRINT, FORMAT= $
    '(%" %3s, %2s, %-12s, %3s, %7s, %3s, %5s, %12s, %20s)", $
    'IDX', 'LVL', 'TAG', 'VR', 'LEN', 'CNT', 'SEQID', $
    'DESCRIPTION', 'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

    ; If the item is nested within another item, indicate the
    ; level using > symbol.
    IF (vTags[xx].Level GT 0) THEN BEGIN
        vLvl = STRJOIN(REPLICATE('>', vTags[xx].Level))
        vtg = vLvl + vTags[xx].Tag
    ENDIF ELSE BEGIN
        vtg = vTags[xx].Tag
    ENDELSE

    ; If the tags are in a group, indicate this.
    IF (vTags[xx].GroupNum GT 0) THEN BEGIN
        PRINT, FORMAT='(%" %15s, %1d)", 'Group', vTags[xx].GroupNum
    ENDIF

    ; Print the fields of the structure.
    PRINT, FORMAT = $
        '(%" %3d, %2d, %-12s, %3s, %7d, %3d, %5d, %12s, %20s)", $
        xx, vTags[xx].Level, vtg, vTags[xx].VR, vTags[xx].Length, $
        vTags[xx].ValueCount, vTags[xx].SeqId, $
        vTags[xx].Description, vTags[xx].Value

ENDFOR

; Clean up references.
OBJ_DESTROY, oImg
END

```

The output of the previous example appears similar to the following figure. The root level sequence (line 1) contains a nested sequence (line 3) with two groups of repeating tags.

IDX	LV	TAG	VR	LEN	CNT	SEQID	DESCRIPTION	VALUE
0	0	0051,0010	LO	15	1	309		Root Private SQ
1	0	0051,1012	SQ	1	1	309		
2	1	>0017,0010	LO	17	1	310		Nested Private SQ
3	1	>0017,1012	SQ	2	2	310		
		Group, 1						
4	2	>>0017,0010	LO	17	1	311		Nested Private SQ
5	2	>>0017,1014	ST	7	1	311		gr1Tag1
6	2	>>0017,1015	ST	7	1	311		gr1Tag2
		Group, 2						
7	2	>>0017,0010	LO	17	1	312		Nested Private SQ
8	2	>>0017,1014	ST	7	1	312		gr2Tag1
9	2	>>0017,1015	ST	7	1	312		gr2Tag2

Figure 3-5: Adding Private Groups (Repeating Tags) to a Nested Sequence

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::AddPrivateSequence

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The `IDLffDicomEx::AddPrivateSequence` function method creates a new private sequence. When calling `AddPrivateSequence` the *PrivateCode*, *Group*, and *Element* arguments identify the characteristics and placement of the private sequence.

The optional `PARENTSEQID` keyword can be used to create a nested sequence, placing the new private sequence within an existing sequence. This existing sequence is identified by a sequence identifier, which may have been returned by a previous call to [IDLffDicomEx::AddPrivateSequence](#) or [IDLffDicomEx::GetPrivateValue](#). Once the sequence has been created, member items can be added via the [IDLffDicomEx::SetPrivateValue](#) method using the return value from this method, the identifier of the new sequence.

Note

The new sequence is not written to the DICOM file until you call the [IDLffDicomEx::Commit](#) method. When you commit changes, the sequence identifier is invalidated. You need to call [IDLffDicomEx::GetPrivateValue](#) to re-access the sequence identifier.

Syntax

Result = Obj->[[IDLffDicomEx::](#)]AddPrivateSequence (*PrivateCode*, *Group*, *Element* [, `PARENTSEQID=integer`])

Return Value

Returns a long integer containing the sequence identifier for the newly created sequence. This identifier can be used by other methods that use the `SEQID` keyword such as [IDLffDicomEx::GetPrivateValue](#) and [IDLffDicomEx::SetPrivateValue](#) methods.

Arguments

PrivateCode

A string identification code that identifies the private block of data. Within a given private group *PrivateCode* labels are stored sequentially in the element addresses ranging from '0010' to '00FF'. For example, the string value stored at DICOM tag address '0029,0010' is the *PrivateCode* for the block of data tagged at

'0029,1000' - '0029,10FF'. The label stored at '0029,0011' would be the *PrivateCode* for the data in tags '0029,1100' - '0029,11 FF'.

Group

A string identifying the group tag number of the private attribute (the first four digits of a DICOM tag). This must be an odd number and in the form 'XXXX'. If this does not reference an existing sequence, then a new private sequence is created.

Element

A string identifying the last two digits of the element associated with the private attribute. This must be in the form 'XX'. Valid values are 10 - FF.

Note

The first two digits of the *Element* are implicit in the *PrivateCode* argument.

Note

If the *PrivateCode* differs, but the *Group*, *Element* combination of arguments already exists, the *Element* value will be internally incremented to avoid overwriting the existing sequence. To modify existing sequences, use the [IDLffDicomEx::SetPrivateValue](#) method.

Keywords

PARENTSEQID

Set this keyword only if adding the new sequence to an existing sequence. Use this keyword to specify a parent sequence as follows:

- If set to a non-zero value, then the sequence will be added as a member item to the private sequence associated with this parent sequence identifier. This sequence identifier may have been returned via a previous call to the [IDLffDicomEx::AddPrivateSequence](#) method.
- If set to 0 or not specified, then the sequence is added to a private sequence at the root level of the DICOM file. This is the default value.

Example

The following example adds a private attribute to the root level of the DICOM file, a private sequence, and two items in the private sequence. This example shows how to

add private attributes, but does not write the tags to the cloned file. The new private attributes are printed to the Output Log window.

Note

This example does not write the cloned file to memory. To do so, simply use the [IDLffDicomEx::Commit](#) method.

```

PRO dicom_setprivate_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE($
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile)

; Add private tags. The following are hypothetical.
; Create a multi-valued tag at the root level.
arr = [1, 2, 3, 4]
oImg->SetPrivateValue, 'Private Test', '0053', '10', 'SS', arr

; Create a sequence at the root level.
vSeqID = oImg->AddPrivateSequence('VOI Min,Max', '0055', '12')

; Add items to the sequence, specifying the sequence identifier
; returned by AddPrivateSequence.
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '13', 'IS', '215', $
    SEQID=vSeqID
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '14', 'IS', '234', $
    SEQID=vSeqID

; Print a range including the new tags to
; the Output Log window.
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0053,0000', STOP_TAG='0057,0000')

; Format the output.
PRINT, FORMAT= $
    '(%" %3s, %2s, %12s, %3s, %12s, %20s)", $
    'IDX', 'LVL', 'TAG', 'VR', 'DESCRIPTION', 'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

    ; If the item is nested within another item, indicate the

```

```

; level using > symbol.
IF (vTags[xx].Level GT 0) THEN BEGIN
    vLvl = STRJOIN(REPLICATE('>',vTags[xx].Level))
    vtg = vLvl + vTags[xx].Tag
ENDIF ELSE BEGIN
    vtg = vTags[xx].Tag
ENDELSE

; If the tags are in a group, indicate this.
IF (vTags[xx].GroupNum GT 0) THEN BEGIN
    PRINT, FORMAT='(%" %15s, %1d)", 'Group', vTags[xx].GroupNum
ENDIF

; Print the fields of the structure.
PRINT, FORMAT = $
    '(%" %3d, %2d, %-12s, %3s, %12s, %20s)", $
    xx, vTags[xx].Level, vtg, vTags[xx].VR, $
    vTags[xx].Description, vTags[xx].Value

ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

END

```

This example creates the following output.

IDX,	LV,	TAG,	VR,	DESCRIPTION,	VALUE
0,	0,	0053,0010	, LO,	,	Private Test
1,	0,	0053,1010	, SS,	,	1\2\3\4
2,	0,	0055,0010	, LO,	,	VOI Min,Max
3,	0,	0055,1012	, SQ,	,	
4,	1,	>0055,0010	, LO,	,	VOI Min,Max
5,	1,	>0055,1013	, IS,	,	215
6,	1,	>0055,1014	, IS,	,	234

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::AddSequence

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The `IDLffDicomEx::AddSequence` function method creates a new sequence. The *DicomTag* argument specifies a sequence (SQ) attribute, which must be part of the standard IOD (Information Object Definition) for the DICOM file type (unless the `NON_CONFORMING` keyword was set when the `IDLffDicomEx` object was created using the `IDLffDicomEx::Init` method).

The optional `PARENTSEQID` keyword can be used to create a nested sequence, placing the new sequence within an existing sequence. This existing sequence is identified by a sequence identifier, which may have been returned by a previous call to `IDLffDicomEx::AddSequence` or `IDLffDicomEx::GetValue`.

Once the sequence has been created, member items can be added via the `IDLffDicomEx::SetValue` method using the return value from this method, the identifier of the new sequence.

Note

Changes are not written to the DICOM file until you call the `IDLffDicomEx::Commit` method. When you commit changes, all sequence identifiers are invalidated. You need to call `IDLffDicomEx::GetValue` to re-access the sequence identifiers. See “Adding Groups to a Nested Sequence” on page 89 for an example.

Syntax

```
Result = Obj->[IDLffDicomEx::]AddSequence (DicomTag  
[, PARENTSEQID=integer ] )
```

Return Value

Returns a long integer containing the sequence identifier for the newly created sequence. This identifier can be used by other methods that use the `SEQID` keyword such as `IDLffDicomEx::GetValue` and `IDLffDicomEx::SetValue` methods.

Arguments

DicomTag

A string that identifies the group and element of a DICOM sequence (SQ) attribute in the form 'XXXX,XXXX'. The *DicomTag* argument must reference a public tag that is part of the standard IOD for the image type and must be of the **SQ** VR type. See “[DICOM Attributes](#)” on page 300 for a list of tags.

Keywords

PARENTSEQID

Set this keyword only if adding the new sequence to an existing sequence. Use this keyword to specify a parent sequence identifier to add the sequence to as follows:

- If set to a non-zero value (a sequence identifier), then the sequence will be added to the existing, specified sequence. This sequence identifier may have been returned via a previous call to the [IDLffDicomEx::AddSequence](#) method.
- If set to 0 or not specified, then the sequence is added to the root level of the DICOM file. This is the default.

Example

The following example adds a sequence to the root-level of a cloned DICOM file and, a nested sequence containing attributes within the first sequence. The `NON_CONFORMING` keyword is set when the clone is created in order to avoid errors encountered when attempting to add non-standard attributes to the selected DICOM file. The newly added attributes are printed to the IDL Output Log window.

Note

For an example that adds groups of repeating tags to a sequence, see the “Examples” section of [“IDLffDicomEx::AddGroup”](#) on page 86.

Note

This example does not write the cloned file to memory. To do so, simply use the [IDLffDicomEx::Commit](#) method.

```
PRO dicom_addpublicattributes_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE($
    PATH=FILEPATH(' ', SUBDIRECTORY=['examples', 'data']), $
```

```

        TITLE='Select DICOM Patient File', FILTER='*.dcm', $
        GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
; Set the NON_CONFORMING keyword to be able to add a public SQ
; of radiopharmaceutical items to any file.
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile, /NON_CONFORMING)

; Add a root-level sequence (Radiopharmaceutical Information).
; *****
vRootSeq = oImg->AddSequence('0054,0016')

; Add an attribute within the sequence.
; *****
oImg->SetValue, '0018,1071', 'DS', '0', SEQID=vRootSeq

; Add a nested sequence (Radionuclide Code Sequence).
; *****
vNestSeq = oImg->AddSequence('0054,0300', PARENTSEQID=vRootSeq)

; Add two items to the nested sequence.
; *****
oImg->SetValue, '0008,0100', 'SH', 'Tc-99m', SEQID=vNestSeq
oImg->SetValue, '0008,0102', 'SH', '99SDM', SEQID=vNestSeq

; Print a range including the new tags to
; the Output Log window.
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0054,0000', STOP_TAG='0056,0000')

; Format the output.
PRINT, FORMAT= $
    '(%%-12s, %3s, %5s, %31s, %10s)', $
    'TAG', 'VR', 'SEQID', $
    'DESCRIPTION', 'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

    ; If the item is nested within another item, indicate the
    ; level using > symbol.
    IF (vTags[xx].Level GT 0) THEN BEGIN
        vLvl = STRJOIN(REPLICATE('>',vTags[xx].Level))
        vtg = vLvl + vTags[xx].Tag
    ENDIF ELSE BEGIN
        vtg = vTags[xx].Tag
    ENDELSE

```

```

; If the tags are in a group, indicate this.
IF (vTags[xx].GroupNum GT 0) THEN BEGIN
    PRINT, FORMAT='(%" %15s, %1d)", 'Group', vTags[xx].GroupNum
ENDIF

; Print the fields of the structure.
PRINT, FORMAT = $
'(%"-12s, %3s, %5d, %31s, %10s)", $
vtg, vTags[xx].VR, vTags[xx].SeqId, $
vTags[xx].Description, vTags[xx].Value

ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

END

```

Running this example generates the following output.

TAG		VR	SEQID	DESCRIPTION	VALUE
0054,0016	,	SQ	337	Radiopharmaceutical Information,	
>0018,1071	,	DS	338	Radiopharmaceutical Volume,	0
>0054,0300	,	SQ	338	Radionuclide Code Sequence,	
>>0008,0100	,	SH	339	Code Value,	Tc-99m
>>0008,0102	,	SH	339	Coding Scheme Designator,	99SDM

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::ChangeTransferSyntax

[Syntax](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The `IDLffDicomEx::ChangeTransferSyntax` procedure method changes the transfer syntax of the `IDLffDicomEx` object and its associated pixel data. This allows you to change the compression setting of the pixel data and ensure that the transfer syntax value and pixel data are synchronized. Directly changing the transfer syntax (0002,0010) is not advised.

Note

There is no support for the JPEG compression algorithms on Macintosh. See “[File Compression and Transfer Syntax Support](#)” on page 68 for information on supported transfer syntaxes

Note

Attempting to change an existing file from a lossy JPEG format to another format will fail. This is prohibited to ensure that a file saved in a lossy format is always known to be less than the original data.

Note

When this method successfully completes the equivalent of an [IDLffDicomEx::Commit](#) call will have occurred (the file is saved to disk) to ensure the pixel data and the transfer syntax are synchronized. This means any sequence identifiers for the object are invalid and must be re-accessed using the [IDLffDicomEx::GetPrivateValue](#) or [IDLffDicomEx::GetValue](#) method.

There are five scenarios in which this method may be used. The following lists the actions of the `ChangeTransferSyntax` method in each case:

1. The original syntax is uncompressed and the new syntax is uncompressed. After calling this method, the pixel data remains unchanged, but the transfer syntax is changed.
2. The original syntax is compressed (lossless only) and the new syntax is uncompressed. After calling this method, the pixel data is retrieved and uncompressed, the transfer syntax is changed, the pixel data is written back into the image in the uncompressed format.
3. The original syntax is uncompressed and the new syntax is compressed (lossless or lossy). After calling this method, the pixel data is retrieved from the image, the transfer syntax is changed, the pixel data is written back into the image in the compressed format.

4. The original syntax is compressed (lossless only) and the new syntax is compressed (lossless or lossy). After calling this method, the pixel data is uncompressed, the transfer syntax is changed, the pixel data is written back into the image in the compressed format.
5. The original transfer syntax is the same as the new syntax. Calling this method saves the file.

Bit Depth Versus Image Compression

The following table provides information on the types of JPEG compression support for images with various bit depths. Not all JPEG formats can be used on all image types. Refer to *Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding* for additional details.

JPEG Format	8 bit	12 bit	16 bit
JPEG Baseline (lossy)	Yes	No	No
JPEG Extended (Process 2 & 4)(lossy)	Yes	Yes	No
JPEG Lossless, Non-Hierarchical	Yes	Yes	Yes
JPEG 2000 Image Compression (Lossless Only)	Yes	Yes	Yes
JPEG 2000 Image Compression	Yes	Yes	Yes

Table 3-6: JPEG Compression Support for Images of Varying Bit Depths

Syntax

Obj->[IDLffDicomEx::]ChangeTransferSyntax, *NewSyntaxUID* [, /LOSSY]

Arguments

NewSyntaxUID

A string that specifies the new transfer syntax for the file. This argument must be one of the values listed in the following table:

Argument Value	Transfer Syntax Name
1.2.840.10008.1.2	Implicit VR Little Endian
1.2.840.10008.1.2.1	Explicit VR Little Endian
1.2.840.10008.1.2.2	Explicit VR Big Endian
1.2.840.10008.1.2.4.50	JPEG Baseline (lossy)
1.2.840.10008.1.2.4.51	JPEG Extended (Process 2 & 4)(lossy)
1.2.840.10008.1.2.4.70	JPEG Lossless, Non-Hierarchical
1.2.840.10008.1.2.4.90	JPEG 2000 Image Compression (Lossless Only)
1.2.840.10008.1.2.4.91	JPEG 2000 Image Compression

Keywords

LOSSY

Set this keyword to control how tags that can indicate lossy compression are updated. The default behavior when the *NewSyntaxUID* argument is set to a lossy transfer syntax is to update the two tags as indicated in the following table. This occurs when the LOSSY keyword is *not* set. If this keyword *is* set the indicated tags remain unchanged. See *Digital Imaging and Communications in Medicine (DICOM) - Part*

3, Section C.7.6.1.1.5 for additional details on what other tags you can update when the compression format is lossy.

DICOM Attribute	Indication of Lossy Compression
Image Type (0008,0008)	The first value in this multi-value tag is updated to read as 'DERIVED'. Note - If the Image Type tag is not present it is added.
Lossy Compression (0028,2110)	This tag is updated to read '01' indicating the image has undergone lossy compression. This value should never be changed once set to 01.

Example

The following example changes the file compression of a selected file to a lossy format. Use the `BITS_STORED` property to query the bit depth of the image as not all images support all types of compression. Do not set the `LOSSY` keyword so the Image Type attribute is modified to state that the image is derived. Following compression, the original and compressed images are shown in a window.

Note

This example is not designed for images with more than a single sample per pixel (e.g. RGB images).

Note

To avoid an error, you must delete the `aImgClone.dcm` file prior to running this example more than a single time. The `ChangeTransferSyntax` method internally calls the [IDLffDicomEx::Commit](#) method and writes the file to disk.

```
PRO dicom_changecompression_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH(' ', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
; Set the NON_CONFORMING keyword to be able to add a public SQ
; of radiopharmaceutical items to any file.
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
```

```

        CLONE=sfile, /NON_CONFORMING)

; Get the value of the Image Type attribute prior to
; changing the transfer syntax.
oImg->GetProperty, IMAGE_TYPE = vImgType, $
    ROWS=vRows, COLUMNS=vCols
PRINT, 'Image Type Property = ', vImgType

; Check to see if the image has multiple frames.
frameTest = oImg->QueryValue('0028,0008')
IF FrameTest EQ 2 THEN BEGIN
    oImg->GetProperty, NUMBER_OF_FRAMES=frame
    frame = frame - 1
ENDIF ELSE BEGIN
    frame = 0
ENDELSE
order = 0

; Get the current transfer syntax.
oImg->GetProperty, TRANSFER_SYNTAX = vSyntax, $
    BITS_STORED = vBits
PRINT, 'Old Syntax ', vSyntax

; Get the pixel data before compression.
vPixOrig = oImg->GetPixelData(ORDER=vOrder, COUNT=vCnt)

; Change the compression of the file to a lossy type based on
; bit-depth of the image. Note that internally, the
; ChangeTransferSyntax calls commit and writes file to disk.
If vBits EQ 8 THEN $
    oImg->ChangeTransferSyntax, '1.2.840.10008.1.2.4.50'

IF vBits NE 8 THEN BEGIN
    ; If vBits not equal to 8 then compress the file using
    ; JPEG 2000 lossy compression.
    oImg->ChangeTransferSyntax, '1.2.840.10008.1.2.4.91'
ENDIF

oImg->GetProperty, TRANSFER_SYNTAX = vSyntax, $
    IMAGE_TYPE = vImgType

PRINT, 'New Syntax ', vSyntax
PRINT, 'New Image Type Property = ', vImgType

; Retrieve the compress pixel data.
vPixLossy = oImg->GetPixelData()

; Display the original and lossy compressed data.
WINDOW, XSIZE = vCols*2, YSIZE = vRows, $

```



```
TITLE = "Original and Compressed Frames"
FOR i = 1, frame+1 DO BEGIN
  TVSCL, vPixOrig[*,*,i-1], 0, ORDER = order
  TVSCL, vPixLossy[*,*,i-1], 1, ORDER = order
  WAIT, 1
ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

END
```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::Cleanup

[Syntax](#) | [Arguments](#) | [Keywords](#) | [Version History](#)

The IDLffDicomEx::Cleanup procedure method performs all cleanup on the object.

Note

Cleanup methods are special *lifecycle methods*, and as such cannot be called outside the context of object creation and destruction. In most cases, you cannot call the Cleanup method directly. However, one exception to this rule does exist. If you write your own subclass of this class, you can call the Cleanup method from within the Init or Cleanup method of the subclass.

Syntax

OBJ_DESTROY, *Obj*

or

Obj->[IDLffDicomEx::]Cleanup *(In a lifecycle method only.)*

Arguments

None

Keywords

None

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::Commit

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Version History](#)

The IDLffDicomEx::Commit procedure method writes changes to the underlying DICOM file. The in-memory copy of the DICOM file is freed when Commit is called. After a call to Commit any outstanding sequence identifiers for this object will be invalid. You must use [IDLffDicomEx::GetPrivateValue](#) (for private sequences) or [IDLffDicomEx::GetValue](#) (for public sequences) to re-access sequence identifiers prior to making additional modifications to sequence items.

Syntax

Obj->[\[IDLffDicomEx::\]Commit](#)

Return Value

None

Arguments

None

Keywords

None

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::CopyTags

[Syntax](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The `IDLffDicomEx::CopyTags` procedure method copies all the tags from the source object to the destination object beginning with the DICOM attribute tag specified by the *Start_Tag* and copying up to the *Stop_Tag*. This method does a deep copy of a tag, which means it copies all sub-items, even sequences that contain nested sequences and multiple repeating groups.

Note

This method is intended to copy small blocks of tags from one DICOM file to another DICOM file. This method is *not* intended to be used to copy entire DICOM files. To clone an existing DICOM file, use the `IDLffDicomEx::Init` method with the `CLONE` keyword set to copy an entire DICOM file.

Note

If you are copying non-standard tags to a destination object (as defined by its SOP Class definition), open the destination object with the `IDLffDicomEx::Init` method `NON_CONFORMING` keyword to avoid errors.

Note

In rare instances the values of copied tags are changed when they are added to the destination file. If you are copying multiple private block code tags that are not numbered sequentially by 1, they will be numbered in this manner when they are copied into the destination file.

Specifying Start and Stop Tags

The *Start_Tag* and *Stop_Tag* don't have to be precise tags. For example, suppose you provide '0010,0000' as the *Start_Tag* or *Stop_Tag* argument. If the specified tag does not exist in the file, copying will start with the next element after that one or stop on the element right before that one. While the *Start_Tag* and *Stop_Tag* arguments can be loosely defined, the definitions must adhere to the following guidelines:

- The DICOM attributes specified for the start and stop tags must be root level tags. These tags can be sequence tags as long as they exist at the root level, but they cannot be tags contained inside a sequence.
- When copying private DICOM attributes (those with an odd group number) it is necessary to start at the beginning of a private block. An error will be issued if you attempt to copy tags from the middle of a private sequence or private group. For example 0055,0010 is a tag that starts a new private block of tags.

Attempting to copy tags beginning with 0055,1013, which exists inside a private block, would generate an error.

Note

If you are copying a block of tags that includes multiple private blocks, each block must be copied independently as shown in the following “Example” section.

- The DICOM attributes specified for the start and stop tags cannot be set to ‘0000,0000’.

Note

Tags 0002,0003 (Media Storage SOP Instance tag) and 0008,0018 (SOP Instance tag) are not copied from one file to another. This avoids overwriting the unique instance identifiers for a file and prevents identical identifiers from existing in two unique files.

Note

Use the [IDLffDicomEx::EnumerateTags](#) method to view all attributes in a DICOM file.

Syntax

Obj->[IDLffDicomEx::]CopyTags, *DestinationObject*, *Start_Tag*, *Stop_Tag*

Arguments

DestinationObject

An IDLffDicomEx object reference to the file to which the specified tags will be copied.

Start_Tag

A string identifying a DICOM attribute in the form 'xxxx,xxxx' that specifies the first tag to be copied. A *Start_Tag* value of '0000,0000' is not valid. See [Specifying Start and Stop Tags](#) for more information. See “DICOM Attributes” on page 300 for a list of tags.

Stop_Tag

A string identifying a DICOM attribute in the form 'XXXX,XXXX' that specifies the last tag to be copied. A *Stop_Tag* value of '0000,0000' is not valid. See [Specifying Start and Stop Tags](#) for more information. See “[DICOM Attributes](#)” on page 300 for a list of tags.

Keywords

None

Example

The following example adds a number of private tags to a clone of the first selected image and then copies these blocks of private tags to a clone of the second selected file using the `CopyTags` method. The new tags and copied tags are displayed in the Output Log window.

Note

If you are copying a block of tags that include multiple private blocks, each block must be copied independently as shown in the following example.

Note

To avoid errors encountered when attempting to overwrite an existing file, neither cloned image is saved to disk. To do so, call the [IDLffDicomEx::Commit](#) method.

```

PRO print_tags_doc, vTags, vTagCnt

; Format the output.
PRINT, FORMAT= $
  '(%" %3s, %2s, %-12s, %3s, %5s, %12s, %15s")', $
  'IDX', 'LVL', 'TAG', 'VR', 'SEQID', $
  'DESCRIPTION', 'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

  ; If the item is nested within another item, indicate the
  ; level using > symbol.
  IF (vTags[xx].Level GT 0) THEN BEGIN
    vLvl = STRJOIN(REPLICATE('>',vTags[xx].Level))
    vtg = vLvl + vTags[xx].Tag
  ENDIF ELSE BEGIN
    vtg = vTags[xx].Tag
  ENDELSE

```

```

; If the tags are in a group, indicate this.
IF (vTags[xx].GroupNum GT 0) THEN BEGIN
    PRINT, FORMAT='(%"15s, %1d)", 'Group', vTags[xx].GroupNum
ENDIF

; Print the fields of the structure.
PRINT, FORMAT = $
    ('%"3d, %2d, %-12s, %3s, %5s, %12s, %15s")', $
    xx, vTags[xx].Level, vtg, vTags[xx].VR, $
    vTags[xx].SeqId, vTags[xx].Description, $
    vTags[xx].Value

ENDFOR

END

PRO dicom_tagcopyexample_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH(' ', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
; Set the NON_CONFORMING keyword to be able to add a public SQ
; of radiopharmaceutical items to any file.
oCloneImg= OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile, /NON_CONFORMING)

; Add private tags. The following are hypothetical.
; Create a multi-valued tag at the root level.
arr = [1, 2, 3, 4]
oCloneImg->SetPrivateValue, 'Private Test', '0053', '00', 'SS', $
    arr

; Create a sequence at the root level.
vSeqId = oCloneImg->AddPrivateSequence('VOI Min,Max', '0055', $
    '12')

; Add items to the sequence, specifying SQ identifier returned by
; AddPrivateSequence.
oCloneImg->SetPrivateValue, 'VOI Min,Max', '0055', '13', 'IS', $
    '215', SEQID=vSeqID
oCloneImg->SetPrivateValue, 'VOI Min,Max', '0055', '14', 'IS', $
    '234', SEQID=vSeqID

; Print a range including the new tags to

```

```

; the Output Log window.
vTags = oCloneImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0053,0000', STOP_TAG='0057,0000')
print_tags_doc, vTags, vTagCnt

; Open a second file and copy tags to this file.
; Select a DICOM file.
sFile1 = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
; Set the NON_CONFORMING keyword to be able to add a public SQ
; of radiopharmaceutical items to any file.
oDestImg = OBJ_NEW('IDLffDicomEx', path + 'aDestImg.dcm', $
    CLONE=sfile1, /NON_CONFORMING)

; Copy the private tags to the second file. Each block is copied
; independently.
oCloneImg->CopyTags, oDestImg, '0053,0000', '0055,0000'
oCloneImg->CopyTags, oDestImg, '0055,0000', '0057,0000'

; Print a range including the new tags to
; the Output Log window.
PRINT, 'Tags copied to aDestImg file.'
vTags = oDestImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0053,0000', STOP_TAG='0057,0000')
print_tags_doc, vTags, vTagCnt

; Clean up references.
OBJ_DESTROY, [oCloneImg, oDestImg]

END

```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::EnumerateTags

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::EnumerateTags function method returns an array of structures representing the contents of the DICOM file. Each structure contains fields relating to values within the DICOM attribute. This method allows you to access the contents of the DICOM file (specified by start and end tags) and output them to the IDL Workbench Output Log window or to a file.

Note

If the START_TAG and STOP_TAG keywords are not specified, then the return array contains a structure for every DICOM attribute in the file. See the following “Examples” section for sample code.

Syntax

```
Result = Obj->[IDLffDicomEx::]EnumerateTags ( [, START_TAG=string]
      [, STOP_TAG=string] [, COUNT=variable] [, FILENAME=string] [, /QUIET] )
```

Return Value

Returns an array of structures containing the indicated tag values. The array contains a structure for each tag enumerated by this method. Each structure has the following fields:

Field	IDL Data Type	Description
TAG	String	A nine character string containing the DICOM attribute tag (for example, ‘0080,0060’). This field always has a valid value. See “DICOM Attributes” on page 300 for a list of tags.
DESCRIPTION	String	A description of the public tag governed by the DICOM standard. This field is not available for private tags.

Table 3-7: DICOM Tag Structure Fields

Field	IDL Data Type	Description
VR	String	A two character string indicating the value representation of the attribute (for example, 'LO'). This field always has a valid value. See “Value Representations” on page 372 for more information.
LENGTH	ULong	An unsigned long value indicating the length of the value field of the DICOM attribute in bytes. If the VR field is SQ (a sequence), then the LENGTH field indicates the number of repeating groups in a sequence.
VALUECOUNT	Long	A long integer indicating the number of values in the value field of the DICOM attribute. If the attribute is multi-valued, then the individual values in the VALUE field are separated by a backslash '\
SEQID	Long	A long integer containing the sequence identifier of the DICOM attribute. This field contains a non-zero value even when the attribute is not a sequence, so that the value contained in this field can be used without error with any IDLffDicomEx method that has a SEQID keyword. Root level tags have identical valid values. Note - All sequence identifiers are invalidated when you call the IDLffDicomEx::Commit method. You must use IDLffDicomEx::GetValue to re-access sequence identifiers if needed.
GROUPNUM	Long	A long integer containing the group number for a tag that is in a repeating group. This value can be used when formatting output to display repeating groups of tags. This value equals 0 for tags not in a repeating group.

Table 3-7: DICOM Tag Structure Fields (Continued)

Field	IDL Data Type	Description
LEVEL	Long	A long integer indicating the nesting level of an attribute. A value of 0 indicates the tag is at the root level. A value greater than one indicates the tag is not at the root level. This value can be used to indent tags so there is a visual indication of tags inside sequences or nested sequences.
VALUE	String	A string containing the value of the DICOM attribute tag with the following caveats: <ul style="list-style-type: none"> • For tags with multiple values, the values are separated by a backslash character (“\”). • When the VR field is OB, OW, or OF the value field is not filled in as the tag contains binary data that is not suitable for presentation in a string. Typically the OB and OW tags are used for pixel data. • When the VR is SQ then value is not filled in as the value field for a sequence does not contain data. The sequence identifiers for the tags in the sequence are returned in the SEQID field.

Table 3-7: DICOM Tag Structure Fields (Continued)

Arguments

None

Keywords

START_TAG

Set this keyword to a string that identifies the first a DICOM attribute to be enumerated. The START_TAG has the format of 'XXXX, XXXX' indicating the group and element of the attribute. A START_TAG value of '0000, 0000' is valid. See [“DICOM Attributes”](#) on page 300 for a list of tags.

Note

The DICOM attributes specified for the start and stop tags must be root level tags. These tags can be sequence tags as long as they exist at the root level, but they cannot be tags contained inside a sequence.

STOP_TAG

Set this keyword to a string that identifies the final DICOM attribute to be enumerated. STOP_TAG must have the format of 'XXXX,XXXX' indicating the group and element of the attribute. A STOP_TAG value of '0000,0000' is valid. See “[DICOM Attributes](#)” on page 300 for a list of tags.

Note

The DICOM attributes specified for the start and stop tags must be root level tags. These tags can be sequence tags as long as they exist at the root level, but they cannot be tags contained inside a sequence.

COUNT

Set this keyword to a named variable that will contain a long integer indicating the number of structures in the array returned by this method. This equals the number of DICOM attributes for which values are enumerated.

FILENAME

Set this keyword to a string specifying name of the file to which the enumerated tags are to be written. This can either be an absolute path ('C:\myDicomTags.txt') or simply a filename ('myDicomTags.txt'). When only a filename is provided, the file is saved in the IDL working directory.

QUIET

Set this keyword to suppress the following message in the Output Log window:

```
"Warning: Skipping tag, unsupported VR type (tag/vr)"
```

This message is displayed when the DICOM file contains a DICOM attribute that has a VR type of [UN](#) (unknown). This can happen when a vendor adds a private tag using the UN value representation. See “[Value Representations](#)” on page 372 for more information.

Example

The following code prints all tags in the selected file to the Output Log window and to a file (`dicomtags.txt`) in your working directory. Set the Output Log window to a monospaced font such as Courier to display properly aligned columns.

```

PRO read_dicomtags_doc

; Select a DICOM file to examine.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm')

; Open the selected file in read-only mode.
oImg = OBJ_NEW('IDLffDicomEx', sfile)

; Use the EnumerateTags method to access the values of
; each DICOM attribute. Do not specify start or stop
; tag values to return all tags. Write the tags to a file
; in the IDL working directory.
vTags = oImg->EnumerateTags(COUNT = vTagCnt, $
    FILENAME = 'dicomtags.txt')

; Print the tags to the Output Log window:

; Identify the name of the DICOM file and number of tags.
oImg->GetProperty, FILENAME = vfilename
PRINT, ' Tags in = ', vfilename, ' tag count = ', vTagCnt

; Format the output.
PRINT, FORMAT= $
    '(%"%3s, %2s, %12s, %3s, %7s, %3s, %5s, %30s, %50s")', $
    'IDX', 'LVL', 'TAG', 'VR', 'LEN', 'CNT', 'SEQID', $
    'DESCRIPTION', 'VALUE'

; Cycle through all of the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

    ; If the item is nested within another item, indicate the
    ; level using > symbol.
    IF (vTags[xx].Level GT 0) THEN BEGIN
        vLvl = STRJOIN(REPLICATE('>', vTags[xx].Level))
        vtg = vLvl + vTags[xx].Tag
    ENDIF ELSE BEGIN
        vtg = vTags[xx].Tag
    ENDELSE

    ; If the tags are in a group, indicate this.

```

```

IF (vTags[xx].GroupNum GT 0) THEN BEGIN
    PRINT, FORMAT='(%%15s, %1d)', 'Group', vTags[xx].GroupNum
ENDIF

; Print the fields of the structure.
PRINT, FORMAT= $
    '(%3d, %2d, %12s, %3s, %7d, %3d, %5d, %30s, %50s)', $
    xx, vTags[xx].Level, vtg, vTags[xx].VR, vTags[xx].Length, $
    vTags[xx].ValueCount, vTags[xx].SeqId, $
    vTags[xx].Description, vTags[xx].Value

ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

END

```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetDescription

[Syntax](#) | [Return Value](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::GetDescription function method returns the description associated with a public DICOM attribute specified by a standard DICOM attribute tag.

Syntax

```
Result = Obj->[IDLffDicomEx::]GetDescription(DicomTag )
```

Return Value

Returns a string containing the attribute description.

Arguments

DicomTag

A string that identifies the group and element of a DICOM attribute in the form 'XXXX, XXXX'. The *DicomTag* argument must reference a public tag. See “[DICOM Attributes](#)” on page 300 for a list of tags.

Keywords

None

Example

The following example returns the description of Photometric Interpretation attribute from a DICOM file in the `examples\data` directory. See [PHOTOMETRIC_INTERPRETATION](#) for more information on this attribute.

```
PRO read_attrdescription_doc

; Select a DICOM file to examine.
sfile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Open the selected file in read-only mode.
oImg = OBJ_NEW('IDLffDicomEx', sfile)
```

```
; Return the photometric interpretation description.  
result = oImg->GetDescription('0028,0004')  
  
Print, 'Result is ', result  
  
END
```

For the `mr_knee.dcm` file, the following appears in the Output Log window:

```
Result is Photometric Interpretation
```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetPixelData

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Examples](#) | [Version History](#)

The IDLffDicomEx::GetPixelData function method returns the uncompressed pixel data from the DICOM image file. (If pixel data is stored in a compressed format, it is uncompressed before it is returned.) A DICOM file may store a single-frame image or a multiple-frame image. In the case of a multi-frame image, this method allows you to return the pixel data of all of the frames, or of a specified frame when you set the FRAME keyword. The NUMBER_OF_FRAMES property can be used to determine whether the image contains single or multiple frames. If the Number of Frames attribute does not exist in the DICOM image file, then it contains a single-frame image.

Warning

By default, GetPixelData returns the pixel data array in standard IDL format where the first pixel in the returned array is the bottom left-hand pixel in the frame. You must set the ORDER keyword to return the array DICOM format where the first pixel in the returned array is the top left-hand pixel in the frame.

Note

If you are not sure that the image contains multiple frames, use [IDLffDicomEx::QueryValue](#) to check for Number of Frames attribute before attempting to access the value. Not all DICOM SOP classes support multi-frame pixel data. Attempting to return a property value associated with a nonexistent attribute or an attribute that does not have a value will result in an error.

Tip

Use the following settings when displaying planar pixel data (where the PLANAR_CONFIGURATION property value equals 1): set the TVSCL method TRUE keyword to 3, or set the IDLgrImage object INTERLEAVE property to 2.

When accessing pixel data, the following tags (also exposed as properties) are used in the construction of the array of pixel data:

DICOM Attribute	Description
BITS_ALLOCATED	Determines the width of the elements in the returned array. Typical values are 8 bits or 16 bits. If this tag is missing an error is issued.
SAMPLES_PER_PIXEL	Typical values are 1 for monochrome frames and 3 or 4 for RGB frames. If this tag is missing, an error is issued.
ROWS	Number of horizontal lines in a frame. If this tag is missing an error is issued
COLUMNS	Number of vertical lines in a frame. If this tag is missing an error is issued
PIXEL_REPRESENTATION	Determines how to return the data in the correct format for images with greater than 8 bit signed or unsigned data. The GetPixelData method will use a value of 0 if this tag is not present.
PLANAR_CONFIGURATION	Determines how the ORDER keyword operates on the pixel data. This tag is required for non-monochrome images. The GetPixelData method will use a value of 0 if this tag is not present.
NUMBER_OF_FRAMES	Determines the frames component of a multi-frame image array and is required for multi-frame images. This tag is only allowed in SOP Classes that support multi-frame images. The GetPixelData method will use a value of 1 if this tag is not present.

Table 3-8: DICOM Attributes Queried to Determine the Pixel Data Array

Syntax

```
Result = Obj->[IDLffDicomEx::]GetPixelData ( [, FRAME=integer] [, /ORDER]
[, COUNT=variable])
```

Return Value

Returns a multi-dimensional array. The data type of the array is based upon the `BITS_ALLOCATED` property of the DICOM file as follows:

- Byte — the image data is 8 bits and signed or unsigned
- Unsigned integer — the image data is greater than 8 bits and unsigned
- Integer — the image data is greater than 8 bits and signed

The following table describes the possible arrangements of the multi-dimensional array.

Dimensions	Arrangement	Description
Two-dimensional	[columns, rows]	A single monochrome frame.
Three-dimensional	[columns, rows, frames]	Two or more monochrome frames.
	[3, columns, rows]	A single RGB or HSV pixel interleaved frame.
	[columns, rows, 3]	A single RGB or HSV planar interleaved frame.
	[4, columns, rows]	A single CMYK pixel interleaved frame.
	[columns, rows, 4]	A single CMYK planar interleaved frame.
Four-dimensional	[3, columns, rows, frames]	Two or more RGB or HSV pixel interleaved frames.
	[columns, rows, 3, frames]	Two or more RGB or HSV planar interleaved frames.
	[4, columns, rows, frames]	Two or more CMYK pixel interleaved frames.
	[columns, rows, 4, frames]	Two or more CMYK planar interleaved frame.

Table 3-9: Pixel Data Array Possibilities

Arguments

None

Keywords

FRAME

Set this keyword to a long integer to specify which frame or pixel data within a multi-frame image is to be returned. Allowable values denoted the zero-based index value of the frame, from 0 to [NUMBER_OF_FRAMES](#) -1. If not specified, the pixel data of all frames is returned.

ORDER

Set the keyword to return the pixel data in DICOM format where the first pixel in the returned array is the top left-hand pixel in the frame. If this keyword is not set, the pixel data array is returned in standard IDL format where the first pixel in the returned array is the bottom left-hand pixel in the frame.

COUNT

Set this keyword to a named variable that will contain a long integer indicating the number frames returned in the pixel data array.

Examples

Filtering Monochrome DICOM Data

The following example applies the ROBERTS edge-detection filter to every frame within a single- or multiple-frame monochrome DICOM file. Each frame is then sequentially displayed in a Direct Graphics widow.

Note

For an example that writes RGB pixel data to an `IDLffDicomEx` object, see the “Example” section of [IDLffDicomEx::SetPixelData](#).

Example Code

The code for `filter_clonedicom_doc.pro` is provided in the IDL distribution, in the `examples/doc/dicom` subdirectory of the main IDL directory. Run the example procedure by entering `filter_clonedicom` at the IDL command prompt

or view the file in an IDL Editor window by entering `.EDIT filter_clonedicom.pro`.

```

PRO filter_clonedicom_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('',SUBDIRECTORY=['examples','data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile)

; Get image attributes.
oImg->GetProperty, BITS_ALLOCATED = vBA, ROWS=rows, $
    COLUMNS=cols, SAMPLES_PER_PIXEL=samples

; Allow user to select monochrome image.
IF samples gt 1 THEN BEGIN
    v= DIALOG_MESSAGE('This application requires ' + $
        'a monochrome image.', /ERROR)
    sFile = DIALOG_PICKFILE( $
        PATH=FILEPATH('',SUBDIRECTORY=['examples','data']), $
        TITLE='Select DICOM Patient File', FILTER='*.dcm', $
        GET_PATH=path)
    ; Create a clone (aImgClone.dcm) of the selected file (sfile).
    oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
        CLONE=sfile)
ENDIF

; Check to see if the image has multiple frames.
; First check for the presence of the Number of Frames tag.
FrameTest = oImg->QueryValue('NUMBER_OF_FRAMES')

; If the tag exists and has a value, retrieve it. Pixel data
; FRAME index is zero-based so subtract 1 from the value.
; ORDER is set for IDL consistency.
IF FrameTest EQ 2 THEN BEGIN
    oImg->GetProperty, NUMBER_OF_FRAMES=frame
    FRAME = frame - 1
; Otherwise, set FRAME to 0 indicating is is a single frame
; image. ORDER is set for IDL consistency.
ENDIF ELSE BEGIN
    FRAME = 0
ENDELSE
ORDER = 0

```

```

; Return all of the frames of pixel data by
; not specifying a value for FRAME.
vPixels = oImg->GetPixelData(ORDER=order, COUNT=cnt)
PRINT, 'Returned pixel data for number of frames = ', cnt

; Initialize and array of the proper type for the
; filtered pixel data.
IF vBA GT 8 THEN BEGIN
    vFilterArr = INTARR([rows,cols,frame+1])
ENDIF ELSE BEGIN
    vFilterArr = BYTARR([rows,cols,frame+1])
ENDELSE

; Filter each frame of data or the single frame.
IF frame GT 0 THEN BEGIN
    FOR n = 1, frame+1 DO BEGIN
        vFilterPixels = ROBERTS(vPixels[*,*,n-1])
        vFilterArr[*,*,n-1] = vFilterPixels
    ENDFOR
ENDIF ELSE BEGIN
    vFilterArr = ROBERTS(vPixels)
ENDELSE

; Roberts function changes byte data to integer.
; SetPixelData requires array of original type.
; If original array was byte (as indicated by
; BITS_ALLOCATED = 8), change the array back to byte.
IF vBA EQ 8 THEN BEGIN
    vFilterArr = BYTE(vFilterArr)
ENDIF

; Set the pixel data of the frame(s) back to the image.
oImg->SetPixelData, vFilterarr, ORDER=order

; Write the pixel data changes to the file.
oImg->Commit

; Sequentially display each frame of the original
; and filtered data.
WINDOW, XSIZE=cols*2, YSIZE=rows, $
    TITLE = 'Original and Filtered Frames'
FOR i = 1, frame+1 DO BEGIN
    TVSCL, vPixels[*,*,i-1], 0, ORDER = order
    TVSCL, vfilterarr[*,*,i-1], 1, ORDER = order
    WAIT, 1
ENDFOR

; Clean up references.

```

```
OBJ_DESTROY, oImg

; Note: the following line allows you to run the program
; multiple times without having to manually delete the file.
; You cannot duplicate an existing file when creating or cloning
; a DICOM file.
FILE_DELETE, path + 'aImgClone.dcm', /ALLOW_NONEXISTENT

END
```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetPrivateValue

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::GetPrivateValue function method returns the value of a private DICOM attribute using a private code defined by the author of the private tag, a group number and part of the element tag instead of a standard DICOM attribute tag.

Note

GetPrivateValue will fail if you attempt to return a value for an attribute that does not exist, an attribute that does not have a value, or an attribute that has been removed. If you are not sure an attribute exists or has a value use [IDLffDicomEx::QueryPrivateValue](#) before calling GetPrivateValue.

Note

In the majority of cases, [IDLffDicomEx::GetValue](#) can be used to read a private tag.

Syntax

*Result = Obj->[IDLffDicomEx::]GetPrivateValue(PrivateCode, Group, Element
[, SEQID=integer] [, COUNT=variable])*

Return Value

Returns one of the following:

- A scalar value for a private attribute with one value.
- A vector of scalar values for private attributes with multiple values.
- A long integer if the private attribute is a sequence. This value is used as the SEQID keyword in subsequent calls to GetPrivateValue to access items contained in the sequence.
- A vector of values when the sequence contains groups (set of repeating tags within the sequence). See the [IDLffDicomEx::GetPrivateValueLength](#) method “Example” section for code that uses such an array.

Arguments

PrivateCode

A string identification code that identifies the private block of data. Within a given private group *PrivateCode* labels are stored sequentially in the element addresses ranging from '0010' to '00FF'. For example, the string value stored at DICOM tag address '0029,0010' is the *PrivateCode* for the block of data tagged at '0029,1000' - '0029,10FF'. The label stored at '0029,0011' would be the *PrivateCode* for the data in tags '0029,1100' - '0029,11 FF'.

Group

A string identifying the group tag number of the private attribute (the first four digits of a DICOM tag). This must be an odd number and in the form 'XXXX'.

Element

A string identifying the last two digits of the element associated with the private attribute. This must be in the form 'XX'. Valid values are 10 - FF.

Note

The first two digits of the *Element* are implicit in the *PrivateCode* argument.

Keywords

SEQID

Set this keyword only if retrieving the value of a private attribute that exists within a sequence. Use this keyword to specify sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the `GetPrivateValue` method.
- Set to 0 or do not specify this keyword to indicate the private attribute exists at the root level of the DICOM file. This is the default.

COUNT

Set this keyword equal to a named variable that will contain an unsigned long value indicating the number of elements in this method's return value. Possible values are:

- 1 indicating the return value is a scalar value.

- n where n is the number of elements in the returned array. This corresponds to the number of values in the multi-valued attribute.

Example

The following example uses the results of the `IDLffDicomEx::GetPrivateValue` method COUNT keyword to cycle through a multi-valued private attribute that has been added to a file. To avoid errors arising from attempting to write to an existing file, the cloned image is not saved to the database. To save the changes, call the `IDLffDicomEx::Commit` method.

```

PRO dicom_getprivate_value_count_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile)

; Add private tags. The following are hypothetical.
; Create a multi-valued tag at the root level.
arr = [11, 12, 13, 14]
oImg->SetPrivateValue, 'Private Test', '0053', '10', 'SS', arr

; Create a sequence at the root level.
vSeqID = oImg->AddPrivateSequence('VOI Min,Max', '0055', '12')

; Add items to the sequence, specifying SQ identifier returned by
; AddPrivateSequence.
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '13', 'IS', '215', $
    SEQID=vSeqID
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '14', 'IS', '234', $
    SEQID=vSeqID

; Get the value of a multi-valued root-level private attribute.
; Get the number of items in the multi-valued attribute using
; either the COUNT keyword to GetPrivateValue or
; GetPrivateValueCount.
vValue = oImg->GetPrivateValue('Private Test', '0053', '10', $
    COUNT=vCount)

; Get the VR.
vVR = oImg->GetPrivateVR('Private Test', '0053', '10')

```

```
FOR i = 1, vCount DO BEGIN
    Print, 'Value number', i, + ' is ', vValue[i-1], + $
        ' and VR is ', vVR
ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

END
```

The following appears in the Output Log window.

```
Value number      1 is      11 and VR is SS
Value number      2 is      12 and VR is SS
Value number      3 is      13 and VR is SS
Value number      4 is      14 and VR is SS
```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetPrivateValueCount

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::GetPrivateValueCount function method returns the number of values contained in the value field of a private DICOM attribute. This method uses a private code defined by the author of the private tag, a group number, and part of the element tag instead of a standard DICOM attribute tag to identify the DICOM attribute.

Note

GetPrivateValueCount will fail if you attempt to return a value for an attribute that does not exist or an attribute that has been removed. If you are not sure if an attribute exists use [IDLffDicomEx::QueryPrivateValue](#) before calling GetPrivateValueCount.

Syntax

```
Result = Obj->[IDLffDicomEx::]GetPrivateValueCount(PrivateCode, Group,  
Element [, SEQID=integer] )
```

Return Value

Returns an unsigned long value indicating the number of values in the value field of the specified attribute as follows:

- A value of 0 indicates the tag had no value
- A value greater than 0 (*n*) indicates the number of values in the value field

Arguments

PrivateCode

A string identification code that identifies the private block of data. Within a given private group *PrivateCode* labels are stored sequentially in the element addresses ranging from '0010' to '00FF'. For example, the string value stored at DICOM tag address '0029,0010' is the *PrivateCode* for the block of data tagged at '0029,1000' - '0029,10FF'. The label stored at '0029,0011' would be the *PrivateCode* for the data in tags '0029,1100' - '0029,11 FF'.

Group

A string identifying the group tag number of the private attribute (the first four digits of a DICOM tag). This must be an odd number and in the form 'XXXX'.

Element

A string identifying the last two digits of the element associated with the private attribute. This must be in the form 'XX'. Valid values are 10 - FF.

Note

The first two digits of the *Element* are implicit in the *PrivateCode* argument.

Keywords

SEQID

Set this keyword only if the private attribute exists within a sequence. Use this keyword to specify sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the `GetPrivateValue` method.
- Set to 0 or do not specify this keyword to indicate the private attribute exists at the root level of the DICOM file. This is the default.

Example

The following example uses `GetPrivateValueCount` to cycle through a multi-valued private attribute that has been added to a file. To avoid errors arising from attempting to write to an existing file, the cloned image is not saved to the database. To save the changes, call the `IDLffDicomEx::Commit` method.

```
PRO dicom_getprivate_value_doc

; Select a DICOM file.
sfile = DIALOG_PICKFILE( $
    PATH=FILEPATH(' ', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile)
```

```

; Add private tags. The following are hypothetical.
; Create a multi-valued tag at the root level.
arr = [11, 12, 13, 14]
oImg->SetPrivateValue, 'Private Test', '0053', '10', 'SS', arr

; Create a sequence at the root level.
vSeqID = oImg->AddPrivateSequence('VOI Min,Max', '0055', '12')

; Add items to the sequence, specifying SQ identifier returned by
; AddPrivateSequence.
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '13', 'IS', '215', $
    SEQID=vSeqID
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '14', 'IS', '234', $
    SEQID=vSeqID

; Get the value of a multi-valued root-level private attribute.
vValue = oImg->GetPrivateValue('Private Test', '0053', '10')

; Get the VR.
vVR = oImg->GetPrivateVR('Private Test', '0053', '10')

; Get the number of items in the multi-valued attribute. Use either
; the COUNT keyword to GetPrivateValue or GetPrivateValueCount.
vCount = oImg->GetPrivateValueCount('Private Test', '0053', '10')

FOR i = 1, vCount DO BEGIN
    Print, 'Value number', i, + ' is ', vValue[i-1], + $
        ' and VR is ', vVR
ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

END

```

The following appears in the Output Log window.

```

Value number      1 is      11 and VR is SS
Value number      2 is      12 and VR is SS
Value number      3 is      13 and VR is SS
Value number      4 is      14 and VR is SS

```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetPrivateValueLength

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Examples](#) | [Version History](#)

The IDLffDicomEx::GetPrivateValueLength function method returns the length of all values or of a specified value (in bytes) in a private DICOM attribute. This method uses a private code defined by the author of the private tag, a group number, and part of the element tag instead of a standard DICOM attribute tag to identify the private DICOM attribute.

Note

GetPrivateValueLength will fail if you attempt to return a value for an attribute that does not exist or an attribute that has been removed. If you are not sure if an attribute exists use [IDLffDicomEx::QueryPrivateValue](#) before calling GetPrivateValueLength.

Syntax

```
Result = Obj->[IDLffDicomEx::]GetPrivateValueLength(PrivateCode, Group,
  Element [, SEQID=integer] [, VALUEINDEX=integer] )
```

Return Value

Returns a long integer indicating the length of one of the following:

- The length (in bytes) of all values when the VALUEINDEX keyword is not set
- The length (in bytes) of a single value specified by the VALUEINDEX keyword
- The number of repeating groups contained within a sequence if the *PrivateCode*, *Group* and *Element* arguments identify a sequence.

Arguments

PrivateCode

A string identification code that identifies the private block of data. Within a given private group *PrivateCode* labels are stored sequentially in the element addresses ranging from '0010' to '00FF'. For example, the string value stored at DICOM tag address '0029,0010' is the *PrivateCode* for the block of data tagged at '0029,1000' - '0029,10FF'. The label stored at '0029,0011' would be the *PrivateCode* for the data in tags '0029,1100' - '0029,11 FF'.

Group

A string identifying the group tag number of the private attribute (the first four digits of a DICOM tag). This must be an odd number and in the form 'XXXX'.

Element

A string identifying the last two digits of the element associated with the private attribute. This must be in the form 'XX'. Valid values are 10 - FF.

Note

The first two digits of the *Element* are implicit in the *PrivateCode* argument.

Keywords

SEQID

Set this keyword only if the private attribute exists within a sequence. Use this keyword to specify sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the `GetPrivateValue` method.
- Set to 0 or do not specify this keyword to indicate the private attribute exists at the root level of the DICOM file. This is the default.

VALUEINDEX

Set this keyword to an integer indicating the one-based index number of the value for which to return the length. If not set, this method returns the length of a single value for a single-valued attribute, or the length of all values for a multi-valued attribute.

Note

An error will be issued if you specify a value larger than the number of values in the private attribute.

Examples

The following example adds private tags to the clone of a selected DICOM file, and commits this file to memory. It then queries for a private sequence to make sure it exists and proceeds to use `GetPrivateValue` (to return a vector of sequence identifiers, one for each group) and `GetPrivateValueLength` (to return the number of repeating

groups) to access the length and value of a private attribute that is repeated within the sequence.

```

PRO print_tags_doc, vTags, vTagCnt

; Format the output.
PRINT, FORMAT= $
  '(%" %3s, %2s, %-12s, %3s, %5s, %12s, %15s")', $
  'IDX', 'LVL', 'TAG', 'VR', 'SEQID', $
  'DESCRIPTION', 'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

  ; If the item is nested within another item, indicate the
  ; level using > symbol.
  IF (vTags[xx].Level GT 0) THEN BEGIN
    vLvl = STRJOIN(REPLICATE('>',vTags[xx].Level))
    vtg = vLvl + vTags[xx].Tag
  ENDIF ELSE BEGIN
    vtg = vTags[xx].Tag
  ENDELSE

  ; If the tags are in a group, indicate this.
  IF (vTags[xx].GroupNum GT 0) THEN BEGIN
    PRINT, FORMAT='(%" %15s, %1d")', 'Group', vTags[xx].GroupNum
  ENDIF

  ; Print the fields of the structure.
  PRINT, FORMAT = $
    '(%" %3d, %2d, %-12s, %3s, %5s, %12s, %15s")', $
    xx, vTags[xx].Level, vtg, vTags[xx].VR, $
    vTags[xx].SeqId, vTags[xx].Description, $
    vTags[xx].Value
ENDFOR

END

PRO dicom_getprivate_length_doc

; Select a DICOM file.
sfile = DIALOG_PICKFILE( $
  PATH=FILEPATH('',SUBDIRECTORY=['examples','data']), $
  TITLE='Select DICOM Patient File', FILTER='*.dcm', $
  GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
  CLONE=sfile)

```

```

; Add two sets of repeating tags (groups) to a private sequence
; (0051, 0012), which is created by AddPrivateGroup. This sequence
; exists at the root-level of the DICOM file. Add two sets of
; repeating tags to the root sequence.
vGrp1 = oImg->AddPrivateGroup('Root Private SQ', '0051', '12')
oImg->SetPrivateValue, 'Root Private SQ', '0051', '14', 'ST', $
    'gr1Tag1', SEQID=vGrp1
oImg->SetPrivateValue, 'Root Private SQ', '0051', '15', 'ST', $
    'gr1Tag2', SEQID=vGrp1
vGrp2 = oImg->AddPrivateGroup('Root Private SQ', '0051', '12')
oImg->SetPrivateValue, 'Root Private SQ', '0051', '14', 'ST', $
    'gr2Tag1', SEQID=vGrp2
oImg->SetPrivateValue, 'Root Private SQ', '0051', '15', 'ST', $
    'gr2Tag2', SEQID=vGrp2

; Print a range including the new tags to
; the Output Log window.
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0051,0000', STOP_TAG='0057,0000')
print_tags_doc, vTags, vTagCnt

; Commit the changes.
oImg->Commit

; Make sure the private sequence exists.
vQuery = oImg->QueryPrivateValue('Root Private SQ', '0051', '12')
If vQuery NE 0 THEN BEGIN

; Retrieve the sequence identifier, lost after a commit. When the
; sequence contains multiple groups, this returns an zero based
; vector of sequence identifiers, one for each group.
vSeqId = oImg->GetPrivateValue('Root Private SQ', '0051', '12')

; Return the number of sets of repeating tags in the private
; sequence. This value is used to access a private value in
; each group.
vSeqLength = oImg->GetPrivateValueLength('Root Private SQ', $
    '0051', '12')

For i = 1, vSeqLength do begin
    ; Return the length and value of each private attribute.
    vLength = oImg->GetPrivateValueLength('Root Private SQ', $
        '0051', '14', SEQID=vSeqId[i-1])
    vResult = oImg->GetPrivateValue('Root Private SQ', $
        '0051', '14', SEQID=vSeqId[i-1])
    Print, 'Sequence group ', i, + '(0051,1014) length is ', $
        vLength, + ' and value is ', vResult
ENDFOR
ENDIF

```

```

; Clean up references.
OBJ_DESTROY, oImg

; Note: the following line allows you to run the project
; multiple times without having to manually delete the file.
; You cannot duplicate an existing file when creating or cloning
; a DICOM file.
FILE_DELETE, path + 'aImgClone.dcm', /ALLOW_NONEXISTENT

END

```

The following appears in the Output Log window.

```

IDX, LV, TAG          , VR, SEQID, DESCRIPTION,          VALUE
0,  0, 0051,0010     , LO,      ,              , Root Private SQ
1,  0, 0051,1012     , SQ,      ,              ,
   Group, 1
2,  1, >0051,0010    , LO,      ,              , Root Private SQ
3,  1, >0051,1014    , ST,      ,              ,      gr1Tag1
4,  1, >0051,1015    , ST,      ,              ,      gr1Tag2
   Group, 2
5,  1, >0051,0010    , LO,      ,              , Root Private SQ
6,  1, >0051,1014    , ST,      ,              ,      gr2Tag1
7,  1, >0051,1015    , ST,      ,              ,      gr2Tag2
Sequence group 1(0051,1014) length is 8 and value is gr1Tag1
Sequence group 2(0051,1014) length is 8 and value is gr2Tag1

```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetPrivateVR

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::GetPrivateVR function method returns the Value Representation (VR) of a private DICOM attribute. This method uses a private code defined by the author of the private tag, a group number, and part of the element tag instead of a standard DICOM attribute tag to identify the private DICOM attribute.

Note

GetPrivateVR will fail if you attempt to return a VR for an attribute that does not exist or an attribute that has been removed. If you are not sure an attribute exists use [IDLffDicomEx::QueryPrivateValue](#) before calling GetPrivateVR.

Syntax

```
Result = Obj->[IDLffDicomEx::]GetPrivateValueLength(PrivateCode, Group,
Element [, SEQID=integer] )
```

Return Value

Returns a string indicating the Value Representation of the attribute. See “[Value Representations](#)” on page 372 for details on each type of VR.

Arguments

PrivateCode

A string identification code that identifies the private block of data. Within a given private group *PrivateCode* labels are stored sequentially in the element addresses ranging from '0010' to '00FF'. For example, the string value stored at DICOM tag address '0029,0010' is the *PrivateCode* for the block of data tagged at '0029,1000' - '0029,10FF'. The label stored at '0029,0011' would be the *PrivateCode* for the data in tags '0029,1100' - '0029,11 FF'.

Group

A string identifying the group tag number of the private attribute (the first four digits of a DICOM tag). This must be an odd number and in the form 'XXXX'.

Element

A string identifying the last two digits of the element associated with the private attribute. This must be in the form 'XX'. Valid values are 10 - FF.

Note

The first two digits of the *Element* are implicit in the *PrivateCode* argument.

Keywords

SEQID

Set this keyword only if the private attribute exists within a sequence. Use this keyword to specify a sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the `GetPrivateValue` method.
- Set to 0 or do not specify this keyword to indicate the private attribute exists at the root level of the DICOM file. This is the default.

Example

The following example uses `GetPrivateValueCount` to cycle through a multi-valued private attribute that has been added to a file. The VR and value of each item is printed to the Output Log window.

Note

To avoid errors caused by trying to write to an existing file, the cloned image is not saved to the database. To save the changes, call the [IDLffDicomEx::Commit](#) method.

```
PRO dicom_getprivate_value_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH(' ', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile)
```

```

; Add private tags. The following are hypothetical.
; Create a multi-valued tag at the root level.
arr = [11, 12, 13, 14]
oImg->SetPrivateValue, 'Private Test', '0053', '10', 'SS', arr

; Create a sequence at the root level.
vSeqID = oImg->AddPrivateSequence('VOI Min,Max', '0055', '12')

; Add items to the sequence, specifying SQ identifier returned by
; AddPrivateSequence.
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '13', 'IS', '215', $
    SEQID=vSeqID
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '14', 'IS', '234', $
    SEQID=vSeqID

; Get the value of a multi-valued root-level private attribute.
vValue = oImg->GetPrivateValue('Private Test', '0053', '10')

; Get the VR.
vVR = oImg->GetPrivateVR('Private Test', '0053', '10')

; Get the number of items in the multi-valued attribute. Use either
; the COUNT keyword to GetPrivateValue or GetPrivateValueCount.
vCount = oImg->GetPrivateValueCount('Private Test', '0053', '10')

FOR i = 1, vCount DO BEGIN
    Print, 'Value number', i, + ' is ', vValue[i-1], + $
        ' and VR is ', vVR
ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

END

```

The following appears in the Output Log window.

```

Value number      1 is      11 and VR is SS
Value number      2 is      12 and VR is SS
Value number      3 is      13 and VR is SS
Value number      4 is      14 and VR is SS

```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetProperty

[Syntax](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::GetProperty procedure method retrieves the value of an IDLffDicomEx property.

Note

GetProperty will fail if you attempt to return a value for an attribute that does not exist, an attribute that does not have a value, or an attribute that has been removed. If you are not sure an attribute exists or has a value use [IDLffDicomEx::QueryValue](#) before calling GetProperty.

Syntax

Obj->[\[IDLffDicomEx::\]GetProperty](#)[, *PROPERTY=variable*]

Arguments

None

Keywords

Any property listed under “[IDLffDicomEx Properties](#)” on page 70 that contains the word “Yes” in the “Get” column of the properties table can be retrieved using this method. To retrieve the value of a property, specify the property name as a keyword set equal to a named variable that will contain the value of the property.

Example

See the [IDLffDicomEx::GetPixelData](#) method “Example” section for sample code that retrieves property values using GetProperty.

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetValue

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The `IDLffDicomEx::GetValue` function method returns the value of a DICOM attribute specified by a standard DICOM attribute tag. This method allows you to return values of public attributes. See [IDLffDicomEx::GetPrivateValue](#) for information on returning the values of private attributes.

Note

`GetValue` will fail if you attempt to return a value for an attribute that does not exist, an attribute that does not have a value, or an attribute that has been removed. If you are not sure if an attribute exists or has a value use [IDLffDicomEx::QueryValue](#) before calling `GetValue`.

Syntax

```
Result = Obj->[IDLffDicomEx::]GetValue(DicomTag [, SEQID=integer]
    [, COUNT=variable])
```

Return Value

Returns one of the following:

- A scalar value for an attribute with one value.
- A vector of scalar values for attributes with multiple values.
- A long integer if the attribute is a sequence. This value is used as the `SEQID` keyword in subsequent calls to `GetValue` to access items contained in the sequence.
- A vector of values when the sequence contains groups (set of repeating tags within the sequence. See the [IDLffDicomEx::GetValueLength](#) method “Example” section for code that uses such an array.

Note

The DICOM standard requires that all values are an even length. If you assign an odd length string to a value, a trailing white space will be added. It may be necessary to trim trailing spaces (for VR types that allow trailing spaces to be ignored) when comparing an odd length string value to the value returned by `GetValue`.

Arguments

DicomTag

A string that identifies the group and element of a DICOM attribute in the form 'XXXX,XXXX'. The *DicomTag* argument must reference a public tag. See “[DICOM Attributes](#)” on page 300 for a list of tags.

Keywords

SEQID

Set this keyword only if retrieving the value of an attribute that exists within a sequence. Use this keyword to specify sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the `GetValue` method.
- Set to 0 or do not specify this keyword to indicate the attribute exists at the root level of the DICOM file. This is the default.

COUNT

Set this keyword equal to a named variable that will contain an unsigned long value indicating the number of elements in this method’s return value. Possible values are:

- 1 indicating the return value is a scalar value.
- *n* where *n* is the number of elements in the returned array. This corresponds to the number of values in the multi-valued attribute.

Example

The following example reads the multi-valued Image Type attribute from a DICOM file in the `examples\data` directory. See [IMAGE_TYPE](#) for more information on this attribute.

```
PRO read_imagetypeattr_doc

; Select a DICOM file to examine.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH(' ', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)
```

```
; Open the selected file in read-only mode.
oImg = OBJ_NEW('IDLffDicomEx', sfile)

; Return the Image Type attribute that can have 2 to n values.
result = oImg->GetValue('0008,0008', COUNT=vCount)

FOR i = 1, vCount DO BEGIN
    Print, 'Result number', i, + ' is ', result[i-1]
ENDFOR

END
```

For the `mr_knee.dcm` file, the following appears in the Output Log window:

```
Result number      1 is ORIGINAL
Result number      2 is PRIMARY
```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetValueCount

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::GetValueCount function method returns the number of values in a DICOM attribute specified by a standard DICOM attribute tag. This method allows you to return the number of values contained in public attributes. See [IDLffDicomEx::GetPrivateValueCount](#) for information on returning the number of values within a private attribute.

Note

GetValueCount will fail if you attempt to return a value for an attribute that does not exist or an attribute that has been removed. If you are not sure if an attribute exists use [IDLffDicomEx::QueryValue](#) before calling GetValueCount.

Syntax

```
Result = Obj->[IDLffDicomEx::]GetValueCount(DicomTag [, SEQID=integer] )
```

Return Value

Returns an unsigned long value indicating the number of values in the value field of the specified attribute as follows:

- A value of 0 indicates the tag had no value
- A value greater than 0 (*n*) indicates the number of values in the value field

Arguments

DicomTag

A string that identifies the group and element of a DICOM attribute in the form 'XXXX,XXXX'. The *DicomTag* argument must reference a public tag. See “[DICOM Attributes](#)” on page 300 for a list of tags.

Keywords

SEQID

Set this keyword only if retrieving the value of an attribute that exists within a sequence. Use this keyword to specify sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the `GetValue` method.
- Set to 0 or do not specify this keyword to indicate the attribute exists at the root level of the DICOM file. This is the default.

Example

The following example returns the number of items in a multi-valued Image Type attribute (0008,0008) and uses this value to cycle through the collection of values. For more information on the attribute, see [IMAGE_TYPE](#).

```
PRO read_count_imagetypeattr

; Select a DICOM file to examine.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('',SUBDIRECTORY=['examples','data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm')

; Open the selected file in read-only mode.
oImg = OBJ_NEW('IDLffDicomEx', sfile)

; Return the Image Type attribute count that can have
; 1 to n values. And then return the value of the tag.
vCount = oImg->GetValueCount('0008,0008')
result = oImg->GetValue('0008,0008')

FOR i = 1, vCount DO BEGIN
    Print, 'Result number', i, + ' is ', result[i-1]
ENDFOR

END
```

For example, when you select `us_test.dcm`, the following is printed to the Output Log window:

```
Result number      1 is ORIGINAL
Result number      2 is PRIMARY
Result number      3 is EPICARDIAL
```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetValueLength

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Examples](#) | [Version History](#)

The IDLffDicomEx::GetValueLength function method returns the length of all values or of a specified value (in bytes) in a standard DICOM attribute. This method allows you to return the length of values contained in public attributes. See [IDLffDicomEx::GetPrivateValueLength](#) for information on returning the length of values within a private attribute.

Note

GetValueLength will fail if you attempt to return a value for an attribute that does not exist or an attribute that has been removed. If you are not sure if an attribute exists use [IDLffDicomEx::QueryValue](#) before calling GetValueLength.

Syntax

```
Result = Obj->[IDLffDicomEx::]GetValueLength(DicomTag [, SEQID=integer]  
[, VALUEINDEX=integer])
```

Return Value

Returns a long integer indicating the length of one of the following:

- The length (in bytes) of all values when the VALUEINDEX keyword is not set
- The length (in bytes) of a single value specified by the VALUEINDEX keyword
- The number of repeating groups contained within a sequence if the attribute identified by the *DicomTag* argument is a sequence

Arguments

DicomTag

A string that identifies the group and element of a DICOM attribute in the form 'xxxxx, xxxxx'. The *DicomTag* argument must reference a public tag. See [“DICOM Attributes”](#) on page 300 for a list of tags.

Keywords

SEQID

Set this keyword only if retrieving the value of an attribute that exists within a sequence. Use this keyword to specify sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the `GetValue` method.
- Set to 0 or do not specify this keyword to indicate the attribute exists at the root level of the DICOM file. This is the default.

VALUEINDEX

Set this keyword to an integer indicating the one-based index number of the value for which to return the length. If not set, this method returns the length of a single value for a single-valued attribute, or the length of all values for a multi-valued attribute.

Note

An error will be issued if you specify a value larger than the number of values in the attribute.

Examples

Returning Lengths Using `GetValueLength`

The following code accesses the value length of all values in the multi-valued Image Type attribute (0008,0008) as well as the length of the last value in this attribute by using the `VALUEINDEX` keyword.

```
PRO read_vallength_doc

; Select a DICOM file to examine.
sfile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm')

; Open the selected file in read-only mode.
oImg = OBJ_NEW('IDLffDicomEx', sfile)

; Print the length and value of the Image Type attribute.
value = oImg->GetValue('0008,0008', count=vCount)
vLength = oImg->GetValueLength('0008,0008')
```

```

vLastLength = oImg->GetValueLength('0008,0008', VALUEINDEX=vCount)
PRINT, 'Length of all values = ', vLength, ' for values = ', value
PRINT, 'Length of last value = ', vLastLength

END

```

Accessing Repeating Groups Using GetValueLength

The following code uses the `GetValueLength` method to return the number of repeating groups in a sequence. As this method will fail if the specified DICOM attribute does not exist in a file, the example is hard-coded to use the `us_test.dcm` file in the `examples/data` directory.

```

PRO read_attributes_doc

; Select a DICOM file to examine.
sFile = FILEPATH('us_test.dcm', $
    SUBDIRECTORY=['examples','data'])

; Open the selected file in read-only mode.
oImg = OBJ_NEW('IDLffDicomEx', sfile)

; Return information from Sequence of Ultrasound Reigons.
; When a sequence has multiple groups, the sequence identifier
; returned by GetValue is a zero-based vector of values.
vSeqId = oImg->GetValue('0018,6011', COUNT=vCount)

; Using GetValueLength in conjunction with a sequence returns the
; number of repeating groups in the sequence.
vSeqLength = oImg->GetValueLength('0018,6011')

FOR i = 1, vSeqLength DO BEGIN
    ; Return the length and value of each Region Location Max X1
    ; (0018,601c) item for all groups within the sequence.
    vLength = oImg->GetValueLength('0018,601c', SEQID=vSeqId[i-1])
    result = oImg->GetValue('0018,601c', SEQID=vSeqId[i-1])
    Print, 'Sequence group ', i, + ' item length is ', vLength, + $
        ' and value is ', result
ENDFOR

END

```

The following is printed to the Output Log window:

```

Sequence group 1 item length is 4 and value is 625
Sequence group 2 item length is 4 and value is 0
Sequence group 3 item length is 4 and value is 0
Sequence group 4 item length is 4 and value is 0

```

The previous values are consistent with those shown in the following figure. The sequence for which the array of sequence identifiers was returned is shown on the first line. Notice that it is multi-valued, containing four repeating groups (only a portion of which are visible). The highlighted item indicates the first Region Location Max X1 attribute. The second corresponding attribute is shown near the bottom of the figure.

Tag	VR	Data	Length	MV	Description
0018,6011	SQ		4	4	Sequence of Ultrasound Regions
>[Group: 1]					
>0018,0000	UL	212	4	1	Group 0018 Length
>0018,6012	US	1	2	1	Region Spatial Format
>0018,6014	US	1	2	1	Region Data Type
>0018,6016	UL	2	4	1	Region Flags
>0018,6018	UL	10	4	1	Region Location Min X0
>0018,601A	UL	10	4	1	Region Location Min Y0
>0018,601C	UL	625	4	1	Region Location Max X1
>0018,601E	UL	445	4	1	Region Location Max Y1
>0018,6020	SL	319	4	1	Reference Pixel X0
>0018,6022	SL	54	4	1	Reference Pixel Y0
>0018,6024	US	3	2	1	Physical Units X Direction
>0018,6026	US	3	2	1	Physical Units Y Direction
>0018,6028	FD	0	8	1	Reference Pixel Physical Value X
>0018,602A	FD	0	8	1	Reference Pixel Physical Value Y
>0018,602C	FD	0.0445682	8	1	Physical Delta X
>0018,602E	FD	0.0445682	8	1	Physical Delta Y
>0018,6030	UL	3750000	4	1	Transducer Frequency
>0018,6032	UL	0	4	1	Pulse Repetition Frequency
>[Group: 2]					
>0018,0000	UL	212	4	1	Group 0018 Length
>0018,6012	US	0	2	1	Region Spatial Format
>0018,6014	US	0	2	1	Region Data Type
>0018,6016	UL	0	4	1	Region Flags
>0018,6018	UL	0	4	1	Region Location Min X0
>0018,601A	UL	0	4	1	Region Location Min Y0
>0018,601C	UL	0	4	1	Region Location Max X1
>0018,601E	UL	0	4	1	Region Location Max Y1

Figure 3-6: Accessing Values From Repeating Groups in a Sequence

Note

For information on writing the values of DICOM attributes to the output window or a file, see the [IDLffDicomEx::EnumerateTags](#) method.

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::GetVR

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::GetVR function method returns the Value Representation (VR) of a DICOM attribute. This method allows you to return the VR of a public attribute. See [IDLffDicomEx::GetPrivateVR](#) for information on returning the VR of a private attribute.

Note

GetVR will fail if you attempt to return a value for an attribute that does not exist or an attribute that has been removed. If you are not sure if an attribute exists use [IDLffDicomEx::QueryValue](#) before calling GetVR.

Syntax

```
Result = Obj->[IDLffDicomEx::]GetVR(DicomTag [, SEQID=integer] )
```

Return Value

Returns a string indicating the Value Representation (VR) of the DICOM attribute specified by the *DicomTag* argument. See “[Value Representations](#)” on page 372 for more information on individual value representations.

Arguments

DicomTag

A string that identifies the group and element of a DICOM attribute in the form 'XXXX, XXXX'. The *DicomTag* argument must reference a public tag. See “[DICOM Attributes](#)” on page 300 for a list of tags.

Keywords

SEQID

Set this keyword only if retrieving the value of an attribute that exists within a sequence. Use this keyword to specify sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the GetValue method.

- Set to 0 or do not specify this keyword to indicate the attribute exists at the root level of the DICOM file. This is the default.

Example

The following example returns the VR and value of the Modality attribute (0008,0060) of a selected DICOM file. This example does not make sure the attribute exists before returning it as this is a mandatory tag for valid DICOM files.

```
PRO read_vr_doc

; Select a DICOM file to examine.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm')

; Open the selected file in read-only mode.
oImg = OBJ_NEW('IDLffDicomEx', sfile)

; Return information from the Modality attribute.
vValue = oImg->GetValue('0008,0060')
vVR = oImg->GetVR('0008,0060')
PRINT, 'Modality VR = ', vVR, + ' and value is ', vValue

END
```

For the `mr_brain.dcm` file, the output is:

```
Modality VR = CS and value is MR
```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::Init

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Examples](#) | [Version History](#)

The `IDLffDicomEx::Init` method initializes a `IDLffDicomEx` object. The `IDLffDicomEx` object allows you to read and write DICOM files, or create a new DICOM file based on keyword settings. This method can be used to create any type of DICOM file, including files without image data as defined by the SOP Class. See “[Creating a New DICOM File](#)” on page 165 for more information. *The original DICOM file is always preserved.* To change the attributes of a DICOM file, you must either clone the original file or create a new file. See the following sections for details on using keywords to control how a file can be modified:

- [Accessing a DICOM File in Read-Only Mode](#)
- [Cloning a DICOM File](#)
- [Creating a New DICOM File](#)
- [Accessing an Incomplete DICOM File](#)

Note

If you don't need pixel data, you can set the `NO_PIXEL_DATA` property when creating an `IDLffDicomEx` object. This offers a significant performance improvement and should be used when you only need access to attribute information. To initialize the value of a property, specify the property name as a keyword set equal to the appropriate property value.

Note

Init methods are special lifecycle methods, and as such cannot be called outside the context of object creation. This means that in most cases, you cannot call the Init method directly. There is one exception to this rule: if you write your own subclass of this class, you can call the Init method from within the Init method of the subclass.

Accessing a DICOM File in Read-Only Mode

To open an existing DICOM file in read-only mode, create a new `IDLffDicomEx` object and specify only a value for the *Filename* argument, indicating the file to open. You can access all of the DICOM file attributes, but any attempt to write changes to the file using the `IDLffDicomEx::Commit` method will fail. The following code opens a selected DICOM file in read-only mode:

```
; Select a DICOM file to examine.
```

```
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm')

; Open the selected file in read-only mode.
oImg = OBJ_NEW('IDLffDicomEx', sfile)
```

Cloning a DICOM File

To modify an existing DICOM file, you must clone the file by setting the following keywords and arguments:

- Set the *Filename* argument to specify the path and name of the new file
- Set CLONE to the name of the existing DICOM file to be cloned

Note

Init will fail if the file defined by *Filename* already exists.

The following clones the selected file:

```
; Select a DICOM file to clone.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Write the file into the temp directory and then
; create a clone (aImgClone.dcm) of the selected file (sfile).
clonefile = GETENV('IDL_TMPDIR')+PATH_SEP()+ 'aImgClone.dcm'
oImg = OBJ_NEW('IDLffDicomEx', clonefile, CLONE=sfile)
```

When you clone a file, attributes shown in the following table are modified. Unless the internally-generated values equal the number of characters in the original tags, the value of the metadata Group Length tag (0002,0000) is also updated. All other DICOM attribute values are identical to the original attributes.

DICOM Tag	Modification
(0002,0003)	Media Storage SOP Instance UID is set to a new ITT Visual Information Solutions-generated value.
(0002,0012)	Implementation Class UID is set to a new ITT Visual Information Solutions-generated value.

Table 3-10: DICOM Attributes Set When Cloning a File

DICOM Tag	Modification
(0002,0013)	Implementation Version Name is set to the ITT Visual Information Solutions value.
(0002,0016)	Source Application Entity Title is set to the ITT Visual Information Solutions value.
(0008,0018)	SOP Instance UID is set to a new ITT Visual Information Solutions-generated value.

Table 3-10: DICOM Attributes Set When Cloning a File (Continued)

Note

These attributes are written to the cloned DICOM file when you call the [IDLffDicomEx::Commit](#) method.

The resulting cloned file will allow you to modify any attributes that belong to the specified SOP Class for the cloned file. If the `NON_CONFORMING` keyword is set, then you can set any attribute regardless of the SOP Class of the cloned file.

Note

It may also be necessary to modify the Series Instance UID (0020,000E) when cloning a file so that when the cloned file is pushed to a PACS workstation it stored in a new series.

Creating a New DICOM File

To create a new DICOM file, you must set the following:

- Set the *Filename* argument to specify the path and name of the new file
- Set the `SOP_CLASS` keyword to an appropriate value to indicate the type of DICOM file to create
- Set `CREATE` keyword to indicate this is a new DICOM file

The following code creates a new file with a SOP Class of standard Magnetic Resonance (MR):

```
; Create a new image named aMRImg.dcm in the current
; working directory.
oImage = OBJ_NEW('IDLffDicomEx', 'aMRImg.dcm', $
    SOP_CLASS = 'STANDARD_MR', /CREATE, /NON_CONFORMING )
```

When a new file is created, all defined tags for the chosen SOP Class are present, but do not have a value. You must use [IDLffDicomEx::SetValue](#) or [IDLffDicomEx::SetProperty](#) to set valid values prior to calling the `GetValue` or `GetProperty` methods. These methods will return an error if you attempt to return information for an attribute that does not have a value. Any attribute that you have not set a value for will not be persisted in the file when you call `Commit`. Use `SetValue` to configure the attributes required to create a valid DICOM file for the chosen SOP class. (Complete details can be found in *Digital Imaging and Communications in Medicine (DICOM) - Part 3: Information Object Definitions*.) To set attributes that are not defined for the SOP class, set the `NON_CONFORMING` keyword. Creating a new file sets values for the following attributes:

DICOM Tag	Modification
(0002,0002)	Media Storage SOP Class UID is set to the unique identifier associated with the <code>SOP_CLASS</code> keyword.
(0002,0003)	Media Storage SOP Instance UID is set to a new ITT Visual Information Solutions-generated value.
(0002,0010)	Transfer Syntax UID is set to Explicit VR Little Endian by default. After pixel data has been set on the new image, you can use the IDLffDicomEx::ChangeTransferSyntax method to change the file compression.
(0002,0012)	Implementation Class UID is set to a new ITT Visual Information Solutions-generated value.
(0002,0013)	Implementation Version Name is set to the ITT Visual Information Solutions value.
(0002,0016)	Source Application Entity Title is set to the ITT Visual Information Solutions value.
(0008,0016)	SOP Class UID is set to the unique identifier associated with the <code>SOP_CLASS</code> keyword.
(0008,0018)	SOP Instance UID is set to a new ITT Visual Information Solutions-generated value.

Table 3-11: DICOM Attributes Set When Creating a File

Note

These attributes are written to the new DICOM file when you call the `IDLffDicomEx::Commit` method.

When creating a new DICOM file it is also a good idea to set the following tags. These tags are commonly needed when transmitting a file over a DICOM network.

DICOM Tag	Description
(0010,0010)	Patient Name.
(0010,0020)	Patient ID.
(0020,000D)	Study Instance UID. When creating a new study the Image Instance UID can be used as a based value to which a unique suffix can be added. When adding an image to an existing study the existing study instance UID can be used.
(0020,000E)	Series Instance UID. When creating a new series the Image Instance UID can be used as a based value to which a unique suffix can be added. When adding an image to an existing series the existing series instance UID can be used.

Table 3-12: DICOM Attributes Required for Query/Retrieve Transmission

The file is written when the `IDLffDicomEx::Commit` method is called. Attempting to overwrite a file with an existing file name will fail.

Creating New Files That Use Alternative Character Sets

Several Value Representations accept ESC characters from alternative character sets. In such instances, the text will be interpreted as specified by Specific Character Set attribute (0008,0005). For example, if you are setting Japanese escape characters to an attribute value in a new file, you first need to specify your character sets as in the following code:

```
; Given a IDLffDicomEx object named oDCM:
val = STRARR(2)
val[0] = 'ISO 2022 IR 87'
val[1] = 'ISO 2022 IR 13'
oDCM->SetValue, '0008,0005', 'CS', val
```

See Annex H of *Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding* for more information on Japanese character sets.

Accessing an Incomplete DICOM File

A file that conforms to the DICOM Part 10 standard consists of a preamble, metadata, and a body section. Each section contains an even number of bytes. Not all files have all three sections; some files may be missing the preamble, or the preamble and metadata. Not all files contain an even number of bytes in each section. Files that do not conform to the DICOM Part 10 standard are considered *invalid*, but it is possible to access the data in these files using the IDLffDicomEx object as described in the following sections.

Files With a Missing Preamble

The IDLffDicomEx object will attempt to read the transfer syntax from the (0002, 0010) tag and open a file with a missing preamble. The following table shows each file type that can be opened when the preamble section is missing.

Image Type / Transfer Syntax	Recoverable
Implicit VR Little Endian	Yes
Explicit VR Little Endian	Yes
Explicit VB Big Endian	Yes
JPEG Baseline	Yes
JPEG Extended (Process 2 and 4)	Yes
JPEG Lossless, Non-Hierarchical	Yes
JPEG 2000 Lossless	Yes
JPEG 2000	Yes

Table 3-13: Recoverable DICOM Files Missing the Preamble

Files With Missing Preamble and Metadata Sections

The IDLffDicomEx object will attempt to read a file that is missing the preamble and metadata by determining the transfer syntax from the byte ordering and VR type of the first tag in the file. A file containing JPEG pixel data cannot be opened as it is impossible to determine the compression format of the pixel data when the metadata

tags are not in the file. The following table shows each file type that can be opened when missing preamble and metadata sections.

Image Type / Transfer Syntax	Recoverable
Implicit VR Little Endian	Yes
Explicit VR Little Endian	Yes
Explicit VB Big Endian	Yes
JPEG Baseline	No
JPEG Extended (Process 2 and 4)	No
JPEG Lossless, Non-Hierarchical	No
JPEG 2000 Lossless	No
JPEG 2000	No

Table 3-14: Recoverable DICOM Files Missing the Preamble and Metadata

The `IDLffDicomEx` object will add metadata tags to the file so that the file can be successfully saved as a part 10 DICOM file using the `IDLffDicomEx::Commit` method. The following tags will be written in the metadata section of the recovered file.

DICOM Tag	Modification
(0002,0002)	Media Storage SOP Class UID is set to the unique identifier associated with the <code>SOP_CLASS</code> keyword.
(0002,0003)	Media Storage SOP Instance UID is set to a new ITT Visual Information Solutions-generated value.
(0002,0010)	Transfer Syntax UID is set to Implicit VR Little Endian, Explicit VR Little Endian, or Explicit VB Big Endian based on byte ordering and VR type of the first tag in the file.
(0002,0012)	Implementation Class UID is set to a new ITT Visual Information Solutions-generated value.

Table 3-15: DICOM Attributes Set When Recovering a File

DICOM Tag	Modification
(0002,0013)	Implementation Version Name is set to the ITT Visual Information Solutions value.
(0002,0016)	Source Application Entity Title is set to the ITT Visual Information Solutions value.

Table 3-15: DICOM Attributes Set When Recovering a File (Continued)

Note

A new instance UID (0002,0003) is created for the recovered file since the IDLffDicomEx object never modifies the original input file.

Files With an Odd Number of Bytes

A file that conforms to the DICOM Part 10 standard always includes an even number of bytes in each section. If the IDLffDicomEX object encounters a file with a section that consists of an odd number of bytes, IDL will append a zero at the end of the last block of data read and issue a warning message noting that the section contained an odd number of bytes.

Warning

Use files with odd-length sections with extreme caution. Files with an odd number of bytes are not valid DICOM files.

Syntax

```
Obj = OBJ_NEW('IDLffDicomEx' (Filename, [, CLONE=string] [, /CREATE]
    [, SOP_CLASS=string] [, /NON_CONFORMING] )
```

or

```
Result = Obj->[IDLffDicomEx::]Init( Filename [, CLONE=string] [, /CREATE]
    [, SOP_CLASS=string] [, /NON_CONFORMING] ) (In a lifecycle method only.)
```

Return Value

When this method is called indirectly, as part of the call to the OBJ_NEW function, the return value is an object reference to the newly-created object.

When called directly within a subclass Init method, the return value is 1 if initialization was successful, or 0 otherwise.

Arguments

FileName

Set this keyword to a string indicating the filename of a DICOM file. This can either be an absolute path ('C:\my_dcm_file.dcm'), or simply a filename ('my_dcm_file.dcm'). When only a filename is provided, the file is located in the IDL working directory. The exact meaning of the *FileName* argument depends on what keywords are set as follows:

- If no keywords are set, *FileName* indicates the existing DICOM file to open in read-only mode. Any attempt to commit changes to this file (using the `IDLffDicomEx::Commit` method) will fail. An error is issued if the specified file does not exist.
- If the `CLONE` keyword is set, *FileName* specifies the name of the new, cloned file. This file will contain a copy of the file specified by the `CLONE` keyword. An error is issued if the value specified for *FileName* already exists.
- If the `CREATE` keyword is set, *FileName* specifies the name of the new DICOM file to be created. An error is issued if the value specified for *FileName* already exists.

Keywords

CLONE

Set this keyword to a string specifying path and name of the existing DICOM file to be cloned. Define the name of the new, cloned file using the *FileName* argument. See [“Cloning a DICOM File”](#) on page 164 for details.

CREATE

Set this keyword to create a new DICOM image with a name specified by the *FileName* argument. See [“Creating a New DICOM File”](#) on page 165 for details. You must also set the `SOP_CLASS` keyword when creating a new image.

SOP_CLASS

This keyword is set when creating a new DICOM file using the `CREATE` keyword.

Set this keyword to a string consisting of a value from the SOP Class Name column in the following table to define the type of DICOM file that is created.

SOP Class Name	SOP Class UID
STANDARD_CR (Computed Radiography)	1.2.840.10008.5.1.4.1.1.1
STANDARD_DX_PRESENT (Digital X-ray)	1.2.840.10008.5.1.4.1.1.1.1
STANDARD_DX_PROCESS (Digital X-ray)	1.2.840.10008.5.1.4.1.1.1.1.1
STANDARD_MG_PRESENT (Digital Mammography)	1.2.840.10008.5.1.4.1.1.1.2
STANDARD_MG_PROCESS (Digital Mammography)	1.2.840.10008.5.1.4.1.1.1.2.1
STANDARD_IO_PRESENT (Digital Intra-oral)	1.2.840.10008.5.1.4.1.1.1.3
STANDARD_IO_PROCESS (Digital Intra-Oral)	1.2.840.10008.5.1.4.1.1.1.3.1
STANDARD_CT	1.2.840.10008.5.1.4.1.1.2
STANDARD_US_MF_RETIRED	1.2.840.10008.5.1.4.1.1.3
STANDARD_US_MF	1.2.840.10008.5.1.4.1.1.3.1
STANDARD_MR	1.2.840.10008.5.1.4.1.1.4
ENHANCED_MR_IMAGE	1.2.840.10008.5.1.4.1.1.4.1
MR_SPECTROSCOPY	1.2.840.10008.5.1.4.1.1.4.2
STANDARD_NM_RETIRED	1.2.840.10008.5.1.4.1.1.5
STANDARD_US_RETIRED	1.2.840.10008.5.1.4.1.1.6
STANDARD_US	1.2.840.10008.5.1.4.1.1.6.1
STANDARD_SEC_CAPTURE	1.2.840.10008.5.1.4.1.1.7
SC_MULTIFRAME_SINGLE_BIT	1.2.840.10008.5.1.4.1.1.7.1
SC_MULTIFRAME_GRAYSCALE_BYTE	1.2.840.10008.5.1.4.1.1.7.2
SC_MULTIFRAME_GRAYSCALE_WORD	1.2.840.10008.5.1.4.1.1.7.3
SC_MULTIFRAME_TRUE_COLOR	1.2.840.10008.5.1.4.1.1.7.4

Table 3-16: Allowable SOP Class Values

SOP Class Name	SOP Class UID
STANDARD_OVERLAY	1.2.840.10008.5.1.4.1.1.8
STANDARD_CURVE	1.2.840.10008.5.1.4.1.1.9
STANDARD_MODALITY_LUT	1.2.840.10008.5.1.4.1.1.10
STANDARD_VOI_LUT	1.2.840.10008.5.1.4.1.1.11
STANDARD_GRAYSCALE_SOFTCOPY_PS	1.2.840.10008.5.1.4.1.1.11.1
STANDARD_XRAY_ANGIO	1.2.840.10008.5.1.4.1.1.12.1
STANDARD_XRAY_RF	1.2.840.10008.5.1.4.1.1.12.2
STANDARD_XRAY_ANGIO_BIPLANE (retired)	1.2.840.10008.5.1.4.1.1.12.3
STANDARD_NM	1.2.840.10008.5.1.4.1.1.20
RAW_DATA	1.2.840.10008.5.1.4.1.1.66
STANDARD_VL_ENDOSCOPIC	1.2.840.10008.5.1.4.1.1.77.1.1
STANDARD_VL_MICROSCOPIC	1.2.840.10008.5.1.4.1.1.77.1.2
STANDARD_VL_SLIDE_MICROSCOPIC	1.2.840.10008.5.1.4.1.1.77.1.3
STANDARD_VL_PHOTOGRAPHIC	1.2.840.10008.5.1.4.1.1.77.1.4
STANDARD_BASIC_TEXT_SR	1.2.840.10008.5.1.4.1.1.88.11
STANDARD_ENHANCED_SR	1.2.840.10008.5.1.4.1.1.88.22
STANDARD_COMPREHENSIVE_SR	1.2.840.10008.5.1.4.1.1.88.33
STANDARD_PET (Positron Emission Tomography)	1.2.840.10008.5.1.4.1.1.128
STANDARD_PET_CURVE	1.2.840.10008.5.1.4.1.1.129
STANDARD_RT_IMAGE	1.2.840.10008.5.1.4.1.1.481.1
STANDARD_RT_DOSE	1.2.840.10008.5.1.4.1.1.481.2
STANDARD_RT_STRUCTURE_SET	1.2.840.10008.5.1.4.1.1.481.3
STANDARD_RT_BEAMS_TREAT	1.2.840.10008.5.1.4.1.1.481.4

Table 3-16: Allowable SOP Class Values (Continued)

SOP Class Name	SOP Class UID
STANDARD_RT_PLAN	1.2.840.10008.5.1.4.1.1.481.5
STANDARD_RT_BRACHY_TREAT	1.2.840.10008.5.1.4.1.1.481.6
STANDARD_RT_TREAT_SUM	1.2.840.10008.5.1.4.1.1.481.7

Table 3-16: Allowable SOP Class Values (Continued)

NON_CONFORMING

This keyword is set only when the CLONE or CREATE keyword is also set.

Set this keyword to be able to add any DICOM attribute to a DICOM file regardless of whether or not the attribute is supported by the SOP Class (as defined in *Digital Imaging and Communications in Medicine (DICOM) - Part 3: Information Object Definitions*).

If this keyword is not set, attempting to use `IDLffDicomEx::SetValue` to set non-standard attributes will result in an invalid tag error similar to the following:

```
IDLFFDICOMEX::SETVALUE: Error: Failed to set value (tag/err),
0018,603F, Tag parameter invalid
```

Examples

The following examples show various ways of initializing an `IDLffDicomEx` object.

Open a DICOM File in Read-only Mode

The following opens a file in read-only mode:

```
; Select a DICOM file to examine.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm')

; Open the selected file in read-only mode.
oImg = OBJ_NEW('IDLffDicomEx', sfile)
```

See the `IDLffDicomEx::EnumerateTags` method “Examples” section for a complete example.

Open and Clone a DICOM File

The following clones the selected file:

```

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile)

```

See “[Cloning a DICOM File](#)” on page 164 for details on what DICOM attributes are modified. See the [IDLffDicomEx::GetPixelData](#) method “Examples” section for a complete example.

Create a New DICOM File

The following code creates a new file of modality MR:

```

; Create a new image named aMRImg.dcm in the current
; working directory.
oImage = OBJ_NEW('IDLffDicomEx', 'aMRImg.dcm', $
    SOP_CLASS = 'STANDARD_MR', /CREATE, /NON_CONFORMING )

```

See “[Creating a New DICOM File](#)” on page 165 for details on what DICOM attributes are automatically defined when you create a new file. See the [IDLffDicomEx::SetPixelData](#) method “Examples” section for a complete example that creates new monochrome and RGB images.

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::QueryPrivateValue

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::QueryPrivateValue function method checks a DICOM file for the presence of a specified private attribute. This method allows you to verify the presence of a tag prior to calling a method that requires a DICOM attribute to be present in order to succeed. Attempting to call GetPrivateValue, GetPrivateValueCount, GetPrivateValueLength, and GetPrivateVR methods all return an error when you attempt to access an attribute that does not exist in a DICOM file. GetPrivateValue also fails when attempting to access an attribute that does not have a value.

Syntax

```
Result = Obj->[IDLffDicomEx::]QueryPrivateValue(PrivateCode, Group, Element  
[, SEQID=integer])
```

Return Value

This method returns one of the following:

- 0 = tag not found
- 1 = tag found but does not have a value
- 2 = tag found and has a value

A return value of 0 or 1 indicates attempting to call GetPrivateValue would cause an error. A return value of 2 means GetPrivateValue would succeed for the specified attribute.

Arguments

PrivateCode

A string identification code that identifies the private block of data. Within a given private group *PrivateCode* labels are stored sequentially in the element addresses ranging from '0010' to '00FF'. For example, the string value stored at DICOM tag address '0029,0010' is the *PrivateCode* for the block of data tagged at '0029,1000' - '0029,10FF'. The label stored at '0029,0011' would be the *PrivateCode* for the data in tags '0029,1100' - '0029,11 FF'.

Group

A string identifying the group tag number of the private attribute (the first four digits of a DICOM tag). This must be an odd number and in the form 'XXXX'.

Element

A string identifying the last two digits of the element associated with the private attribute. This must be in the form 'XX'. Valid values are 10 - FF.

Note

The first two digits of the *Element* are implicit in the *PrivateCode* argument.

Keywords

SEQID

Set this keyword to a long integer only if the private attribute exists within a sequence. Use this keyword to specify a sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the `GetPrivateValue` method.
- Set to 0 or do not specify this keyword to indicate the private attribute exists at the root level of the DICOM file. This is the default.

Example

The following example verifies the existence of a private tag that has just been committed to a file. If the tag exists, a message is printed to the Output Log window. Typically you would proceed to access private values as shown in the code in the “Examples” section of [IDLffDicomEx::GetPrivateValueLength](#).

```
PRO dicom_queryprivate_doc

; Select a DICOM file.
sfile = DIALOG_PICKFILE( $
    PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile)
```

```

; Create a sequence at the root level.
vSeqID = oImg->AddPrivateSequence('VOI Min,Max', '0055', '12')

; Add items to the sequence, specifying SQ identifier returned by
; AddPrivateSequence.
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '13', 'IS', '215', $
    SEQID=vSeqID
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '14', 'IS', '234', $
    SEQID=vSeqID

; Commit the changes.
oImg->Commit

; Make sure the private sequence exists.
vQuery = oImg->QueryPrivateValue('VOI Min,Max', '0055', '12')
If vQuery NE 0 THEN $
    PRINT, 'Private Sequence Exists in File.'

; Clean up object references.
OBJ_DESTROY, oImg

; Note: the following line allows you to run the project
; multiple times without having to manually delete the file.
; You cannot duplicate an existing file when creating or cloning
; a DICOM file.
FILE_DELETE, path + ' aImgClone.dcm', /ALLOW_NONEXISTENT

END

```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::QueryValue

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::QueryValue function method checks a DICOM file for the presence of a specified attribute. This method allows you to verify the presence of a tag prior to calling a method that requires a DICOM attribute to be present and have a value in order to succeed. Attempting to call GetValue, GetValueCount, GetValueLength, GetVR, and GetProperty methods all return an error when you attempt to access an attribute that does not exist in a DICOM file. GetValue and GetProperty also fail when attempting to access an attribute that does not have a value.

This method is especially useful for determining the number of frames in an image prior to attempting to use this value when accessing pixel data. The Number of Frames tag is typically only present in multi-frame image files so when writing code that handles both single frame and multi-frame images, QueryValue can be used to determine if the Number of Frames DICOM attribute is present in the file.

Tip

For IDLffDicomEx properties, you can query using a property name (e.g., [NUMBER_OF_FRAMES](#)) instead of having to specify the DICOM attribute (e.g., 0028,0008).

Syntax

Result = Obj->[IDLffDicomEx::]QueryValue(*DicomTag* [, [SEQID=integer](#)])

Return Value

This method returns one of the following:

- 0 = tag not found
- 1 = tag found but does not have a value
- 2 = tag found and has a value

A return value of 0 or 1 indicates attempting to call GetValue would cause an error. A return value of 2 means GetValue would succeed for the specified attribute.

Arguments

DicomTag

A string that identifies either of the following:

- A group and element of a DICOM attribute in the form 'XXXX,XXXX'. The *DicomTag* argument must reference a public tag. See “[DICOM Attributes](#)” on page 300 for a list of tags.
- An IDLffDicomEx property name, such as BITS_ALLOCATED. Allowable property names are any of those listed in “[IDLffDicomEx Properties](#)” on page 70.

Note

When querying an IDLffDicomEx property name, the SEQID keyword is ignored. All named properties exist at the root level of the DICOM file, not within sequences.

Keywords

SEQID

Set this keyword only if retrieving the value of an attribute that exists within a sequence. Use this keyword to specify a sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the GetValue method.
- Set to 0 or do not specify this keyword to indicate the attribute exists at the root level of the DICOM file. This is the default.

Example

The following code checks for the Number of Frames attribute (0028,0008) indicating the image contains multiple frames. Attempting to just return a value for this attribute may fail as not all image SOP Classes require this tag to be present for single-frame images. For a complete example, see the “[Example](#)” section of [IDLffDicomEx::ChangeTransferSyntax](#).

Note

In an image containing multiple frames, the returned frames are indexed in a zero-based array; hence the Number of Frames (attribute or property) value minus one will return the desired frame when accessing pixel data.

```
; Check to see if the image has multiple frames.  
frameTest = oImg->QueryValue('0028,0008')  
IF FrameTest EQ 2 THEN BEGIN  
    oImg->GetProperty, NUMBER_OF_FRAMES=frames  
    frames = frames - 1  
ENDIF ELSE BEGIN  
    frames = 0  
ENDELSE
```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::SetPixelData

[Syntax](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The `IDLffDicomEx::SetPixelData` function method writes pixel data to the DICOM image file. DICOM files can store pixel data for a single-frame image or a multi-frame image. This method accepts uncompressed pixel data. If the transfer syntax of the DICOM file indicates the image is to be stored in a compressed format, the data will be compressed by this method. As long as the specified pixel data array has the correct number of bytes, it can have any dimensions. See the *PixelData* argument for details.

Note

Pixel data changes are written to the DICOM file only when you call the [IDLffDicomEx::Commit](#) method.

Writing Frames of Pixel Data to Lossy and Lossless Formats

When the transfer syntax indicates a lossy JPEG compression format (JPEG Baseline, JPEG Extended, or JPEG 2000), you must pass in all the pixel data for all the frames in a multi-frame image. This method does not support writing a single frame of pixel data to a multi-frame image that is stored in a lossy JPEG compression format. This is prohibited to prevent the degradation of data that occurs when a frame is uncompressed then recompressed in a lossy format.

This method does support writing a single frame of pixel data to multi-frame image stored in a lossless JPEG compression format (JPEG Lossless, JPEG 2000 Lossless). When the compression format is lossless, a frame of data can be uncompressed and recompressed without losing of data.

Specifying Pixel Data For a New Image

When assigning pixel data to a brand new image, you must set the following properties either *before* setting the pixel data or *while* setting pixel data. Once these values are set in a new image, and the `SetPixelData` method has been called, do not change the values as the underlying pixel data will not reflect changes to these tags.

Note

Many other tags are needed (and specified by the DICOM standard) for the creation of a valid DICOM image for a particular SOP Class. (Complete details can be found in *Digital Imaging and Communications in Medicine (DICOM) - Part 3*:

Information Object Definitions.) This list shows only the tags needed to permit pixel data to be added to an image.

When setting pixel data on a new image, the following attributes are required to the construct the pixel data array.

Note

You can set these attributes using `IDLffDicomEx::SetValue`, `IDLffDicomEx:: SetProperty`, or by setting keywords to this method.

DICOM Attribute	Description
<code>BITS_ALLOCATED</code>	Typical values are 8 bits or 16 bits. An error is issued if this is absent.
<code>COLUMNS</code>	Number of vertical lines in a frame. An error is issued if this is absent.
<code>NUMBER_OF_FRAMES</code>	This tag is required for a multi-frame image. This tag is only allowed in SOP Classes that support multi-frame images. The default value is 1.
<code>PHOTOMETRIC_INTERPRETATION</code>	This tag is required for RGB images, but should be set when creating a new image.
<code>PIXEL_REPRESENTATION</code>	This tag is not required to set pixel data, but is required to properly determine how to return the data (using the <code>GetPixelData</code> method) in the correct format for images with greater than 8 bit signed or unsigned data. The default value is 0.

Table 3-17: Property Specifications Required Before or While Setting Pixel Data

DICOM Attribute	Description
PLANAR_CONFIGURATION	This tag is required for non-monochrome images as it is used in determining how the GetPixelData method ORDER keyword operates on the pixel data. The default value is 0.
ROWS	Number of horizontal lines in a frame. An error is issued if this is absent.
SAMPLES_PER_PIXEL	Typical values are 1 for monochrome frames and 3 for RGB frames. An error is issued if this is absent.

Table 3-17: Property Specifications Required Before or While Setting Pixel Data (Continued)

Typical values for attributes that are dependent upon the data type of the image are shown in the following table:

Property	UINT	INT	BYTE	RGB
BITS_ALLOCATED	16	16	8	8
BITS_STORED	10	16	8	8
HIGH_BIT	9	15	7	7
PHOTOMETRIC_INTERPRETATION	†	†	†	RGB
SAMPLES_PER_PIXEL	1	1	1	3
PIXEL_REPRESENTATION	0	1	0	0
PLANAR_CONFIGURATION	N/A	N/A	N/A	0 or 1

Table 3-18: Attribute Values Based on Data Types

† MONOCHROME2

Syntax

Obj->[IDLffDicomEx::] SetPixelData, *PixelData* [, **FRAME=integer**] [, **/ORDER**] [, **BITS_ALLOCATED=integer**] [, **COLUMNS=integer**]


```
[, NUMBER_OF_FRAMES=string]
[, PHOTOMETRIC_INTERPRETATION={MONOCHROME1 |
MONOCHROME2 | PALETTECOLOR | RGB | HSV | CMYK}]
[, PIXEL_REPRESENTATION={0 | 1}]
[, PLANAR_CONFIGURATION={0 | 1}] [, ROWS=integer]
[, SAMPLES_PER_PIXEL={1 | 3 | 4}]
```

Arguments

PixelData

The pixel data array for one frame of data or a pixel data array for all the frames.

The incoming array of pixel data must be the exact size of a single frame (when setting a single frame) or of all the frames (when setting all frames). While the array can have any dimensions, it must conform to the following:

- The format of the array must conform to the PLANAR_CONFIGURATION and PHOTOMETRIC_INTERPRETATION values of the image.
- The size of the pixel data array must be equal to:

$$\text{SamplesPerPixel} * \text{Rows} * \text{Columns} * \text{NumFramesToWrite} * \text{NumBytesPerSample}$$

where:

- SamplesPerPixel = DICOM attribute (0028,0002)
- Rows = DICOM attribute (0028,0010)
- Columns = DICOM attribute (0028,0011)
- NumFramesToWrite = 1 or DICOM attribute Number of Frames (0028,0008) in the image file
- NumBytesPerSample = 1 when the Bits Allocated (0028,0100) is less than or equal to 8, or NumBytesPerSample = 2 when the Bits Allocated (0028,0100) is greater than 8.

Keywords

Note

See “[Specifying Pixel Data For a New Image](#)” on page 182 for information on keywords listed in the syntax, but not shown here.

FRAME

Set this keyword to the zero-based index of the frame of pixel data to write. Allowable values for FRAME range from 0 to NUMBER_OF_FRAMES - 1. If FRAME is not specified, the pixel data of all frames in the PixelData array are written to the image. Otherwise, only the pixel data of the one, specified frame is written.

ORDER

Set the keyword when passing in pixel data in DICOM format, where the first pixel in the array is the top left-hand pixel in the frame. The SetPixelData method will not flip the rows when writing the data. If this keyword is not set, the array is in standard IDL format, where the first pixel in the array is the bottom left-hand pixel in the frame. The SetPixelData method will flip the rows before writing the pixel data to the DICOM image.

Example

The following example allows you to select an image file of any format. Based on properties of the data, the required pixel data attributes are set. The example creates a MR file for monochrome or palette image data, or a US file for RGB data. Note that this example sets only the smallest possible number of attributes required for setting pixel data. This does not create a valid DICOM file as the other tags mandated by the SOP class have not been set as required by the DICOM standard. (Complete details can be found in *Digital Imaging and Communications in Medicine (DICOM) - Part 3: Information Object Definitions*.)

Example Code

The code for `dicomex_importimage_doc.pro` is provided in the IDL distribution, in the `examples/doc/dicom` subdirectory of the main IDL directory. You can run the example code directly by entering `.EDIT dicomex_importimage_doc` at the IDL prompt.

Note

By default, the transfer syntax is set to Explicit VR Little Endian when a new file is created. After pixel data has been set on the new image, you can use the [IDLffDicomEx::ChangeTransferSyntax](#) method to change the file compression.

Note

The last few lines of the program delete the files that are created, so that the example can be run multiple times without an error occurring because you are

attempting to create a file with an existing filename. Comment out these lines to retain the images.

```

PRO dicomex_importimage_doc
; Import in the pixel data of an image, and
; then set it as the pixel data for a new Image
; object.

; Determine the full path to the image file.
sFile = DIALOG_PICKFILE(/MUST_EXIST, $
    TITLE = 'Select an Image File', $
    FILTER = ['*.bmp', '*.jpg', '*.png', $
        '*.ppm', '*.srf', '*.tif'], $
    GET_PATH=path)

; If no file is selected, return to the previous
; level.
IF (sFile EQ '') THEN RETURN

; Query the image file.
vOpenStatus = QUERY_IMAGE(sFile, vQueryInfo)

; If the file cannot be opened with IDL, return
; to the previous level.
IF (vOpenStatus NE 1) THEN RETURN

; Initialize some of the image parameters.
vNumSamples = vQueryInfo.channels
vCols = vQueryInfo.dimensions[0]
vRows = vQueryInfo.dimensions[1]
vImgSize = vQueryInfo.dimensions
vNumFrames = vQueryInfo.num_images
vPixelType = vQueryInfo.pixel_type

; Handle single channel images.
If vNumSamples EQ 1 THEN BEGIN
    CASE vPixelType of
        1: BEGIN
            ; Set properties for byte data.
            vBitsAlloc = 8
            vPixelRep = 0 ; accept the default.
            vPhotoInterp = 'MONOCHROME2'
        END

        2: BEGIN
            ; Set properties for signed integer.
            vBitsAlloc = 10
            vPixelRep = 1
    
```

```

        vPhotoInterp = 'MONOCHROME2'
    END

    12: BEGIN
        ; Set properties for unsigned integer.
        vBitsAlloc = 16
        vPixelRep = 0
        vPhotoInterp = 'MONOCHROME2'
    END
ENDCASE

; If the file contains multiple images, access these
; images as multiple frames. If the file contains
; only one image, access just that image.
IF (vNumFrames GT 1) THEN BEGIN
    vPixelData = MAKE_ARRAY(vCols, vRows, vNumFrames, $
        TYPE = vPixelType)
    FOR vIndex = 0L, (vNumFrames - 1) DO $
        vPixelData[*, *, vIndex] = READ_IMAGE(sFile, $
            IMAGE_INDEX = vIndex)
ENDIF ELSE BEGIN
    vPixelData = READ_IMAGE(sFile)
ENDELSE

; Create a new DICOM file and set properties.
oImg = OBJ_NEW('IDLffDicomEx', $
    path+PATH_SEP()+ 'aNewMonoImg.dcm', $
    SOP_CLASS = 'STANDARD_MR', /NON_CONFORMING, /CREATE)

; Call set pixel data with only required properties.
oImg->SetPixelData, vPixelData, $
    BITS_ALLOCATED = vBitsAlloc, $
    COLUMNS = vCols, $
    NUMBER_OF_FRAMES = vNumFrames, $
    PHOTOMETRIC_INTERPRETATION = vPhotoInterp, $
    PIXEL_REPRESENTATION = vPixelRep, $
    ROWS = vRows, $
    SAMPLES_PER_PIXEL = vNumSamples, $
    /ORDER

; Commit the file.
oImg->Commit

; Display monochrome image (frames).
WINDOW, XSIZE=vcols, YSIZE=vrows, $
    TITLE = 'New Monochrome DICOM Image'

FOR i = 1, vNumFrames DO BEGIN
    TVSCL, vPixelData[*,*,i-1]

```

```

        WAIT, 1
    ENDFOR

ENDIF

; If it is an RGB image, determine interleaving.
IF (vNumSamples EQ 3) THEN BEGIN

    ; Determine the size of all the dimensions of the pixel
    ; data array.
    vDataSize = SIZE(vPixelData, /DIMENSIONS)
    ; Determine the planar configuration of the image.
    vInterleave = WHERE((vDataSize NE vCols) AND $
        (vDataSize NE vRows))

    ; Return if line interleaved (vCols,3,vRows)
    IF (vInterleave[0] EQ 1) THEN RETURN

    ; If pixel interleaved (3,vCols,vRows), set to 0.
    ; If planar interleaved (vCols,vRows,3), set to 1
    IF (vInterleave[0] EQ 0) THEN vPlanarConfig = 0 $
        ELSE vPlanarConfig = 1

    ; Set other properties for RGB images.
    vBitsAlloc = 8
    vPhotoInterp = 'RGB'
    vPixelRep = 0

    ; Use READ_IMAGE to access the image array.
    vPixelData = READ_IMAGE(sFile)

    ; Create a new DICOM file and set properties.
    oImg = OBJ_NEW('IDLffDicomEx', $
        path+PATH_SEP()+ 'aNewRBGImg.dcm', $
        SOP_CLASS = 'STANDARD_US', /NON_CONFORMING, /CREATE)

    ; Call set pixel data with required properties
    oImg->SetPixelData, vPixelData, $
        BITS_ALLOCATED = vBitsAlloc, $
        COLUMNS = vCols, $
        NUMBER_OF_FRAMES = vNumFrames, $
        PHOTOMETRIC_INTERPRETATION = vPhotoInterp, $
        PIXEL_REPRESENTATION = vPixelRep, $
        PLANAR_CONFIGURATION = vPlanarConfig, $
        ROWS = vRows, $
        SAMPLES_PER_PIXEL = vNumSamples, $
        /ORDER
    oImg->Commit

```

```

; Display RGB image.
WINDOW, XSIZE=vcols, YSIZE=vrows, TITLE = 'New RGB DICOM Image'

IF vPlanarConfig EQ 0 THEN vTrue = 1 ELSE vTrue = 3
TV, vImageData, TRUE = vTrue

ENDIF

; Clean up the object references.
OBJ_DESTROY, [oImg]

; Note: the following lines allow you to run the program
; multiple times without having to manually delete files.
; You cannot duplicate an existing file when creating or cloning
; a DICOM file.
FILE_DELETE, path+PATH_SEP()+ 'aNewMonoImg.dcm', /ALLOW_NONEXISTENT
FILE_DELETE, path+PATH_SEP()+ 'aNewRBGImg.dcm', /ALLOW_NONEXISTENT

END

```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::SetPrivateValue

[Syntax](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The `IDLffDicomEx::SetPrivateValue` procedure method allows you to add and alter private attributes including items contained in sequences. When modifying the value of an existing private tag that is contained in a sequence, you must supply a SEQID keyword value. Use the [IDLffDicomEx::AddPrivateSequence](#) method or the [IDLffDicomEx::GetPrivateValue](#) method to return the SEQID keyword value.

Note

Use [IDLffDicomEx::AddPrivateSequence](#) to create the sequence, and then call `SetPrivateValue`, using the returned SEQID from the `AddPrivateSequence` call, to add private attributes to the sequence.

This method allows you to:

- Modify existing private attributes, those that exist at the root level of a file, and those contained within sequences.
- Add a private attribute (with or without a value) to the root level of a file, or as an item in a sequence. Use the `VR` argument to determine the Value Representation of the value. The value passed in will be converted to the specified VR as shown in the “[VR Conversion Table](#)” section.
- Clear single or multiple values from an attribute at the root level or within a sequence using the `CLEAR` keyword. The private attribute will exist but will not have any associated value.
- Remove a single attribute from the root level or from within a sequence using the `REMOVE` keyword. When you specify the `REMOVE` keyword in conjunction with a private sequence attribute (`SQ`), this removes all attributes in the sequence, including all nested sequences and all repeating groups of tags. You can also remove a block of private tags using the `BLOCKREMOVE` keyword. Once any change has been committed, attempting to re-access a value that has been removed will fail.

Note

You must call the [IDLffDicomEx::Commit](#) method to write any changes to the DICOM file.

VR Conversion Table

A private tag can have a single value or a tag can have multiple values. Correspondingly, the *Value* argument consists of either a single value or an array of values. The *VR* argument determines the Value Representation of the associated value(s). The VR types that can be used in `SetPrivateValue` are listed in the following table. These are the same VR types described in “[Value Representations](#)” on page 372. When `SetPrivateValue` is called to add or modify an attribute value, the conversions listed in the following table are applied to the data values specified in the *Value* argument. This lets you pass in values of one type and if possible the values will be converted according to the *VR* argument.

Value Representation	Conversion
AE (Application Entity)	STRING
AS (Age String)	
CS (Code String)	
DA (Date)	
DS (Decimal String)	
DT (Date Time)	
IS (Integer String)	
LO (Long String)	
LT (Long Text)	
PN (Patient Name)	
SH (Short String)	
ST (Short Text)	
TM (Time)	
UI (Unique Identifier)	
UT (Unlimited Text)	
SS (Signed Short)	FIX
US (Unsigned Short)	UINT
SL (Signed Long)	LONG

Table 3-19: Conversion of Value Argument Per VR Argument

Value Representation	Conversion
UL (Unsigned Long) AT (Attribute Tag)	ULONG
FL (Floating Point Single)	FLOAT
FD (Floating Point Double)	DOUBLE
SQ (Sequence)	No conversion. SQ can only specified for removal. To add a sequence, use the AddSequence method.
OB (Other Byte)	No conversion.
OW (Other Word)	No conversion.
OF (Other Float)	FLT

Table 3-19: Conversion of Value Argument Per VR Argument (Continued)

Syntax

Obj->[IDLffDicomEx::] SetPrivateValue, *PrivateCode*, *Group*, *Element*, *VR* [, *Value*]
[, *SEQID*=integer] [, /CLEAR] [, /REMOVE] [, / BLOCKREMOVE]

Arguments

PrivateCode

A string identification code that identifies the private block of data. Within a given private group *PrivateCode* labels are stored sequentially in the element addresses ranging from '0010' to '00FF'. For example, the string value stored at DICOM tag address '0029,0010' is the *PrivateCode* for the block of data tagged at '0029,1000' - '0029,10FF'. The label stored at '0029,0011' would be the *PrivateCode* for the data in tags '0029,1100' - '0029,11 FF'.

Group

A string identifying the group tag number of the private attribute (the first four digits of a DICOM tag). This must be an odd number and in the form 'XXXX'. If this does not reference an existing sequence, then a new private sequence is created.

Element

A string identifying the last two digits of the element associated with the private attribute. This must be in the form 'XX'. Valid values are 10 - FF.

Note

The first two digits of the *Element* are implicit in the *PrivateCode* argument.

Note

This argument is ignored when the BLOCKREMOVE keyword is set. All private attributes associated with the block of attributes identified by the *Group* and *PrivateCode* arguments will be removed.

VR

A two-character string of the attribute, indicating the Value Representation of the supplied *Value* argument. This argument is required even when removing an attribute. When adding an attribute value, the data specified in the *Value* argument is converted to the data type defined by this argument. See the “[VR Conversion Table](#)” on page 192 for how values are converted. See “[Value Representations](#)” on page 372 for descriptive VR list.

Value

A private attribute can have a single value or multiple values. Set this argument to a single value or array of value(s) to store in the attribute as follows:

- Set a single value to a tag by specifying a single value for the tag being written. This value matches the VR type specified in the *VR* argument.
- Set multiple values into a tag by specifying an array of values for the tag being written; values in the array match the VR type specified in the *VR* argument.

Note

If the *Value* argument is null, the value of the tag being written is set to null (the tag will not have a value). When the REMOVE or BLOCKREMOVE keywords are set the *Value* argument is ignored.

Keywords

SEQID

Set this keyword only if setting the value of an attribute that exists within a sequence. Use this keyword to specify a sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the [IDLffDicomEx::AddPrivateSequence](#) or [IDLffDicomEx::GetPrivateValue](#) method.
- Set to 0 or do not specify this keyword to indicate the attribute exists at the root level of the DICOM file. This is the default.

CLEAR

Set this keyword to remove all values from the private attribute.

REMOVE

Set this keyword to remove the private attribute from the DICOM file. If the private attribute is a sequence then the sequence and all of the private attributes included in the sequence are removed.

BLOCKREMOVE

Set this keyword to remove an entire block of private attributes from the DICOM file. The private block is identified by the *PrivateCode* and *Group* arguments.

Note

When this keyword is set the *Element* and *Value* arguments are ignored.

Example

The following example adds a private attribute to the root level of the DICOM file, a private sequence, and two items in the private sequence. This example shows how to add private attributes but does not write the tags to the cloned file. To persist any changes, call the [IDLffDicomEx::Commit](#) method. The new private attributes are printed to the Output Log window. Then the CLEAR and BLOCKREMOVE keywords are used to clear the value of the multi-valued attribute and delete the group of attributes containing the private sequence.

```
PRO print_tags_doc, vTags, vTagCnt
```

```

; Format the output.
PRINT, FORMAT= $
  '(%"%3s, %2s, %-12s, %3s, %7s, %3s, %5s, %15s")', $
  'IDX', 'LVL', 'TAG', 'VR', 'LEN', 'CNT', 'SEQID', $
  'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

  ; If the item is nested within another item, indicate the
  ; level using > symbol.
  IF (vTags[xx].Level GT 0) THEN BEGIN
    vLvl = STRJOIN(REPLICATE('>',vTags[xx].Level))
    vtg = vLvl + vTags[xx].Tag
  ENDIF ELSE BEGIN
    vtg = vTags[xx].Tag
  ENDELSE

  ; If the tags are in a group, indicate this.
  IF (vTags[xx].GroupNum GT 0) THEN BEGIN
    PRINT, FORMAT='(%" %15s, %1d")', 'Group', vTags[xx].GroupNum
  ENDIF

  ; Print the fields of the structure.
  PRINT, FORMAT = $
    '(%"%3d, %2d, %-12s, %3s, %7d, %3d, %5d, %15s")', $
    xx, vTags[xx].Level, vtg, vTags[xx].VR, vTags[xx].Length, $
    vTags[xx].ValueCount, vTags[xx].SeqId, $
    vTags[xx].Value

ENDFOR

END

PRO dicom_setprivate_remove_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
  PATH=FILEPATH('', SUBDIRECTORY=['examples', 'data']), $
  TITLE='Select DICOM Patient File', FILTER='*.dcm', $
  GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
  CLONE=sfile)

; Add private tags. The following are hypothetical.
; Create a tag at the root level.

```

```
arr = [1, 2, 3, 4]
oImg->SetPrivateValue, 'Private Test', '0053', '12', 'SS', arr

; Create a sequence at the root level.
vSeqID = oImg->AddPrivateSequence('VOI Min,Max', '0055', '12')

; Add items to the sequence, specifying SQ identifier returned by
; AddPrivateSequence.
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '13', 'IS', '215', $
    SEQID=vSeqID
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '14', 'IS', '234', $
    SEQID=vSeqID

; Return and print a range including the new tags to the
; Output Log window.
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0053,0000', STOP_TAG='0057,0000')

print_tags_doc, vTags, vTagCnt

; Clear the values from the multi-valued private attribute.
oImg->SetPrivateValue, 'Private Test', '0053', '12', 'SS', arr, $
    /CLEAR

; Remove the block of private attributes containing the
; private sequence.
oImg->SetPrivateValue, 'VOI Min,Max', '0055', '12', 'LO', 'x', $
    /BLOCKREMOVE

; Print tag modifications.
PRINT, 'Modified tags'
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0053,0000', STOP_TAG='0057,0000')

print_tags_doc, vTags, vTagCnt

END
```

The following output is printed to the IDL Output Log window. Notice how using BLOCKREMOVE deletes all attributes with a group value of '0055', including the block length tag, (0055,0010).

```

IDX, LV, TAG, VR, LEN, CNT, SEQID, VALUE
0, 0, 0053,0010, LO, 12, 1, 121, Private Test
1, 0, 0053,1012, SS, 8, 4, 121, 1\2\3\4
2, 0, 0055,0010, LO, 11, 1, 121, VOI Min,Max
3, 0, 0055,1012, SQ, 1, 1, 121,
4, 1, >0055,0010, LO, 11, 1, 122, VOI Min,Max
5, 1, >0055,1013, IS, 3, 1, 122, 215
6, 1, >0055,1014, IS, 3, 1, 122, 234

Modified tags
IDX, LV, TAG, VR, LEN, CNT, SEQID, VALUE
0, 0, 0053,0010, LO, 12, 1, 121, Private Test
1, 0, 0053,1012, SS, 0, 1, 121,

```

Figure 3-7: Setting and Removing Private Attributes

Version History

6.1	Introduced
-----	------------

IDLffDicomEx:: SetProperty

[Syntax](#) | [Return Value](#) | [Arguments](#) | [Keywords](#) | [Example](#) | [Version History](#)

The IDLffDicomEx::SetProperty procedure method sets the value of a property or group of properties for the IDLffDicomEx object.

Syntax

```
Obj->[IDLffDicomEx::] SetProperty [, PROPERTY = value]
```

Return Value

None

Arguments

None

Keywords

Any property listed under “IDLffDicomEx Properties” on page 70 that contains the word “Yes” in the “Set” column of the properties table can be set using this method. To set the value of a property, specify the property name as a keyword set equal to the appropriate property value.

Example

The following code shows setting several properties on a new monochrome image object. These properties must be set prior to setting pixel data or while setting pixel data. See the “Examples” section of [IDLffDicomEx::SetPixelData](#) for a complete example.

Note

These property values must match the data of the pixel data array.

```
; Create a new DICOM file and set properties.
oImg = OBJ_NEW('IDLffDicomEx', $
    path+PATH_SEP()+ 'aNewMonoImg.dcm', $
    SOP_CLASS = 'STANDARD_MR', /NON_CONFORMING, /CREATE)

; Set only the required properties.
oImg->SetProperty, $
```

```
    BITS_ALLOCATED = 16, $
    COLUMNS = 256, $
    NUMBER_OF_FRAMES = 1, $
    PHOTOMETRIC_INTERPRETATION = 'MONOCHROME2', $
    PIXEL_REPRESENTATION = 0, $
    ROWS = 256, $
    SAMPLES_PER_PIXEL = 1

; Call set pixel data.
oImg->SetPixelData, vPixelData, /ORDER
```

Version History

6.1	Introduced
-----	------------

IDLffDicomEx::SetValue

[Syntax](#) | [Arguments](#) | [Keywords](#) | [Examples](#) | [Version History](#)

The `IDLffDicomEx::SetValue` procedure method allows you to add and alter attributes including items contained in sequences. When modifying the value of an existing tag that is contained in a sequence, you must supply a SEQID keyword value. Use the [IDLffDicomEx::AddSequence](#) method or the [IDLffDicomEx::GetValue](#) method to return the SEQID keyword value.

Note

Use [IDLffDicomEx::AddSequence](#) to create the sequence, and then call `SetValue`, using the returned SEQID from the `AddSequence` call, to add attributes to the sequence.

This method allows you to:

- Modify existing attributes, those that exist at the root level of a file, and those contained within sequences.
- Add an attribute to the root level of a file, or as an item in a sequence. Use the `VR` argument to assign the Value Representation of the value. The value passed in will be converted to the specified VR as shown in the following [VR Conversion Table](#) section.
- Clear single or multiple values from an attribute at the root level or contained in a sequence using the `CLEAR` keyword. The attribute will exist but will not have any associated value.
- Remove a single attribute from the root level or from within a sequence using the `REMOVE` keyword. When you specify the `REMOVE` keyword in conjunction with a sequence attribute (`SQ`), this removes all attributes in the sequence, including all nested sequences and all repeating groups of tags. Once any change has been committed, attempting to re-access a value that has been removed will fail.

Note

You must call the [IDLffDicomEx::Commit](#) method to write any changes to the DICOM file.

Note

Always use the [IDLffDicomEx::SetPixelData](#) method to alter pixel data. Do not use the `SetValue` method.

Note

To ensure the pixel data is stored in the correct format before being further modified, use the [IDLffDicomEx::ChangeTransferSyntax](#) method to modify the compression of the pixel data. Do not use `SetValue` to directly modify the Transfer Syntax UID attribute (0002,0010).

VR Conversion Table

A tag can have a single value or a tag can have multiple values. Correspondingly, the *Value* argument consists of either a single value or an array of values. The *VR* argument determines the Value Representation of the associated value(s). The VR types that can be used in `SetPrivateValue` are listed in the following table. These are the same VR types described in “[Value Representations](#)” on page 372. When `SetValue` is called to add or modify an attribute value, the conversions listed in the following table are applied to the data values specified in the *Value* argument. This

lets you pass in values of one type and if possible the values will be converted according to the VR argument.

Value Representation	Conversion
AE (Application Entity)	STRING
AS (Age String)	
CS (Code String)	
DA (Date)	
DS (Decimal String)	
DT (Date Time)	
IS (Integer String)	
LO (Long String)	
LT (Long Text)	
PN (Patient Name)	
SH (Short String)	
ST (Short Text)	
TM (Time)	
UI (Unique Identifier)	
UT (Unlimited Text)	
SS (Signed Short)	FIX
US (Unsigned Short)	UINT
SL (Signed Long)	LONG
UL (Unsigned Long)	ULONG
AT (Attribute Tag)	
FL (Floating Point Single)	FLOAT
FD (Floating Point Double)	DOUBLE

Table 3-20: Conversion of Value Argument Per VR Argument

Value Representation	Conversion
SQ (Sequence)	No conversion. SQ can only specified for removal. To add a sequence, use the AddSequence method.
OB (Other Byte)	No conversion.
OW (Other Word)	No conversion.
OF (Other Float)	FLT

Table 3-20: Conversion of Value Argument Per VR Argument (Continued)

Syntax

```
Obj->[IDLffDicomEx::]SetValue, DicomTag, VR, Value [, SEQID=integer]
    [, /CLEAR] [, /REMOVE]
```

Note

The *VR* and *Value* arguments are optional when the CLEAR or REMOVE keywords are set.

Arguments

DicomTag

A string that identifies the group and element of a DICOM attribute in the form 'XXXX, XXXX'. The *DicomTag* argument must reference a public tag. See “DICOM Attributes” on page 300 for a list of tags.

Note

When adding a DICOM attribute, the tag must be part of the standard IOD for the image type unless the IDLffDicomEx object was initialized with the NON_CONFORMING keyword. Attempting to set an attribute that does not belong to the image type will result in an error. See [IDLffDicomEx::Init](#) for details.

VR

A two-character string of the attribute, indicating the Value Representation of the supplied *Value* argument. When adding an attribute value, the data specified in the *Value* argument is converted to the data type defined by this argument. See the “[VR Conversion Table](#)” on page 192 for how values are converted. See “[Value Representations](#)” on page 372 for a descriptive VR list.

Note

The *VR* argument is optional when the REMOVE or CLEAR keyword is set.

Value

An attribute can have a single value or multiple values (only a subset of standard DICOM attributes support multiple values). Set this argument to a single value or array of value(s) to store in the attribute as follows:

- Set a single value to a tag by specifying a single value for the tag being written. This value is converted to match VR type specified in the *VR* argument if it is not of the specified type.
- Set multiple values into a tag by specifying an array of values for the tag being written. Values in the array are converted match the VR type specified in the *VR* argument if they are not of the specified type.

Note

See the “[VR Conversion Table](#)” on page 192 for the conversions used per VR type.

Note

The *Value* argument is optional when the CLEAR or REMOVE keyword is set.

Keywords

SEQID

Set this keyword only if setting the value of an attribute that exists within a sequence. Use this keyword to specify a sequence identifier as follows:

- Set to a non-zero value (a sequence identifier) indicating the sequence in which the value is contained. This sequence identifier may have been returned via a previous call to the [IDLffDicomEx::AddSequence](#) or [IDLffDicomEx::GetValue](#) method.

- Set to 0 or do not set this keyword to indicate the attribute exists at the root level of the DICOM file. This is the default.

CLEAR

Set this keyword to remove all values from the attribute.

Note

Some attributes require one or more values in order to be valid. You should always replace any cleared mandatory values if you wish to maintain a valid DICOM file.

REMOVE

Set this keyword to remove the attribute from the DICOM file. If the attribute is a sequence then the sequence and all of the attributes included in the sequence are removed.

Note

Some attributes are required in a valid DICOM file. You should always replace any mandatory attributes that you remove if you wish to maintain a valid DICOM file.

Examples

Adding Attributes

The following code provides examples of:

- Adding attributes to the root level of a selected DICOM file
- Adding a sequence
- Adding attributes to the root-level sequence
- Adding a sequence nested inside the first sequence
- Adding attributes inside the nested sequence

The NON_CONFORMING keyword is set when the clone is created in order to avoid errors when attempting to add non-standard attributes to the selected DICOM file. The newly added attributes are printed to the IDL Output Log window.

Note

For an example that adds groups of repeating tags to a sequence, see the “Examples” section of “IDLffDicomEx::AddGroup” on page 86.

Note

This example does not write the cloned file to memory. To do so, simply use the `IDLffDicomEx::Commit` method.

```

PRO dicom_addpublicattributes_doc

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('',SUBDIRECTORY=['examples','data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
; Set the NON_CONFORMING keyword to be able to add a public SQ
; of radiopharmaceutical items to any file.
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile, /NON_CONFORMING)

; Add a root-level sequence (Radiopharmaceutical Information).
; *****
vRootSeq = oImg->AddSequence('0054,0016')

; Add an attribute within the sequence.
; *****
oImg->SetValue, '0018,1071', 'DS', '0', SEQID=vRootSeq

; Add a nested sequence (Radionuclide Code Sequence).
; *****
vNestSeq = oImg->AddSequence('0054,0300', PARENTSEQID=vRootSeq)

; Add two items to the nested sequence.
; *****
oImg->SetValue, '0008,0100', 'SH', 'Tc-99m', SEQID=vNestSeq
oImg->SetValue, '0008,0102', 'SH', '99SDM', SEQID=vNestSeq

; Print a range including the new tags to
; the Output Log window.
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0054,0000', STOP_TAG='0056,0000')

; Format the output.
PRINT, FORMAT= $
    '(%%-12s, %3s, %5s, %31s, %10s)', $
    'TAG', 'VR', 'SEQID', $
    'DESCRIPTION', 'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

```

```

; If the item is nested within another item, indicate the
; level using > symbol.
IF (vTags[xx].Level GT 0) THEN BEGIN
    vLvl = STRJOIN(REPLICATE('>',vTags[xx].Level))
    vtg = vLvl + vTags[xx].Tag
ENDIF ELSE BEGIN
    vtg = vTags[xx].Tag
ENDELSE

; If the tags are in a group, indicate this.
IF (vTags[xx].GroupNum GT 0) THEN BEGIN
    PRINT, FORMAT='(%" %15s, %1d"', 'Group', vTags[xx].GroupNum
ENDIF

; Print the fields of the structure.
PRINT, FORMAT = $
'(%"-12s, %3s, %5d, %31s, %10s"', $
vtg, vTags[xx].VR, vTags[xx].SeqId, $
vTags[xx].Description, vTags[xx].Value

ENDFOR

; Clean up references.
OBJ_DESTROY, oImg

END

```

This code produces the following output.

TAG	VR	SEQID	DESCRIPTION	VALUE
0054,0016	SQ	123	Radiopharmaceutical Information	
>0018,1071	DS	124	Radiopharmaceutical Volume	0
>0054,0300	SQ	124	Radionuclide Code Sequence	
>>0008,0100	SH	125	Code Value	Tc-99m
>>0008,0102	SH	125	Coding Scheme Designator	99SDM

Removing Attributes

The following example clears the value of a root-level attribute, deletes a nested sequence (and all of its items) and modifies the value of another item within a sequence. The output of the additions and modifications are printed to the Output Log window.

```

PRO print_tags_doc, vTags, vTagCnt

; Format the output.
PRINT, FORMAT= $
'(%" %3s, %2s, %-12s, %3s, %7s, %3s, %5s, %30s, %10s"', $

```



```

        'IDX', 'LVL', 'TAG', 'VR', 'LEN', 'CNT', 'SEQID', $
        'DESCRIPTION', 'VALUE'

; Cycle through the tags.
FOR xx = 0, vTagCnt-1 DO BEGIN

    ; If the item is nested within another item, indicate the
    ; level using > symbol.
    IF (vTags[xx].Level GT 0) THEN BEGIN
        vLvl = STRJOIN(REPLICATE('>',vTags[xx].Level))
        vtg = vLvl + vTags[xx].Tag
    ENDIF ELSE BEGIN
        vtg = vTags[xx].Tag
    ENDELSE

    ; If the tags are in a group, indicate this.
    IF (vTags[xx].GroupNum GT 0) THEN BEGIN
        PRINT, FORMAT='(%" %15s, %1d")', 'Group', vTags[xx].GroupNum
    ENDIF

; Print the fields of the structure.
PRINT, FORMAT = $
    '(%" %3d, %2d, %-12s, %3s, %7d, %3d, %5d, %30s, %10s")', $
    xx, vTags[xx].Level, vtg, vTags[xx].VR, vTags[xx].Length, $
    vTags[xx].ValueCount, vTags[xx].SeqId, $
    vTags[xx].Description, vTags[xx].Value

ENDFOR

END

PRO dicom_clearpublicattributes_doc
; Add and modify public attributes within a DICOM file.

; Select a DICOM file.
sFile = DIALOG_PICKFILE( $
    PATH=FILEPATH('',SUBDIRECTORY=['examples','data']), $
    TITLE='Select DICOM Patient File', FILTER='*.dcm', $
    GET_PATH=path)

; Create a clone (aImgClone.dcm) of the selected file (sfile).
; Set the NON_CONFORMING keyword to be able to add a public SQ
; of radiopharmaceutical items to any file.
oImg = OBJ_NEW('IDLffDicomEx', path + 'aImgClone.dcm', $
    CLONE=sfile, /NON_CONFORMING)

; Add a public attribute, Image ID to the root level of the file.
; *****

```

```

oImg->SetValue, '0054,0400', 'SH', 32

; Add a root-level sequence (Radiopharmaceutical Information).
; *****
vRootSeq = oImg->AddSequence('0054,0016')

; Add an attribute within the sequence.
oImg->SetValue, '0018,1071', 'DS', '0', SEQID=vRootSeq

; Add a nested sequence (Radionuclide Code Sequence).
vNestSeq = oImg->AddSequence('0054,0300', PARENTSEQID=vRootSeq)

; Add two items to the nested sequence.
oImg->SetValue, '0008,0100', 'SH', 'Tc-99m', SEQID=vNestSeq
oImg->SetValue, '0008,0102', 'SH', '99SDM', SEQID=vNestSeq

; Print a range including the new tags to
; the Output Log window.
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0054,0000', STOP_TAG='0056,0000')
print_tags_doc, vTags, vTagCnt

; ***** Clear Values *****
; Clear the values from an attribute at the root level.
oImg->SetValue, '0054,0400', /CLEAR

; Retrieve the root-level sequence identifier to modify
; items within the sequence.
vSeqId = oImg->GetValue('0054,0016')

; Remove the nested sequence. This also removes all attributes
; contained within the sequence.
oImg->SetValue, '0054,0300', SEQID=vSeqId, /REMOVE

; Change the value of Radiopharmaceutical Volume from 0 to 55.
oImg->SetValue, '0018,1071', 'DS', 55, SEQID=vSeqId

; Print a range including the new tags to
; the Output Log window.
PRINT, '***** Modified Values *****'
vTags = oImg->EnumerateTags(COUNT=vTagCnt, $
    START_TAG='0054,0000', STOP_TAG='0056,0000')
print_tags_doc, vTags, vTagCnt

; Cleanup objects.
OBJ_DESTROY, oImg
END

```

Running this program produces the following output. the Volume attribute value is changed from 0 to 55, the Radionuclide Code sequence and all member items have been removed, and the Image ID value has been cleared.

```

IDX, LV, TAG          , VR, SEQID, DESCRIPTION, VALUE
0, 0, 0054,0016      , SQ, 132, Radiopharmaceutical Information,
1, 1, >0018,1071    , DS, 133, Radiopharmaceutical Volume,      0
2, 1, >0054,0300    , SQ, 133, Radionuclide Code Sequence,
3, 2, >>0008,0100   , SH, 134, Code Value, Tc-99m
4, 2, >>0008,0102   , SH, 134, Coding Scheme Designator, 99SDM
5, 0, 0054,0400     , SH, 132, Image ID, 32

|***** Modified Values *****
IDX, LV, TAG          , VR, SEQID, DESCRIPTION, VALUE
0, 0, 0054,0016      , SQ, 132, Radiopharmaceutical Information,
1, 1, >0018,1071    , DS, 133, Radiopharmaceutical Volume,      55
2, 0, 0054,0400     , SH, 132, Image ID,

```

Figure 3-8: Setting and Modifying Public Attributes

Version History

6.1	Introduced
-----	------------

IDLffDicomExCfg

The IDLffDicomExCfg object allows you to set and retrieve the values of IDL DICOM Network Service configuration parameters. For an introduction to IDL's DICOM Network Service feature, see [Chapter 2, "Using IDL DICOM Network Services"](#).

The values accessible to this object can also be modified using the graphical user interface provided by the DICOM Network Services utility. See ["Starting the Network Services Utility"](#) on page 18 for a description of that utility. Parameters accessible through this object correspond to those shown on the **Configuration** tab of the DICOM Network Services utility.

Note

This feature requires an additional-cost license key to access the functionality. For more information, contact your ITT Visual Information Solutions sales representative or technical support.

Warning

Although configuration parameters for the DICOM Network Service are stored in human-readable XML files, it is important to note that the format of the configuration file may change at any time. Because of this, configuration files should always be modified either by using the graphical DICOM Network Services utility or using an instance of the IDLffDicomExCfg object, and never directly by the user.

Superclasses

None

Creation

See ["IDLffDicomExCfg::Init"](#) on page 227.

Properties

Objects of this class have no properties.

Methods

This class has the following methods:

- [IDLffDicomExCfg::Cleanup](#)
- [IDLffDicomExCfg::Commit](#)
- [IDLffDicomExCfg::Echo](#)
- [IDLffDicomExCfg::GetApplicationEntities](#)
- [IDLffDicomExCfg::GetApplicationEntity](#)
- [IDLffDicomExCfg::GetServiceLists](#)
- [IDLffDicomExCfg::GetServiceTypes](#)
- [IDLffDicomExCfg::GetValue](#)
- [IDLffDicomExCfg::Init](#)
- [IDLffDicomExCfg::IsDirty](#)
- [IDLffDicomExCfg::RemoveApplicationEntity](#)
- [IDLffDicomExCfg::SetApplicationEntity](#)
- [IDLffDicomExCfg::SetValue](#)
- [IDLffDicomExCfg::StorageScpService](#)

In addition, this class inherits the methods of its superclasses (if any).

Examples

See the Example sections of the following methods for examples using the `IDLffDicomExCfg` object:

- [IDLffDicomExCfg::Init](#)
- [IDLffDicomExQuery::Query](#)
- [IDLffDicomExQuery::Retrieve](#)
- [IDLffDicomExStorScu::Send](#)

In addition, the DICOM Network Services utility, which is included in the IDL distribution, includes the `cw_dicomex_config.pro` routine, which makes extensive use of this object. The `cw_dicomex_config.pro` routine is located in the `lib/dicomex` subdirectory of the IDL distribution.

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg Properties

Objects of this class have no properties.

IDLffDicomExCfg::Cleanup

The IDLffDicomExCfg::Cleanup procedure method performs all cleanup on the object, including closing the associated configuration file.

Note

Cleanup methods are special *lifecycle methods*, and as such cannot be called outside the context of object creation and destruction. In most cases, you cannot call the Cleanup method directly. However, one exception to this rule does exist. If you write your own subclass of this class, you can call the Cleanup method from within the Init or Cleanup method of the subclass.

Syntax

OBJ_DESTROY, *Obj*

or

Obj->[IDLffDicomExCfg::]Cleanup() *(In a lifecycle method only.)*

Arguments

None

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::Commit

The IDLffDicomExCfg::Commit procedure method commits changes made to the object and saves them to the underlying configuration file. Changes are not applied to the current configuration or saved in the configuration file until this method is called.

Syntax

Obj->[IDLffDicomExCfg::]Commit

Arguments

None

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::Echo

The IDLffDicomExCfg::Echo function method lets you test the network connection to a remote machine that supports Query SCP or Storage SCP service types. See [“Returning Connection Status with Echo”](#) on page 48 for additional information.

Syntax

```
Result = Obj->[IDLffDicomExCfg:]Echo(ApplicationEntityName
[, COUNT=variable])
```

Return Value

An array of strings containing the results of the echo. The number of elements and information returned depends on whether the echo succeeded.

	Contents	Example Value
0	Success value	Echo Succeeded:
1	Application Entity Name	AE Name: (StorScp)
2	Application Entity Title	AE Title: (IDL_STORE_SCP)
3	Host Name	Host Name: (10.17.2.59)
4	IP Address	IP Address: (10.17.2.59)
5	TCP/IP Port	TCP/IP Port: (104)
6	Implementation Version	Implementation Version: (MergeCOM3_321)
7	Implementation Class UID	Implementation Class UID: (2.16.840.1.114226)

Table 3-21: String Array Elements Returned by a Successful Echo

	Contents	Example Value
0	Success value	Echo Failed:
1	Application Entity Name	AE Name: (StorScp)

Table 3-22: String Array Elements Returned by a Failed Echo

	Contents	Example Value
2	Application Entity Title	AE Title: (IDL_STORE_SCP)
3	Reason	Open Association Failed (Failed to connect to remote host)

Table 3-22: String Array Elements Returned by a Failed Echo (Continued)

Arguments

ApplicationEntityName

A string containing the name of a defined Application Entity.

Keywords

COUNT

Set this keyword equal to a named variable that will contain an integer representing the number of strings in the returned string array. COUNT will be 8 when the echo succeeds or 4 when the echo fails.

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::GetApplicationEntities

The IDLffDicomExCfg::GetApplicationEntities function method returns an array of structures containing the definitions of all the application entities defined or all the application entities of a specified service type.

Syntax

```
Result = Obj->[IDLffDicomExCfg::]GetApplicationEntities( [, COUNT=variable]  
[, /SERVICE_TYPE=string] )
```

Return Value

An array of IDLFFDICOMEXCFGAE structures, one per existing Application Entity. Each IDLFFDICOMEXCFGAE structure has the following fields:

```
APPLENTITYNAME (string)  
AET (string)  
PORT (unsigned long integer)  
HOSTNAME (string)  
SERVICELISTNAME (string)  
SERVICETYPE (string)
```

Arguments

None

Keywords

COUNT

Set this keyword equal to a named variable that will contain an integer representing the number of structures representing application entities returned.

SERVICE_TYPE

Set this keyword equal to a string containing the name of an existing service type to return only application entities that match the service type. See [IDLffDicomExCfg::GetServiceTypes](#) for a list of allowed values.

Note

The value of the SERVICE_TYPE keyword is case-sensitive.

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::GetApplicationEntity

The IDLffDicomExCfg::GetApplicationEntity function method returns a structure containing the parameters for a specific Application Entity.

Syntax

Result = Obj->[IDLffDicomExCfg:]GetApplicationEntity(*ApplicationEntityName*)

Return Value

An IDLFFDICOMEXCFGAE structure representing the specified Application Entity. Each IDLFFDICOMEXCFGAE structure has the following fields:

```
APPLENTITYNAME (string)
AET (string)
PORT (unsigned long integer)
HOSTNAME (string)
SERVICELISTNAME (string)
SERVICETYPE (string)
```

Arguments

ApplicationEntityName

A string containing the name of one of the defined application entities.

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::GetServiceLists

The IDLffDicomExCfg::GetServiceLists function method returns an array of strings containing the names of the service lists that can be assigned to an Application Entity.

Syntax

Result = *Obj*->[IDLffDicomExCfg::]GetServiceLists([, COUNT=*variable*])

Return Value

An array of strings containing the names of the service lists that can be assigned to an Application Entity. Possible values are:

```
Echo_SCU_Service_List
Storage_SCP_Service_List
Storage_SCU_Service_List
Query_SCU_Service_List
Query_SCP_Service_List
```

Arguments

None

Keywords

COUNT

Set this keyword equal to a named variable that will contain an integer representing the number of service list names returned.

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::GetServiceTypes

The IDLffDicomExCfg::GetServiceTypes function method returns an array of strings containing the list of service types that can be assigned to an Application Entity.

Syntax

Result = *Obj*->[IDLffDicomExCfg:]GetServiceTypes([, COUNT=*variable*])

Return Value

An array of strings containing the names of the service types that can be assigned to an Application Entity. Possible values are:

```
Query_SCU
Query_SCP
Storage_SCU
Storage_SCP
Echo_SCU
```

Arguments

None

Keywords

COUNT

Set this keyword equal to a named variable that will contain an integer representing the number of service types returned.

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::GetValue

The IDLffDicomExCfg::GetValue function method returns the value for the specified parameter.

Syntax

Result = *Obj*->[IDLffDicomExCfg::]GetValue(*ConfigurationParameterName*)

Return Value

A string containing the value of the requested parameter.

Arguments

ConfigurationParameterName

One of the following strings:

String	Returned Value
QRScuServiceAE	The Application Entity name for the Query/Retrieve service.
StorScpServiceAE	The Application Entity name for the Storage SCP service.
StorScuServiceAE	The Application Entity name for the Storage SCU service.
EchoScuServiceAE	The Application Entity name for the Echo SCU service.
MaxQueryResponses	The number of responses accepted by the Query/Retrieve service.
StorScpDir	The directory where the Storage SCP service writes received DICOM files.
AcceptAny	Yes = The Storage SCP service will only accept associations from Application Entities defined in the system configuration file. No = The Storage SCP service will accept all associations.

String	Returned Value
FileExtension	The file extension used by the Storage SCP service when writing received files to disk (a three-character string).
StorScpCtrlPort	The port at which the Storage SCP service listens for control messages.

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::Init

The `IDLffDicomExCfg::Init` function method initializes the `IDLffDicomExCfg` object. The `IDLffDicomExCfg` object allows you to set and retrieve the values of IDL DICOM Network Service configuration parameters. For an introduction to IDL's DICOM Network Service feature, see [Chapter 2, "Using IDL DICOM Network Services"](#).

By default, the `IDLffDicomExCfg::Init` method creates an association between the returned object and the user's local DICOM Network Services configuration file. Use the `SYSTEM` keyword to create an object associated with the system-wide configuration file. See ["Local Versus System Configuration"](#) on page 18 for additional information on the difference between the system and local configurations.

Note

Init methods are special lifecycle methods, and as such cannot be called outside the context of object creation. This means that in most cases, you cannot call the `Init` method directly. There is one exception to this rule: if you write your own subclass of this class, you can call the `Init` method from within the `Init` method of the subclass.

Syntax

```
Obj = OBJ_NEW('IDLffDicomExCfg' [, /SYSTEM])
```

or

```
Result = Obj->[IDLffDicomExCfg::]Init( [, /SYSTEM] )  
(In a lifecycle method only.)
```

Return Value

When this method is called indirectly, as part of the call to the `OBJ_NEW` function, the return value is an object reference to the newly-created object.

When called directly within a subclass `Init` method, the return value is 1 if initialization was successful, or zero otherwise.

Note

This feature requires an additional-cost license key to access the functionality. If a valid feature license can not be found, an error is returned. For more information,

contact your ITT Visual Information Solutions sales representative or technical support.

Arguments

None

Keywords

SYSTEM

Set this keyword to create an IDLffDicomExCfg object associated with the DICOM Network Services system configuration file. By default, the created object is associated with the user's local configuration file.

Example

The following example code shows how to write and read a configuration file.

Note

Executing these commands will change the directory used by your local Storage SCP service. If you have already configured this value, be sure to change it back to your original value before retrieving files.

```
; Create a new object associated with the system configuration file.
ocfg = OBJ_NEW('IDLffDicomExCfg', /SYSTEM)

; Stop the Storage SCP Service before making changes.
status = ocfg->StorageScpService('stop')
PRINT, 'Stopping Storage SCP. Status: ', status
WAIT, 5

; Add a Storage SCP Application Entity.
ocfg->SetApplicationEntity, 'my_stor_scp_aen', $
    'my_stor_scp', 'my_machine_name', 2510, $
    'Storage_SCP_Service_List', 'Storage_SCP'

; Set the Storage SCP service to the entry created above.
ocfg->SetValue, 'StorScpServiceAE', 'my_stor_scp_aen'

; Set the local IDL Storage SCP directory to the 'MyStorScpDir'
; directory under the user's HOME directory.
myStorScpDir = GETENV('HOME') + PATH_SEP() + 'MyStorScpDir'
ocfg->SetValue, 'StorScpDir', myStorScpDir
```

```
; Save the changes and write the configuration file back to disk.
ocfg->Commit

; Restart the Storage SCP Service.
status = ocfg->StorageScpService('start')
PRINT, 'Starting Storage SCP. Status: ', status
WAIT, 5
status = ocfg->StorageScpService('status')
PRINT, 'Storage SCP Status: ', status

; Retrieve the definition for the Application Entity
; created above.
defAE = ocfg->GetApplicationEntity('my_stor_scp_aen')
HELP, defAE, /STRUCTURE

; Retrieve the name of the Application Entity for the
; Storage SCP service.
serviceAE = ocfg->GetValue('StorScpServiceAE')
PRINT, serviceAE

; Retrieve the directory to be used by the Storage SCP service.
scpDir = ocfg->GetValue('StorScpDir')
PRINT, scpDir

; Destroy the configuration object.
OBJ_DESTROY, ocfg
```

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::IsDirty

The IDLffDicomExCfg::IsDirty function method can be used to determine if configuration values have been changed since the configuration file was read into the configuration object. This information is useful if you are creating a user interface and want to warn the user to save the configuration values prior to exiting.

Syntax

Result = *Obj*->[IDLffDicomExCfg::]IsDirty()

Return Value

Returns one of the following:

0	The configuration in memory has not been changed since it was read in from the configuration file.
1	The configuration in memory has been changed since it was read in from the configuration file.

Arguments

None

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::RemoveApplicationEntity

The IDLffDicomExCfg::RemoveApplicationEntity procedure method marks a specified Application Entity for removal from the collection of application entities.

Note

The Application Entity is not actually removed from the collection until the [IDLffDicomExCfg::Commit](#) method is called.

Syntax

Obj->[IDLffDicomExCfg::]RemoveApplicationEntity, *ApplicationEntityName*

Arguments

ApplicationEntityName

A string containing the name of the Application Entity to be removed.

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::SetApplicationEntity

The IDLffDicomExCfg::SetApplicationEntity procedure method defines a new Application Entity or modifies the definition of an existing Application Entity.

Note

The Application Entity is not actually added or modified until the [IDLffDicomExCfg::Commit](#) method is called.

Syntax

Obj->[IDLffDicomExCfg::]SetApplicationEntity, *ApplicationEntityName*, *AET*,
Hostname, *Port*, *ServiceListName*, *ServiceType*

Arguments

ApplicationEntityName

A string containing the name of an Application Entity. If the specified Application Entity does not exist, it will be created. If the specified Application Entity already exists, it will be modified as specified by the other parameters.

ApplicationEntityName can have a maximum of 30 characters.

AET

A string containing the title of the Application Entity. The title need not be unique.

AET can have a maximum of 15 characters.

Hostname

A string containing a valid network computer name or a IP address. If the DICOM Network Service is running on the local machine, you can use the string `LocalHost`.

Hostname can have a maximum of 30 characters.

Port

A number or string containing the TCP/IP port number at which the Application Entity listens.

Port can have a maximum of 5 characters.

ServiceListName

A string containing a valid service list name. (See [IDLffDicomExCfg::GetServiceLists](#) for a list of valid strings.)

ServiceType

A string containing a valid service type. (See [IDLffDicomExCfg::GetServiceTypes](#) for a list of valid strings.)

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::SetValue

The IDLffDicomExCfg::SetValue procedure method sets the value for the specified parameter.

Note

The parameter value is not actually set until the [IDLffDicomExCfg::Commit](#) method is called.

Syntax

Obj->[IDLffDicomExCfg::]SetValue, *ConfigurationParameterName*, *Value*

Arguments

ConfigurationParameterName, Value

One of the following parameter name and value pairs:

Parameter Name	Parameter Value
QRScuServiceAE	A string containing the name of an existing Application Entity to be associated with the Query/Retrieve service. The maximum length is 30 characters.
StorScpServiceAE	A string containing the name of an existing Application Entity to be associated with the Storage SCP service. The maximum length is 30 characters.
StorScuServiceAE	A string containing the name of an existing Application Entity to be associated with the Storage SCU service. The maximum length is 30 characters.
EchoScuServiceAE	A string containing the name of an existing Application Entity to be associated with the Echo SCU service. The maximum length is 30 characters.
MaxQueryResponses	A string containing an integer specifying the maximum number of responses to return for a query. The maximum length is 5 characters.

Parameter Name	Parameter Value
StorScpDir	A string containing the name of the directory where the Storage SCP service writes received DICOM files. The maximum length is 256 characters.
AcceptAny	A string containing either Yes or No. The string Yes means the Storage SCP service will accept any association. The string No means the Storage SCP service will only accept associations from Application Entities defined in the system configuration file.
FileExtension	A string containing the file extension used by the Storage SCP service when writing received DICOM files to disk. The length should be exactly 3 characters.
StorScpCtrlPort	A string containing the an integer port number at which the Storage SCP service listens for control messages. The maximum length is 6 characters.

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExCfg::StorageScpService

The IDLffDicomExCfg::StorageScpService function method is used to start, stop or check the status of the local IDL Storage SCP service.

The local IDL Storage SCP service is a process that listens at a specified port for incoming DICOM files and writes the files to disk. On Microsoft Windows systems, the local IDL Storage SCP service is a Windows service. On UNIX systems, the local IDL Storage SCP service is a regular process. See [“About the Storage SCP Service”](#) on page 49 for additional information.

Syntax

Result = *Obj*->[IDLffDicomExCfg:]StorageScpService(*Command*)

Return Value

The return value is a string indicating the status of the Storage SCP service command specified by *Command*. The specific string issued depends on the type of system (Windows or UNIX), the *Command* specified (Start, Stop, or Status), and the state of the Storage SCP service.

Possible return values for the Start command:

Windows	UNIX
start request issued	start request issued
service already running	service already running
start access denied	
service logon failed	
service is disabled	
path not found	
start request failed	start failed
failed. administrator privileges maybe needed.	
failed, is IDL DicomEx Storage SCP service installed?	

Table 3-23: Return Values for the Start Command

Possible return values for the Stop command:

Windows	UNIX
stop request issued	stop request issued
service stopped	
service not running	service not running
stop access denied	
stop request failed	
	failed to send shutdown messages
failed. administrator privileges maybe needed.	
failed, is IDL DicomEx Storage SCP service installed?	

Table 3-24: Return Values for the Stop Command

Possible return values for the Status command:

Windows	UNIX
service running	service running
service not running	service not running
query access denied	
query status failed	
service does not exist	
service start pending	
service stop pending	
service pause pending	
service paused	
service status unknown	

Table 3-25: Return Values for the Status Command

Windows	UNIX
failed. administrator privileges maybe needed.	
failed, is IDL DicomEx Storage SCP service installed?	

Table 3-25: Return Values for the Status Command (Continued)

Arguments

Command

One of the following strings:

Command	Meaning
Start	Attempt to start the local IDL Storage SCP service if it is not already running.
Stop	Attempt to stop the local IDL Storage SCP service if it is running.
Status	Attempt to determine the state of the local IDL Storage SCP service (running or not running).

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExQuery

The IDLffDicomExQuery object allows you to query the contents of a remote DICOM node that has been configured to work with IDL's DICOM Network Service feature, and to retrieve DICOM files available on that node. See [“Defining a Machine to Be Queried”](#) on page 30 for an overview of the process and information about parameters that need to be set on the remote and local machines. For an introduction to IDL's DICOM Network Service feature, see [Chapter 2, “Using IDL DICOM Network Services”](#).

Any query that can be performed using this object can also be performed using the graphical user interface provided by the DICOM Network Services utility. See [“Querying a Remote Machine”](#) on page 33 for a description of that utility. Parameters accessible through this object correspond to those shown on the **Query Retrieve SCU** tab of the DICOM Network Services utility.

Note

This feature requires an additional-cost license key to access the functionality. For more information, contact your ITT Visual Information Solutions sales representative or technical support.

Superclasses

None

Creation

See [“IDLffDicomExQuery::Init”](#) on page 263.

Properties

The IDLffDicomExQuery object has the following properties:

- [ACCESSION_NUMBER](#)
- [CALLBACK_DATA](#)
- [CALLBACK_FUNCTION](#)
- [FAMILY_NAME](#)
- [GIVEN_NAME](#)
- [INSTANCE_NUMBER](#)
- [MIDDLE_NAME](#)
- [MODALITY](#)
- [PATIENT_ID](#)
- [PATIENT_NAME](#)

- PREFIX
- QUERY_MODEL
- SERIES_INSTANCE_UID
- SOP_INSTANCE_UID
- STUDY_DATE
- STUDY_INSTANCE_UID
- SUFFIX
- QUERY_LEVEL
- QUERY_SCP
- SERIES_NUMBER
- STORAGE_SCP
- STUDY_ID
- STUDY_TIME

See “IDLffDicomExQuery Properties” on page 242 for details on individual properties.

Methods

This class has the following methods:

- IDLffDicomExQuery::Cleanup
- IDLffDicomExQuery::ClearProperties
- IDLffDicomExQuery::GetProperty
- IDLffDicomExQuery::Init
- IDLffDicomExQuery::Query
- IDLffDicomExQuery::QueryModelsSupported
- IDLffDicomExQuery::Retrieve
- IDLffDicomExQuery::SetProperty

In addition, this class inherits the methods of its superclasses (if any).

Examples

See the Example sections of the following methods for examples using the IDLffDicomExQuery object:

- IDLffDicomExQuery::Query
- IDLffDicomExQuery::Retrieve

In addition, the DICOM Network Services utility, which is included in the IDL distribution, includes the `cw_dicomex_query.pro` routine, which makes extensive

use of this object. The `cw_dicomex_query.pro` routine is located in the `lib/dicomex` subdirectory of the IDL distribution.

Version History

6.3	Introduced
-----	------------

IDLffDicomExQuery Properties

IDLffDicomExQuery objects have the following properties. Properties with the word “Yes” in the “Get” column of the property table can be retrieved via [IDLffDicomExQuery::GetProperty](#). Properties with the word “Yes” in the “Init” column of the property table can be set via [IDLffDicomExQuery::Init](#). Properties with the word “Yes” in the “Set” column in the property table can be set via [IDLffDicomExQuery::SetProperty](#).

Note

For a discussion of the property description tables shown below, see [“About Object Property Descriptions”](#) on page 70. Tables for properties of this object do not include the **DICOM Attribute** or **Multi-value** fields.

There are two categories of properties for this object: *general* properties and *query field* properties.

- General properties are used to specify information about the process of querying for or retrieving data: the name of the Query SCP or the level at which the query is performed, for example.
- Query field properties are used to narrow the number of matches returned by a query: patient names and instance numbers are examples. Values of query fields can include wildcard characters: see [“Attribute Matching and Wildcards”](#) on page 254 for details.

Note

Depending on the implementation of the remote node being queried, query field property values may be case sensitive.

The IDLffDicomExQuery object has the following properties:

- [ACCESSION_NUMBER](#)
- [CALLBACK_DATA](#)
- [CALLBACK_FUNCTION](#)
- [FAMILY_NAME](#)
- [GIVEN_NAME](#)
- [INSTANCE_NUMBER](#)
- [MIDDLE_NAME](#)
- [MODALITY](#)
- [PATIENT_ID](#)
- [PATIENT_NAME](#)
- [PREFIX](#)
- [QUERY_LEVEL](#)
- [QUERY_MODEL](#)
- [QUERY_SCP](#)

- [SERIES_INSTANCE_UID](#)
- [SOP_INSTANCE_UID](#)
- [STUDY_DATE](#)
- [STUDY_INSTANCE_UID](#)
- [SUFFIX](#)
- [SERIES_NUMBER](#)
- [STORAGE_SCP](#)
- [STUDY_ID](#)
- [STUDY_TIME](#)

ACCESSION_NUMBER

This query field property represents the Accession Number attribute (0008,0050). If this property is not set, the query is issued with this property set to a null string.

This property is used by the `IDLffDicomExQuery::Query` method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: SH
Get: Yes	Set: Yes	Init: No	Registered: No

CALLBACK_DATA

This general property contains data that is to be passed to the function defined by the `CALLBACK_FUNCTION` property when a callback is initiated. If this property is not set, the integer value zero is passed to the callback function. See “[Using Callbacks with the IDLffDicomExQuery Object](#)” on page 258 for additional details.

This property is used by the `IDLffDicomExQuery::Query` and `IDLffDicomExQuery::Retrieve` methods.

Property Type	Any type		
Name String	<i>not displayed</i>		VR: <i>not applicable</i>
Get: Yes	Set: Yes	Init: No	Registered: No

CALLBACK_FUNCTION

This general property contains the name of an IDL function to be called by the `Query` and `Retrieve` methods. If no value is defined, no callback is initiated. See “[Using Callbacks with the IDLffDicomExQuery Object](#)” on page 258 for additional details.

This property is used by the [IDLffDicomExQuery::Query](#) and [IDLffDicomExQuery::Retrieve](#) methods.

Property Type	String		
Name String	<i>not displayed</i>		VR: <i>not applicable</i>
Get: Yes	Set: Yes	Init: No	Registered: No

FAMILY_NAME

This query field property represents the family name (last name) associated with a patient record.

Note

The Family_Name, Given_Name, Middle_Name, Prefix, and Suffix properties are combined to populate the DICOM Patient Name attribute (0010,0010). If the Patient_Name property is set, this property is ignored.

If a query uses the [PATIENT_NAME](#) property but no value is specified for either the Patient_Name property or this property, the query is issued with Family_Name set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: PN
Get: Yes	Set: Yes	Init: No	Registered: No

GIVEN_NAME

This query field property represents the given name (first name) associated with a patient record.

Note

The Family_Name, Given_Name, Middle_Name, Prefix, and Suffix properties are combined to populate the DICOM Patient Name attribute (0010,0010). If the Patient_Name property is set, this property is ignored.

If a query uses the [PATIENT_NAME](#) property but no value is specified for either the Patient_Name property or this property, the query is issued with Given_Name set to a null string.

This property is used by the `IDLffDicomExQuery::Query` method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: PN
Get: Yes	Set: Yes	Init: No	Registered: No

INSTANCE_NUMBER

This query field property represents the Instance Number attribute (0020,0013). If this property is not set, the query is issued with this property set to a null string.

This property is used by the `IDLffDicomExQuery::Query` method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: IS
Get: Yes	Set: Yes	Init: No	Registered: No

MIDDLE_NAME

This query field property represents the middle name associated with a patient record.

Note

The Family_Name, Given_Name, Middle_Name, Prefix, and Suffix properties are combined to populate the DICOM Patient Name attribute (0010,0010). If the Patient_Name property is set, this property is ignored.

If a query uses the [PATIENT_NAME](#) property but no value is specified for either the Patient_Name property or this property, the query is issued with Middle_Name set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: PN
Get: Yes	Set: Yes	Init: No	Registered: No

MODALITY

This query field property represents the Modality attribute (0008,0060). If this property is not set, the query is issued with this property set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: CS
Get: Yes	Set: Yes	Init: No	Registered: No

PATIENT_ID

This query field property represents the Patient ID attribute (0010,0020). If this property is not set and the query level is “patient” or “study,” the query is issued with this property set to “*”. If this property is not set and the query level is “series” or “image,” the query is issued with this property set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: LO
Get: Yes	Set: Yes	Init: No	Registered: No

PATIENT_NAME

This query field property represents the Patient ID attribute (0010,0010).

Note

The Family_Name, Given_Name, Middle_Name, Prefix, and Suffix properties are combined to populate the DICOM Patient Name attribute (0010,0010). If the Patient_Name property is set, all the other name properties are ignored.

The Patient_Name property string comprises the five name sub-properties, separated by the “^” character:

```
Family Name^Given Name^Middle Name^Prefix^Suffix
```

If any of the sub-properties are omitted, you must substitute an empty string in the correct location. For example, to specify a Patient_Name without a middle name:

```
Family Name^Given Name^^Prefix^Suffix
```

See “PN” on page 379 for details.

If a query uses this query field but no value is specified for this property, IDL constructs the value of the Patient_Name property from the values of the Family_Name, Given_Name, Middle_Name, Prefix, and Suffix properties.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: PN
Get: Yes	Set: Yes	Init: No	Registered: No

PREFIX

This query field property represents the title (or honorific) associated with a patient record.

Note

The Family_Name, Given_Name, Middle_Name, Prefix, and Suffix properties are combined to populate the DICOM Patient Name attribute (0010,0010). If the Patient_Name property is set, this property is ignored.

If a query uses the [PATIENT_NAME](#) property but no value is specified for either the Patient_Name property or this property, the query is issued with Prefix set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: PN
Get: Yes	Set: Yes	Init: No	Registered: No

QUERY_LEVEL

This general property sets the query level to be used in a query. Valid values are:

Value	Query Level
0	Patient
1	Study
2	Series
3	Image

Warning

The default value of zero is not a valid value if the `QUERY_MODEL` property is set equal to one. Attempting to set `QUERY_LEVEL=0` and `QUERY_MODEL=1` will generate a warning message, and `QUERY_LEVEL` will automatically be set equal to one.

Because the default value for `QUERY_LEVEL` is zero, you should set the value of this property *before* setting the value of `QUERY_MODEL` to avoid receiving the error message.

This property is used by the [IDLffDicomExQuery::Query](#) and [IDLffDicomExQuery::Retrieve](#) methods.

Property Type	Integer		
Name String	<i>not displayed</i>		VR: <i>not applicable</i>
Get: Yes	Set: Yes	Init: No	Registered: No

QUERY_MODEL

This general property sets the query model to be used in a query. Valid values are:

Value	Query Model
0	Patient Root
1	Study Root
2	Patient Study Only

If this property is not set, the query is issued with this property set to zero (Patient Root).

Warning

Attempting to set QUERY_LEVEL=0 and QUERY_MODEL=1 will generate a warning message, and QUERY_LEVEL will automatically be set equal to one.

This property is used by the [IDLffDicomExQuery::Query](#) and [IDLffDicomExQuery::Retrieve](#) methods.

Property Type	Integer		
Name String	<i>not displayed</i>		VR: <i>not applicable</i>
Get: Yes	Set: Yes	Init: No	Registered: No

QUERY_SCP

This general property specifies the Application Entity name of the node to be queried. The Application Entity name must be defined in the DICOM Network Services configuration file.

Note

This property *must* be set prior to calling the Query or Retrieve methods.

This property is used by the [IDLffDicomExQuery::Query](#), [IDLffDicomExQuery::QueryModelsSupported](#), and [IDLffDicomExQuery::Retrieve](#) methods.

Property Type	String		
Name String	<i>not displayed</i>		VR: <i>not applicable</i>
Get: Yes	Set: Yes	Init: No	Registered: No

SERIES_INSTANCE_UID

This query field property represents the Series Instance UID attribute (0020,000E). If this property is not set, the query is issued with this property set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: UI
Get: Yes	Set: Yes	Init: No	Registered: No

SERIES_NUMBER

This query field property represents the Series Number attribute (0020,0011). If this property is not set, the query is issued with this property set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: IS
Get: Yes	Set: Yes	Init: No	Registered: No

SOP_INSTANCE_UID

This query field property represents the SOP Instance UID attribute (0008,0018). If this property is not set, the query is issued with this property set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: UI
Get: Yes	Set: Yes	Init: No	Registered: No

STORAGE_SCP

This general property specifies the Application Entity name of the Storage SCP node that receives retrieved files. The Application Entity name must be defined in the DICOM Network Services configuration file.

Note

This property *must* be set prior to calling the Retrieve method.

This property is used by the [IDLffDicomExQuery::Retrieve](#) method.

Property Type	String		
Name String	<i>not displayed</i>		VR: <i>not applicable</i>
Get: Yes	Set: Yes	Init: No	Registered: No

STUDY_DATE

This query field property represents the Study Date attribute (0008,0020). If this property is not set, the query is issued with this property set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: DA
Get: Yes	Set: Yes	Init: No	Registered: No

STUDY_ID

This query field property represents the Study ID attribute (0020,0010). If this property is not set, the query is issued with this property set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: SH
Get: Yes	Set: Yes	Init: No	Registered: No

STUDY_INSTANCE_UID

This query field property represents the Study Instance UID attribute (0020,000D). If this property is not set, the query is issued with this property set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: UI
Get: Yes	Set: Yes	Init: No	Registered: No

STUDY_TIME

This query field property represents the Study Time attribute (0008,0030). If this property is not set, the query is issued with this property set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: TM
Get: Yes	Set: Yes	Init: No	Registered: No

SUFFIX

This query field property represents the name suffix (Jr. or Sr., for example) associated with a patient record.

Note

The Family_Name, Given_Name, Middle_Name, Prefix, and Suffix properties are combined to populate the DICOM Patient Name attribute (0010,0010). If the Patient_Name property is set, this property is ignored.

If a query uses the [PATIENT_NAME](#) property but no value is specified for either the Patient_Name property or this property, the query is issued with Suffix set to a null string.

This property is used by the [IDLffDicomExQuery::Query](#) method. See “[Query Property Interactions With Query Model/Level](#)” on page 256 to determine when this property can be used in a specific query.

Property Type	String		
Name String	<i>not displayed</i>		VR: PN
Get: Yes	Set: Yes	Init: No	Registered: No

Attribute Matching and Wildcards

The implementation of the Query/Retrieve SCP service on the remote node determines the quality of attribute matching. All attribute matching is performed by the remote service. In addition to case sensitivity, matching can include one or more of the following:

Single Value Matching	Matches single values.
Wildcard Matching	Use * to match zero or more characters. Use ? to match any single character. For example, to return patients with a name beginning with HA, use HA*. To return a list of patients whose names vary by one or more instances of a single value, use the ? character to indicate the wildcard value as in M?NROE.
Range Matching	Use the - character in between date or time values to return any matches within that range. For example, to match any date in the first six months of the year, use 20050101-20050630.
UID List Matching	Matches a list of UIDs separated by \ character.

The following table indicates the value representation (VR) and what wildcards are supported in each query field. A "•" in the column indicates the wildcard is supported.

Field	VR	*	?	-	/
Patient Name	PN	•	•		
Family Name	PN	•	•		
Given Name	PN	•	•		
Middle Name	PN	•	•		
Prefix	PN	•	•		
Suffix	PN	•	•		
Patient ID	LO	•	•		

Table 3-26: Wildcard Support for Query Fields

Field	VR	*	?	-	/
Study Instance UID	UI				•
Study ID	SH	•	•		
Study Date	DA			•	
Study Time	TM			•	
Accession Number	SH	•	•		
Series Instance UID	UI				•
Series Number	IS				
Modality	CS				
SOP Instance UID	UI				•
Instance Number	IS				

Table 3-26: Wildcard Support for Query Fields (Continued)

Query Property Interactions With Query Model/Level

Table 3-27 defines which query properties apply to different combinations of query model and query level. A "•" in the column indicates that the property is used in a query with the specified combination of query model and query level.

A "•" in the column also indicates that the corresponding field in the IDLFFDICOMEXQRESULTS structure returned by the `IDLffDicomExQuery::Query` method will contain a meaningful result value. If the table indicates that the field contains a meaningful value but the result is a null value, this means that the query returned a null value for that field.

	PATIENT_NAME	PATIENT_ID	STUDY_INSTANCE_UID	STUDY_ID	STUDY_DATE	STUDY_TIME	ACCESSION_NUMBER	SERIES_INSTANCE_UID	SERIES_NUMBER	MODALITY	SOP_INSTANCE_UID	INSTANCE_NUMBER	Study Description *	Series Description *
Patient Root / Patient Level	•	•												
Patient Root / Study Level		•	•	•	•	•	•						•	
Patient Root / Series Level		•	•					•	•	•			•	•
Patient Root / Image Level		•	•					•			•	•	•	•
Study Root / Study Level	•	•	•	•	•	•	•						•	
Study Root / Series Level	•	•	•					•	•	•			•	•
Study Root / Image Level	•	•	•					•			•	•	•	•

Table 3-27: Query Model/Level vs. Query Property Usage

	PATIENT_NAME	PATIENT_ID	STUDY_INSTANCE_UID	STUDY_ID	STUDY_DATE	STUDY_TIME	ACCESSION_NUMBER	SERIES_INSTANCE_UID	SERIES_NUMBER	MODALITY	SOP_INSTANCE_UID	INSTANCE_NUMBER	Study Description *	Series Description *
Patient Study Only Root	•	•	•	•	•	•	•						•	

* These structure fields cannot be used to narrow the query results, but they are included in the returned structure.

Table 3-27: Query Model/Level vs. Query Property Usage (Continued)

Patient Name and Name Sub-field Properties

The DICOM standard defines a Patient Name as being made up of five sub-fields:

- [FAMILY_NAME](#)
- [GIVEN_NAME](#)
- [MIDDLE_NAME](#)
- [PREFIX](#)
- [SUFFIX](#)

For each of these sub-fields of the Patient Name, the IDLffDicomExQuery object has an associated property. When creating a query that uses the [PATIENT_NAME](#) property, you can specify *either* the value of the Patient_Name property *or* the values of one or more of the five sub-properties.

- If you provide a value for the Patient_Name property, the values of the sub-properties are ignored.
- If you do not provide a value for the Patient_Name property, the values of any of the sub-properties that are set are used to generate a value for the Patient_Name property.

Using Callbacks with the IDLffDicomExQuery Object

Callbacks from the IDLffDicomExQuery object provide a way to transmit information back to the caller based on the status of a query or retrieve operation. The value returned by the callback function is then used by the IDLffDicomExQuery object to determine whether the query or retrieve operation should continue.

The IDLffDicomExQuery object allows you to define a single function (written in IDL) that will be called based on the intermediate status of a Query or Retrieve method call. If a callback function is specified via the [CALLBACK_FUNCTION](#) property, it is called in the following circumstances:

- For the [IDLffDicomExQuery::Query](#) method, the callback function is called each time a match is returned by the source node.
- For the [IDLffDicomExQuery::Retrieve](#) method:
 - If the source node (the remote Query/Retrieve SCP) implements the optional response mechanism, the callback function is called each time a DICOM file is moved from the source node to the destination node.
 - If the source node (the remote Query/Retrieve SCP) does not implement the optional response mechanism, the callback function is called every 3 seconds.

Both methods invoke the callback function with an array of strings indicating status as the first parameter, and with the value (if any) specified by the [CALLBACK_DATA](#) property as the second parameter.

If the return value of the callback function is zero, a cancel message will be sent to the remote node. If the return value is one, the query or retrieve operation will continue.

Callback Routine Signature

A query/retrieve callback function is written in IDL and has the following signature:

```
Function Callback_Function_Name, StatusInfo, CallbackData
```

where

- *Callback_Function_Name* is the name of the callback function. This value is specified as the value of the [CALLBACK_FUNCTION](#) property.
- *StatusInfo* is an array of strings that contain status information about the query or retrieve operation. The status strings are provided for information and human inspection rather than for programmatic processing; the exact strings

included in the array will depend on numerous factors including the status of the query or retrieve and the type of node being queried or retrieved from.

- *CallbackData* is the data specified via the `CALLBACK_DATA` property. The value of this property is passed to the callback function unmodified. If the `CALLBACK_DATA` property is unspecified, an integer zero is passed to the callback function as the value of this parameter.

The *CallbackData* parameter is useful for passing static information, such as the widget ID of a widget where the status information is displayed, from the `IDLffDicomExQuery` object to the callback routine.

The return value of the callback function should be an integer zero or one. If the return value is zero, a cancel message will be sent to the remote node. If the return value is one, the query or retrieve operation will continue.

IDLffDicomExQuery::Cleanup

The IDLffDicomExQuery::Cleanup procedure method performs all cleanup on the object.

Note

Cleanup methods are special *lifecycle methods*, and as such cannot be called outside the context of object creation and destruction. In most cases, you cannot call the Cleanup method directly. However, one exception to this rule does exist. If you write your own subclass of this class, you can call the Cleanup method from within the Init or Cleanup method of the subclass.

Syntax

OBJ_DESTROY, *Obj*

or

Obj->[IDLffDicomExQuery::]Cleanup() *(In a lifecycle method only.)*

Arguments

None

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExQuery::ClearProperties

The `IDLffDicomExQuery::ClearProperties` procedure method is used to reset the values of all properties of the `IDLffDicomExQuery` object to their default values.

Syntax

Obj->[\[IDLffDicomExQuery::\]ClearProperties](#)

Arguments

None

Keywords

None

Version History

6.3	Introduced
-----	------------

See Also

[IDLffDicomExQuery Properties](#)

IDLffDicomExQuery::GetProperty

The IDLffDicomExQuery::GetProperty procedure method retrieves the value of an IDLffDicomExQuery property.

Syntax

Obj->[IDLffDicomExQuery::]GetProperty[, *PROPERTY=variable*]

Arguments

None

Keywords

Any property listed under “IDLffDicomExQuery Properties” on page 242 that contains the word “Yes” in the “Get” column of the properties table can be retrieved using this method. To retrieve the value of a property, specify the property name as a keyword set equal to a named variable that will contain the value of the property.

Version History

6.3	Introduced
-----	------------

IDLffDicomExQuery::Init

The `IDLffDicomExQuery::Init` function method initializes the `IDLffDicomExQuery` object. The `IDLffDicomExQuery` object allows you query or retrieve DICOM files from a remote node configured as a Query SCP available in IDL's DICOM Network Service configuration. For an introduction to IDL's DICOM Network Service feature, see [Chapter 2, "Using IDL DICOM Network Services"](#).

Note

Init methods are special lifecycle methods, and as such cannot be called outside the context of object creation. This means that in most cases, you cannot call the `Init` method directly. There is one exception to this rule: if you write your own subclass of this class, you can call the `Init` method from within the `Init` method of the subclass.

Syntax

```
Obj = OBJ_NEW('IDLffDicomExQuery' )
```

or

```
Result = Obj->[IDLffDicomExQuery::]Init() (In a lifecycle method only.)
```

Return Value

When this method is called indirectly, as part of the call to the `OBJ_NEW` function, the return value is an object reference to the newly-created object.

When called directly within a subclass `Init` method, the return value is one if initialization was successful, or zero otherwise.

Arguments

None

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExQuery::Query

The `IDLffDicomExQuery::Query` function method is used to query a remote Query/Retrieve SCP Application Entity. A query message is constructed using the appropriate query field properties for the specified query model and query level; the query message is sent to the remote node, which sends a reply for each match. See “[Defining a Machine to Be Queried](#)” on page 30 for an overview of this process.

This method does not return to the caller until one of the following occurs:

- The query is complete (all matches have been returned)
- The number of responses defined by the `MaxQueryResponses` configuration parameter is reached
- The query encounters an error condition
- The callback function defined by the `CALLBACK_FUNCTION` property returns zero

Properties Required by a Query

The `QUERY_SCP` property must contain the name of a valid Application Entity that supports the Query SCP service in order for the Query method to succeed. All other properties used by the Query method have usable default values. If no properties other than `QUERY_SCP` are set, this method will perform a Patient Root / Patient Level query listing all patients on the remote node.

Properties Allowed by a Query

The query field properties that are used in a specific DICOM query are defined by the query model and query level chosen. For example, in a Patient Root / Patient Level query, only the values of the `Patient_Name` and `Patient_ID` properties are sent to the remote node; all other query field properties are ignored.

See “[IDLffDicomExQuery Properties](#)” on page 242 for a complete list of properties used to define a query or that affect the operation of the Query method.

See “[Attribute Matching and Wildcards](#)” on page 254 for a discussion of how to perform queries using wildcard values for query properties.

See “[Query Property Interactions With Query Model/Level](#)” on page 256 for a complete list of the interactions between query properties and the selected query model and level.

Callbacks and Query Status Information

If the [CALLBACK_FUNCTION](#) property contains the name of an IDL function, that function is called each time the remote node returns a match. The function is called with an array of strings containing status information about the query as its first parameter. See “[Using Callbacks with the IDLffDicomExQuery Object](#)” on page 258 for a discussion of callback functions.

Canceling a Query

To cancel a query before all matches are returned or the value of the [MaxQueryResponses](#) configuration parameter is reached, the callback function specified by the [CALLBACK_FUNCTION](#) property must return zero.

Syntax

Result = Obj->[IDLffDicomExQuery::]Query([, COUNT=*variable*])

Return Value

This method returns an array of IDLFFDICOMEXQRESULTS structures containing the information returned by the query, one structure per matched item. The maximum number of structures in the array is defined by the [MaxQueryResponses](#) configuration parameter.

Note

The COUNT keyword returns the number of items that match the query. Always check the number of matches before attempting to reference an element in the returned array.

The IDLFFDICOMEXQRESULTS structure is described below. With two exceptions, each field in the structure corresponds to a property of the IDLffDicomExQuery object. See the description of the property in “[IDLffDicomExQuery Properties](#)” on page 242 for information about the range of possible values returned.

The STUDY_DESCRIPTION and SERIES_DESCRIPTION fields do not correspond to query field properties. For selected queries, these fields will contain string values describing the study or series, if the descriptions are available.

Note that a field in an IDLFFDICOMEXQRESULTS structure can contain a null string if *either* of the following is true:

- No value for the field is returned by a particular combination of query model and query level. See “[Query Property Interactions With Query Model/Level](#)”

on page 256 for a discussion the fields for which values are returned for a given combination of query model and query level.

- The actual value of the field is a null string.

The IDLFFDICOMEXQRESULTS structure has the following fields:

```

QUERY_MODEL (integer)
QUERY_LEVEL (integer)
PATIENT_NAME (string)
FAMILY_NAME (string)
GIVEN_NAME (string)
MIDDLE_NAME (string)
PREFIX (string)
SUFFIX (string)
PATIENT_ID (string)
STUDY_INSTANCE_UID (string)
STUDY_ID (string)
STUDY_DATE (string)
STUDY_TIME (string)
ACCESSION_NUMBER (string)
SERIES_INSTANCE_UID (string)
SERIES_NUMBER (string)
MODALITY (string)
SOP_INSTANCE_UID (string)
INSTANCE_NUMBER (string)
STUDY_DESCRIPTION (string)
SERIES_DESCRIPTION (string)

```

Note on Missing UIDS

If a query returns values for a field that is *supposed* to contain a unique identifier but does not, this method substitutes one of the following strings in the corresponding field of the returned structure:

Field with Missing Data	Substituted String
Patient_ID	Unknown Patient Id
Study_Instance_UID	Unknown Study UID
Series_Instance_UID	Unknown Series UID
SOP_Instance_UID	Unknown Image UID

Arguments

None

Keywords

COUNT

Set this keyword equal to a named variable that will contain a long integer representing the number of `IDLFFDICOMEXQRESULTS` structures returned by the query. If no matches are found, the variable specified by `COUNT` will contain zero.

Example

The following example code shows how to programmatically configure a Query/Retrieve SCP Application Entity and perform a query.

Note

In order to run this code, you must supply values for the `REMOTE_AET`, `REMOTE_HOST`, and `REMOTE_PORT` variables.

To run the example, do the following:

1. Copy the example code and paste it into an IDL editor window or another text editor of your choice.
2. Edit the text to supply the values for the three required variables.
3. Save the file as `query_patient_root_patient_level.pro`.
4. Execute `query_patient_root_patient_level.pro` in IDL.

```
PRO query_patient_root_patient_level

; NOTE:
; To run this example on your own system, define the following
; variables to be valid remote query/retrieve SCP node values.
REMOTE_AET = 'remote_aet'
REMOTE_HOST = 'remote_host_name'
REMOTE_PORT = 'remote_port_number'

; Update the local user configuration file.
; First create a new local user configuration object.
ocfg = OBJ_NEW('IDLffDicomExCfg')

; Add the Remote Query/Retrieve SCP Application Entity.
ocfg->SetApplicationEntity, 'remote_qr_scp_aen', $
    REMOTE_AET, REMOTE_HOST, REMOTE_PORT, $
    'Query_SCP_Service_List', 'Query_SCP'

; Save the changes to the configuration file.
```

```

ocfg->Commit

; Destroy the configuration file object.
OBJ_DESTROY, ocfg

; Perform a patient root, patient level query against
; the remote node. First define a new Query/Retrieve SCU object.
oqr = OBJ_NEW('IDLffDicomExQuery')

; Specify the remote node to query.
oqr->SetProperty, QUERY_SCP = 'remote_qr_scp_aen'

; Run the query.
xResults = oqr->Query(COUNT=cnt)

; Print out the query results.
FOR i=0, cnt-1 DO BEGIN
    PRINT, 'i = ', i, ' patient_id = ', $
           xResults[i].patient_id, ' patient_name = ', $
           xResults[i].patient_name
ENDFOR

; Destroy the Query/Retrieve SCU object.
OBJ_DESTROY, oqr

END

```

Version History

6.3	Introduced
-----	------------

IDLffDicomExQuery::QueryModelsSupported

The IDLffDicomExQuery::QueryModelsSupported function method is used to determine what Query/Retrieve services are supported by Query/Retrieve SCP Application Entity defined by the `QUERY_SCP` property.

The service types available determine the types of queries are supported. You may wish to check this value before setting the `QUERY_MODEL` property.

Syntax

```
Result = Obj->[IDLffDicomExQuery::]QueryModelsSupported(
    [, COUNT=variable])
```

Return Value

This method returns an array of integers. Up to nine different integers can be returned.

Value	Service Type	Meaning
0	PATIENT_ROOT_QR_FIND	Indicates that the remote node supports patient root queries. The IDLffDicomExQuery object uses this service when the query model is set to Patient Root (QUERY_MODEL=0).
1	PATIENT_ROOT_QR_GET	Indicates that the remote node supports the patient root file get command for file retrieval. The IDLffDicomExQuery object does not use this service.
2	PATIENT_ROOT_QR_MOVE	Indicates that the remote node supports the patient root file move command for file retrieval. The IDLffDicomExQuery object uses this service when retrieving files if the query model is set to Patient Root (QUERY_MODEL=0).

Table 3-28: Return Values for QueryModelsSupported Method

Value	Service Type	Meaning
3	PATIENT_STUDY_ONLY_QR_FIND	The remote node supports patient/study root queries. The IDLffDicomExQuery object uses this service when the query model is set to Patient Study Only (QUERY_MODEL=2).
4	PATIENT_STUDY_ONLY_QR_GET	The remote node supports the patient study only file get command for file retrieval. The IDLffDicomExQuery object does not use this service.
5	PATIENT_STUDY_ONLY_QR_MOVE	The remote node supports the patient study only file move command for file retrieval. The IDLffDicomExQuery object uses this service when retrieving files if the query model is set to Patient Study Only (QUERY_MODEL=2).
6	STUDY_ROOT_QR_FIND	The remote node supports study root queries. The IDLffDicomExQuery object uses this service when the query model is set to Study Root (QUERY_MODEL=1).
7	STUDY_ROOT_QR_GET	The remote node supports the study root file get command for file retrieval. The IDLffDicomExQuery object does not use or need this service.
8	STUDY_ROOT_QR_MOVE	The remote node supports the study root file move command for file retrieval. The IDLffDicomExQuery object uses this service when retrieving files if the query model is set to Study Root (QUERY_MODEL=1).

Table 3-28: Return Values for QueryModelsSupported Method (Continued)

Arguments

None

Keywords

COUNT

Set this keyword equal to a named variable that will contain an integer representing the number of integer elements in the returned array.

Version History

6.3	Introduced
-----	------------

IDLffDicomExQuery::Retrieve

The IDLffDicomExQuery::Retrieve function method moves a DICOM file or files from a source node to a destination node. The source node is the remote Query/Retrieve SCP Application Entity that is queried. The destination node is a local or remote Storage SCP Application Entity. To accomplish the retrieval, the local Query/Retrieve SCU issues a move request to the remote Query/Retrieve SCP, which moves the file to the destination node.

Note

In DICOM terminology, the process of transferring data from a Query/Retrieve SCP to a Storage SCP is called a *move*. Note, however, that this process actually places a *copy* of the data on the destination node, leaving the original data intact.

This method does not return to the caller until one of the following occurs:

- The retrieve is complete (all matches have been returned)
- The retrieve encounters an error condition
- The callback function defined by the [CALLBACK_FUNCTION](#) property returns zero

Properties Required by a Retrieve

The following properties must be defined in order for the Retrieve method to succeed:

- The [QUERY_SCP](#) property must contain the name of a valid Query/Retrieve SCP node.
- The [STORAGE_SCP](#) property must contain the name of a valid Storage SCP node.

All other properties used by the Retrieve method have usable default values. If no properties other than [QUERY_SCP](#) and [STORAGE_SCP](#) are set, this method will perform a Patient Root / Patient Level retrieve, retrieving all patients on the remote node.

Retrieval Specification

Specify which files to retrieve using the [PATIENT_ID](#), [STUDY_INSTANCE_UID](#), [SERIES_INSTANCE_UID](#), and [SOP_INSTANCE_UID](#) keywords.

Note

Query field properties are not used by the Retrieve method to specify which files should be retrieved.

- If you specify a Patient ID, all of the DICOM files or images associated with the specified patient are retrieved.
- If you specify a Study Instance UID, only DICOM files or images associated with the specified study and its associated series are retrieved.
- If you specify a Series Instance UID, only DICOM files or images associated with the specified series are retrieved.
- If you specify an SOP Instance UID, a single DICOM file or image is retrieved.

Callbacks and Retrieval Status Information

If the [CALLBACK_FUNCTION](#) property contains the name of an IDL function, that function is called in the following circumstances:

- If the source node (the remote Query/Retrieve SCP) implements the optional response mechanism, the callback function is called each time a DICOM file is moved from the source node to the destination node.
- If the source node (the remote Query/Retrieve SCP) does not implement the optional response mechanism, the callback function is called every 3 seconds.

The function is called with an array of strings containing status information about the query as its first parameter. See [“Using Callbacks with the IDLffDicomExQuery Object”](#) on page 258 for a discussion of callback functions.

Canceling a Retrieve

To cancel a retrieve before all matches are returned, the callback function specified by the [CALLBACK_FUNCTION](#) property must return zero.

Syntax

```
Result = Obj->[IDLffDicomExQuery::]Retrieve( [, COUNT=variable]
    [, PATIENT_ID=string] [, STUDY_INSTANCE_UID=string]
    [, SERIES_INSTANCE_UID=string] [, SOP_INSTANCE_UID=string] )
```

Return Value

This method returns one (success) if the retrieval operation completes successfully or if the retrieval is cancelled based on the return value of the callback function. This method returns zero (failure) if the retrieve does not complete because an error is encountered.

Arguments

None

Keywords

COUNT

Set this keyword equal to a named variable that will contain an integer representing the number of files successfully retrieved as reported by the remote Query SCP node.

PATIENT_ID

Set this keyword equal to a scalar string containing the value of the Patient ID attribute (0010,0020) for the DICOM files to be retrieved.

The combination of query model and query level determine whether this keyword is required. See [“Unique ID Keyword Interactions With Query Model/Level”](#) on page 276 for details.

STUDY_INSTANCE_UID

Set this keyword equal to a scalar string containing the value of the Study Instance UID attribute (0020,000D) for the DICOM files to be retrieved.

The combination of query model and query level determine whether this keyword is required. See [“Unique ID Keyword Interactions With Query Model/Level”](#) on page 276 for details.

SERIES_INSTANCE_UID

Set this keyword equal to a scalar string containing the value of the Series Instance UID attribute (0020,000E) for the DICOM files to be retrieved.

The combination of query model and query level determine whether this keyword is required. See [“Unique ID Keyword Interactions With Query Model/Level”](#) on page 276 for details.

SOP_INSTANCE_UID

Set this keyword equal to a scalar string containing the value of the SOP Instance UID attribute (0008,0018) for the DICOM files to be retrieved.

The combination of query model and query level determine whether this keyword is required. See “[Unique ID Keyword Interactions With Query Model/Level](#)” on page 276 for details.

Unique ID Keyword Interactions With Query Model/Level

[Table 3-29](#) defines which unique ID keywords are required for different combinations of query model and query level. A "•" in the column indicates that the keyword is required when retrieving files using the specified combination of [QUERY_MODEL](#) and [QUERY_LEVEL](#) properties.

	Patient_ID	Study_Instance_UID	Series_Instance_UID	SOP_Instance_UID
Patient Root / Patient Level	•			
Patient Root / Study Level	•	•		
Patient Root / Series Level	•	•	•	
Patient Root / Image Level	•	•	•	•
Study Root / Study Level		•		
Study Root / Series Level		•	•	
Study Root / Image Level		•	•	•
Patient Study Only Root	•	•		

Table 3-29: Query Model/Level vs. Unique ID Keyword Usage

Example

The following example code shows how to programmatically configure a Query/Retrieve SCP Application Entity and retrieve files.

Note

In order to run this code, you must supply values for the `REMOTE_AET`, `REMOTE_HOST`, `REMOTE_PORT`, and `PATIENT_ID` variables. You must also have a local storage SCP configured on a remote query/retrieve SCP node.

To run the example, do the following:

1. Copy the example code and paste it into an IDL editor window or another text editor of your choice.
2. Edit the text to supply the values for the four required variables.
3. Save the file as `retrieve_dicom_patient_files.pro`.
4. Execute `retrieve_dicom_patient_files.pro` in IDL.

```
PRO retrieve_dicom_patient_files

; NOTE:
; To run this example on your own system, define the following
; variables to be valid and existing remote query/retrieve SCP
; node values.
REMOTE_AET = 'remote_aet'
REMOTE_HOST = 'remote_host_name'
REMOTE_PORT = 'remote_port_number'
; In addition, define the following variable to contain
; a numeric Patient ID number for a patient known to exist
; on the remote server.
PATIENT_ID = 000001

; Update the local user configuration file.
; Note that in a production application, you would probably
; not need to update the configuration file each time the
; application runs.
;
; First create a new local user configuration object.
ocfg = OBJ_NEW('IDLffDicomExCfg')

; Add the Remote Query/Retrieve SCP Application Entity.
ocfg->SetApplicationEntity, 'remote_qr_scp_aen', $
    REMOTE_AET, REMOTE_HOST, REMOTE_PORT, $
    'Query_SCP_Service_List', 'Query_SCP'
```

```

; Retrieve the value of the current Storage SCP Application
; Entity. We will use this later to restore the original value.
origStorScpAENAME = ocfg->GetValue('StorScpServiceAE')
origStorScpAE = ocfg->GetApplicationEntity(origStorScpAENAME)

; Add the local IDL Storage SCP Application Entity to the
; local configuration file. Note that 'my_stor_scp' must be
; configured on the remote query/retrieve SCP node.
ocfg->SetApplicationEntity, 'my_stor_scp_aen', $
    'my_stor_scp', 'localhost', 2510, $
    'Storage_SCP_Service_List', 'Storage_SCP'
ocfg->SetValue, 'StorScpServiceAE', 'my_stor_scp_aen'

; Save the changes to the configuration file.
ocfg->Commit

; Destroy the configuration file object.
OBJ_DESTROY, ocfg

; Update the system configuration file.
; Note that in a production application, you would probably
; not need to update the configuration file each time the
; application runs.
;
; First create a new system configuration object.
ocfg = OBJ_NEW('IDLffDicomExCfg', /SYSTEM)

; Stop the Storage SCP Service before making changes.
status = ocfg->StorageScpService('stop')
PRINT, 'Stopping Storage SCP. Status: ', status
WAIT, 5

; Add the Storage SCP Application Entity to the system
; configuration file. Note that 'my_stor_scp' must be
; configured on the remote query/retrieve SCP node.
ocfg->SetApplicationEntity, 'my_stor_scp_aen', $
    'my_stor_scp', 'localhost', 2510, $
    'Storage_SCP_Service_List', 'Storage_SCP'
ocfg->SetValue, 'StorScpServiceAE', 'my_stor_scp_aen'

; Retrieve the original value of the Storage SCP directory.
origStorScpDir = ocfg->GetValue('StorScpDir')

; Set the Storage SCP directory to the temporary directory.
tmpDir = FILEPATH('', /TMP)
ocfg->SetValue, 'StorScpDir', tmpDir
storScpDir = ocfg->GetValue('StorScpDir')
PRINT, 'Setting Storage SCP Directory to: ', storScpDir

```

```
; Save the changes to the configuration file.
ocfg->Commit

; Restart the Storage SCP Service.
status = ocfg->StorageScpService('start')
PRINT, 'Starting Storage SCP. Status: ', status
WAIT, 5
status = ocfg->StorageScpService('status')
PRINT, 'Storage SCP Status: ', status
WAIT, 5

; Destroy the configuration file object.
OBJ_DESTROY, ocfg

; Retrieve all patient files for the patient from
; the remote node.
; First, create a new Query/Retrieve SCU object.
oqr = OBJ_NEW('IDLffDicomExQuery')

; Specify the node to be queried.
oqr->SetProperty, QUERY_SCP = 'remote_qr_scp_aen'

; Specify the node to which files will be retrieved.
oqr->SetProperty, STORAGE_SCP = 'my_stor_scp_aen'

; Retrieve the files
res = oqr->Retrieve(PATIENT_ID=PATIENT_ID, COUNT=cnt )

PRINT, 'Retrieve Status: ', cnt, ' files retrieved into ',
storScpDir

; Destory the Query/Retrieve SCU object.
OBJ_DESTROY, oqr

; Restore the Storage SCP Application Entity to its
; original state.
;
PRINT, ''
PRINT, 'Restoring original Storage SCP configuration.'

; Get a new local user configuration object.
ocfg = OBJ_NEW('IDLffDicomExCfg')
```

```

; Replace the original Storage SCP Application Entity.
ocfg->SetApplicationEntity, origStorScpAE.APPLENTITYNAME, $
    origStorScpAE.AET, origStorScpAE.HOSTNAME, $
    origStorScpAE.PORT, origStorScpAE.SERVICELISTNAME, $
    origStorScpAE.SERVICETYPE

ocfg->SetValue, 'StorScpServiceAE', origStorScpAE.APPLENTITYNAME

; Save the changes to the configuration file.
ocfg->Commit

; Destroy the local configuration file object.
OBJ_DESTROY, ocfg

; Get a new System configuration object.
ocfg = OBJ_NEW('IDLffDicomExCfg', /SYSTEM)

; Stop the Storage SCP Service before making changes
; to the system configuration.
status = ocfg->StorageScpService('stop')
PRINT, 'Stopping Storage SCP. Status: ', status
WAIT, 5

; Replace the original Storage SCP Application Entity.
ocfg->SetApplicationEntity, origStorScpAE.APPLENTITYNAME, $
    origStorScpAE.AET, origStorScpAE.HOSTNAME, $
    origStorScpAE.PORT, origStorScpAE.SERVICELISTNAME, $
    origStorScpAE.SERVICETYPE

ocfg->SetValue, 'StorScpServiceAE', origStorScpAE.APPLENTITYNAME

; Restore the Storage SCP directory.
ocfg->SetValue, 'StorScpDir', origStorScpDir
PRINT, 'Storage SCP Directory is now ', $
    ocfg->GetValue('StorScpDir')

; Restart the Storage SCP Service.
status = ocfg->StorageScpService('start')
PRINT, 'Starting Storage SCP. Status: ', status
WAIT, 5
status = ocfg->StorageScpService('status')
PRINT, 'Storage SCP Status: ', status
WAIT, 5

; Save the changes to the configuration file.
ocfg->Commit

; Destroy the configuration file object.

```



```
OBJ_DESTROY, ocfg
```

```
END
```

Version History

6.3	Introduced
-----	------------

IDLffDicomExQuery:: SetProperty

The IDLffDicomExQuery::SetProperty procedure method sets the value of a property or group of properties for the IDLffDicomExQuery object.

Syntax

Obj->[IDLffDicomExQuery::]SetProperty[, *PROPERTY=value*]

Arguments

None

Keywords

Any property listed under “IDLffDicomExQuery Properties” on page 242 that contains the word “Yes” in the “Set” column of the properties table can be set using this method. To set the value of a property, specify the property name as a keyword set equal to the appropriate property value.

Version History

6.3	Introduced
-----	------------

IDLffDicomExStorScu

The IDLffDicomExStorScu object implements a local DICOM Storage SCU (Service Class User) that allows you to transmit files to a remote destination that is identified as a DICOM Storage SCP (Service Class Provider). This sends a copy of the images to the file storage machine, leaving the original image data intact. For an introduction to IDL's DICOM Network Service feature, see [Chapter 2, “Using IDL DICOM Network Services”](#).

Parameters accessible through this object correspond to those shown on the **Storage SCU** tab of the DICOM Network Services utility.

Superclasses

None

Creation

See [“IDLffDicomExStorScu::Init”](#) on page 292.

Properties

The IDLffDicomExStorScu object has the following properties:

- [CALLBACK_DATA](#)
- [CALLBACK_FUNCTION](#)
- [STORAGE_SCP](#)

See [“IDLffDicomExStorScu Properties”](#) on page 285 for details on individual properties.

Methods

This class has the following methods:

- [IDLffDicomExStorScu::Cleanup](#)
- [IDLffDicomExStorScu::ClearProperties](#)
- [IDLffDicomExStorScu::GetProperty](#)
- [IDLffDicomExStorScu::Init](#)
- [IDLffDicomExStorScu::Send](#)

- [IDLffDicomExStorScu::SetProperty](#)

In addition, this class inherits the methods of its superclasses (if any).

Examples

See the Example sections of the following methods for examples using the `IDLffDicomExStorScu` object:

- [IDLffDicomExStorScu::Send](#)

In addition, the DICOM Network Services utility, which is included in the IDL distribution, includes the `cw_dicomex_stor_scu.pro` routine, which makes extensive use of this object. The `cw_dicomex_stor_scu.pro` routine is located in the `lib/dicomex` subdirectory of the IDL distribution.

Version History

6.3	Introduced
-----	------------

IDLffDicomExStorScu Properties

IDLffDicomExStorScu objects have the following properties. Properties with the word “Yes” in the “Get” column of the property table can be retrieved via [IDLffDicomExStorScu::GetProperty](#). Properties with the word “Yes” in the “Init” column of the property table can be set via [IDLffDicomExStorScu::Init](#). Properties with the word “Yes” in the “Set” column in the property table can be set via [IDLffDicomExStorScu::SetProperty](#).

Note

For a discussion of the property description tables shown below, see [“About Object Property Descriptions”](#) on page 70. Tables for properties of this object do not include the **DICOM Attribute**, **VR**, or **Multi-value** fields.

Objects of this class have the following properties.

- [CALLBACK_DATA](#)
- [CALLBACK_FUNCTION](#)
- [STORAGE_SCP](#)

CALLBACK_DATA

This property contains data that is to be passed to the function defined by the [CALLBACK_FUNCTION](#) property when a callback is initiated. If this property is not set, the integer value zero is passed to the callback function. See [“Using Callbacks With the IDLffDicomExStorScu Object”](#) on page 287 for additional details.

This property is used by the [IDLffDicomExStorScu::Send](#) method.

Property Type	Any type		
Name String	<i>not displayed</i>		
Get: Yes	Set: Yes	Init: No	Registered: No

CALLBACK_FUNCTION

This property contains the name of an IDL function to be called by the [Send](#) method. If no value is defined, no callback is initiated. See [“Using Callbacks With the IDLffDicomExStorScu Object”](#) on page 287 for additional details.

This property is used by the [IDLffDicomExStorScu::Send](#) method.

Property Type	String		
Name String	<i>not displayed</i>		
Get: Yes	Set: Yes	Init: No	Registered: No

STORAGE_SCP

This property specifies the Application Entity name of the Storage SCP node to which DICOM files will be sent by the Send method. The Application Entity name must be defined in the DICOM Network Services configuration file.

Note

This property *must* be set prior to calling the Send method.

This property is used by the [IDLffDicomExStorScu::Send](#) method.

Property Type	String		
Name String	<i>not displayed</i>		
Get: Yes	Set: Yes	Init: No	Registered: No

Using Callbacks With the IDLffDicomExStorScu Object

Callbacks from the IDLffDicomExStorScu object provide a way to transmit information back to the caller based on the status of a send operation. The value returned by the callback function is then used by the IDLffDicomExStorScu object to determine whether the send operation should continue.

The IDLffDicomExStorScu object allows you to define a single function (written in IDL) that will be called based on the intermediate status of a [IDLffDicomExStorScu::Send](#) method call. If a callback function is specified via the [CALLBACK_FUNCTION](#) property, it is called each time a DICOM file is sent to the remote node.

The callback function is invoked with an array of strings indicating status as the first parameter, and with the value (if any) specified by the [CALLBACK_DATA](#) property as the second parameter.

If the return value of the callback function is zero, the send operation will be cancelled. If the return value is one, the send operation will continue.

Callback Routine Signature

A send callback function is written in IDL and has the following signature:

```
Function Callback_Function_Name, StatusInfo, CallbackData
```

where

- *Callback_Function_Name* is the name of the callback function. This value is specified as the value of the [CALLBACK_FUNCTION](#) property.
- *StatusInfo* is an array of strings that contain status information about the send operation. The status strings are provided for information and human inspection rather than for programmatic processing; the exact strings included in the array will depend on numerous factors including the status of the send and the type of node being sent to.
- *CallbackData* is the data specified via the [CALLBACK_DATA](#) property. The value of this property is passed to the callback function unmodified. If the [CALLBACK_DATA](#) property is unspecified, an integer zero is passed to the callback function as the value of this parameter.

The *CallbackData* parameter is useful for passing static information, such as the widget ID of a widget where the status information is displayed, from the IDLffDicomExStorScu object to the callback routine.

The return value of the callback function should be an integer zero or one. If the return value is zero, the send operation will be cancelled. If the return value is one, the send operation will continue.

IDLffDicomExStorScu::Cleanup

The IDLffDicomExStorScu::Cleanup procedure method performs all cleanup on the object.

Note

Cleanup methods are special *lifecycle methods*, and as such cannot be called outside the context of object creation and destruction. In most cases, you cannot call the Cleanup method directly. However, one exception to this rule does exist. If you write your own subclass of this class, you can call the Cleanup method from within the Init or Cleanup method of the subclass.

Syntax

OBJ_DESTROY, *Obj*

or

Obj->[IDLffDicomExStorScu::]Cleanup() *(In a lifecycle method only.)*

Arguments

None

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExStorScu::ClearProperties

The IDLffDicomExStorScu::ClearProperties procedure method resets the values of all properties of the IDLffDicomExStorScu object to their default values.

Syntax

Obj->[IDLffDicomExStorScu::](#)ClearProperties

Arguments

None

Keywords

None

Version History

6.3	Introduced
-----	------------

See Also

[IDLffDicomExStorScu Properties](#)

IDLffDicomExStorScu::GetProperty

The IDLffDicomExStorScu::GetProperty procedure method retrieves the value of an IDLffDicomExStorScu property.

Syntax

Obj->[IDLffDicomExStorScu::]GetProperty[, *PROPERTY=variable*]

Arguments

None

Keywords

Any property listed under “IDLffDicomExStorScu Properties” on page 285 that contains the word “Yes” in the “Get” column of the properties table can be retrieved using this method. To retrieve the value of a property, specify the property name as a keyword set equal to a named variable that will contain the value of the property.

Version History

6.3	Introduced
-----	------------

IDLffDicomExStorScu::Init

The IDLffDicomExStorScu::Init function method initializes the IDLffDicomExStorScu object. The IDLffDicomExStorScu object implements a local DICOM Storage SCU that allows you to transmit files to a remote node that is configured as a DICOM Storage SCP available in IDL's DICOM Network Service configuration. For an introduction to IDL's DICOM Network Service feature, see [Chapter 2, “Using IDL DICOM Network Services”](#).

Note

Init methods are special lifecycle methods, and as such cannot be called outside the context of object creation. This means that in most cases, you cannot call the Init method directly. There is one exception to this rule: if you write your own subclass of this class, you can call the Init method from within the Init method of the subclass.

Syntax

Obj = OBJ_NEW('IDLffDicomExStorScu')

or

Result = *Obj*->[IDLffDicomExStorScu::]Init() *(In a lifecycle method only.)*

Return Value

When this method is called indirectly, as part of the call to the OBJ_NEW function, the return value is an object reference to the newly-created object.

When called directly within a subclass Init method, the return value is 1 if initialization was successful, or zero otherwise.

Arguments

None

Keywords

None

Version History

6.3	Introduced
-----	------------

IDLffDicomExStorScu::Send

The IDLffDicomExStorScu::Send function method transfers DICOM files to a remote Storage SCP node.

This method does not return to the caller until one of the following occurs:

- The send is complete (all files have been transferred)
- The send encounters an error condition
- The callback function defined by the [CALLBACK_FUNCTION](#) property returns zero

Properties Required For a Send

The [STORAGE_SCP](#) property must contain the name of a valid Application Entity that supports the Storage SCP service in order for the Send method to succeed. All to other properties used by the Send method have usable default values.

Callbacks and Send Status Information

If the [CALLBACK_FUNCTION](#) property contains the name of an IDL function, that function is called each time a DICOM file is sent to the remote node. The function is called with an array of strings containing status information about the send operation as its first parameter. See “[Using Callbacks With the IDLffDicomExStorScu Object](#)” on page 287 for a discussion of callback functions.

Canceling a Send

To cancel a send operation before all files are transferred, the callback function specified by the [CALLBACK_FUNCTION](#) property must return zero.

Syntax

```
Result = Obj->[IDLffDicomExStorScu::]Send(Files)
```

Return Value

This method returns the number of DICOM files that were successfully sent.

Arguments

Files

A scalar string or string array containing the fully qualified paths of the DICOM files to be transferred to the remote Storage SCP node.

Keywords

None

Example

The following example code shows how to programmatically configure a local Storage SCP, create a local Storage SCU Application Entity, and transfer files to the Storage SCP.

To run the example, do the following:

1. Copy the example code and paste it into an IDL editor window or another text editor of your choice.
2. Save the file as `stor_scu_send_files.pro`.
3. Execute `stor_scu_send_files.pro` in IDL.

```
PRO stor_scu_send_files

; Update the local user configuration file.
; Note that in a production application, you would probably
; not need to update the configuration file each time the
; application runs.
;
; First create a new local user configuration object.
ocfg = OBJ_NEW('IDLffDicomExCfg')

; Add the local IDL Storage SCP Application Entity to the
; local configuration file. Note that 'my_stor_scp' must be
; configured on the remote query/retrieve SCP node.
ocfg->SetApplicationEntity, 'my_stor_scp_aen', $
    'my_stor_scp', 'localhost', 2510, $
    'Storage_SCP_Service_List', 'Storage_SCP'
ocfg->SetValue, 'StorScpServiceAE', 'my_stor_scp_aen'

; Save the changes to the configuration file.
ocfg->Commit
```

```

; Destroy the configuration file object.
OBJ_DESTROY, ocfg

; Update the system configuration file.
; Note that in a production application, you would probably
; not need to update the configuration file each time the
; application runs.
;
; First create a new system configuration object.
ocfg = OBJ_NEW('IDLffDicomExCfg', /SYSTEM)

; Add the Storage SCP Application Entity to the system
; configuration file. Note that 'my_stor_scp' must be
; configured on the remote query/retrieve SCP node.
ocfg->SetApplicationEntity, 'my_stor_scp_aen', $
    'my_stor_scp', 'localhost', 2510, $
    'Storage_SCP_Service_List', 'Storage_SCP'
ocfg->SetValue, 'StorScpServiceAE', 'my_stor_scp_aen'

; Set the Storage SCP directory to the temporary directory.
tmpDir = FILEPATH('', /TMP)
ocfg->SetValue, 'StorScpDir', tmpDir
PRINT, 'stor scp dir = ', tmpDir

; Save the changes to the configuration file.
ocfg->Commit

; Stop and restart the Storage SCP Service. Restarting is
; necessary because we have changed the Storage SCP configuration.
status = ocfg->StorageScpService('stop')
PRINT, 'Stopping Storage SCP. Status: ', status
WAIT, 5
status = ocfg->StorageScpService('start')
PRINT, 'Starting Storage SCP. Status: ', status
WAIT, 5

; Destroy the configuration file object.
OBJ_DESTROY, ocfg

; Select a file to transfer to the Storage SCP.
srcfn1 = FILEPATH('mr_brain.dcm', SUBDIR=['examples', 'data'])

; Create a new Storage SCU object and associate the Storage SCP.
oss = OBJ_NEW('IDLffDicomExStorScu')
oss->SetProperty, STORAGE_SCP = 'my_stor_scp_aen'

; Transfer the file to the Storage SCP
cnt = oss->Send(srcfn1)

```



```
; Report the number of files transferred.  
PRINT, 'Number of files transferred: ', cnt  
  
; Destroy the Storage SCU object.  
OBJ_DESTROY, oss  
  
END
```

Version History

6.3	Introduced
-----	------------

IDLffDicomExStorScu:: SetProperty

The IDLffDicomExStorScu::SetProperty procedure method sets the value of a property or group of properties for the IDLffDicomExStorScu object.

Syntax

Obj->[IDLffDicomExStorScu::]SetProperty[, *PROPERTY=value*]

Arguments

None

Keywords

Any property listed under “IDLffDicomExStorScu Properties” on page 285 that contains the word “Yes” in the “Set” column of the properties table can be set using this method. To set the value of a property, specify the property name as a keyword set equal to the appropriate property value.

Version History

6.3	Introduced
-----	------------



Appendix A DICOM Resources

This appendix provides DICOM attribute and VR (value representation) resource information.

DICOM Attributes	300	Value Representations	372
------------------------	-----	-----------------------------	-----

DICOM Attributes

A DICOM *attribute* or *data element* is composed of:

- A *tag*, in the format of *group, element* (xxxx,xxxx) that identifies the element
- A *Value Representation* (VR) that describes the data type and format of the attribute's value
- A *value length* that defines the length of the attribute's value
- A *value field* containing the attribute's data

The basic attribute structure is shown in the following figure.



Figure A-1: DICOM Attribute (Data Element) Structure

Each data element is described by a pair of numbers (group number, data element number). Even numbered groups are elements defined by the DICOM standard and are referred to as public tags. Odd numbered groups can be defined by users of the file format, but must conform to the same structure as standard elements. These are referred to as private tags.

Note

This section does *not* contain a comprehensive IOD list for each modality. Visit <http://medical.nema.org/dicom.html> for additional DICOM information including access to the complete standard.

The following table lists the DICOM attributes organized by attribute name. This information comes from the 2001 NEMA DICOM Standard

(<http://medical.nema.org/dicom/2001.html>). See “Value Representations” on page 372 for definitions of the VR items.

Attribute Name	VR	Tag
Accession Number	SH	(0008,0050)
Acquisition Context Description	ST	(0040,0556)
Acquisition Context Sequence	SQ	(0040,0555)
Acquisition Date	DA	(0008,0022)
Acquisition Datetime	DT	(0008,002A)
Acquisition Device Processing Code	LO	(0018,1401)
Acquisition Device Processing Description	LO	(0018,1400)
Acquisition Matrix	US	(0018,1310)
Acquisition Number	IS	(0020,0012)
Acquisition Start Condition	CS	(0018,0073)
Acquisition Start Condition Data	IS	(0018,0074)
Acquisition Termination Condition	CS	(0018,0071)
Acquisition Termination Condition Data	IS	(0018,0075)
Acquisition Time	TM	(0008,0032)
Acquisition Time Synchronized	CS	(0018,1800)
Acquisitions in Study	IS	(0020,1004)
Active Source Diameter	DS	(300A,0218)
Active Source Length	DS	(300A,021A)
Actual Frame Duration	IS	(0018,1242)
Actual Human Performers Sequence	SQ	(0040,4035)
Additional Drug Sequence	SQ	(0018,002A)

Table A-1: DICOM Attributes

Attribute Name	VR	Tag
Additional Patient History	LT	(0010,21B0)
Administration Route Code Sequence	SQ	(0054,0302)
Admission ID	LO	(0038,0010)
Admitting Date	DA	(0038,0020)
Admitting Diagnoses Code Sequence	SQ	(0008,1084)
Admitting Diagnoses Description	LO	(0008,1080)
Admitting Time	TM	(0038,0021)
Air Kerma Rate Reference Date	DA	(300A,022C)
Air Kerma Rate Reference Time	TM	(300A,022E)
Anatomic Region Modifier Sequence	SQ	(0008,2220)
Anatomic Region Sequence	SQ	(0008,2218)
Anatomic Structure, Space or Region Sequence	SQ	(0008,2229)
Anchor Point	FL	(0070,0014)
Anchor Point Annotation Units	CS	(0070,0004)
Anchor Point Visibility	CS	(0070,0015)
Angio Flag	CS	(0018,0025)
Angular Position	DS	(0018,1141)
Angular Step	DS	(0018,1144)
Angular View Vector	US	(0054,0090)
Annotation Content Sequence	SQ	(2130,0050)
Annotation Display Format ID	CS	(2010,0030)
Annotation Flag	CS	(2000,0065)
Annotation Group Number	US	(0040,A180)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Annotation Position	US	(2030,0010)
Annotation Sequence	SQ	(0040,B020)
Anode Target Material	CS	(0018,1191)
Applicable Frame Range	US	(0028,6102)
Application Setup Check	CS	(3008,0116)
Application Setup Manufacturer	LO	(300A,0238)
Application Setup Name	LO	(300A,0236)
Application Setup Number	IS	(300A,0234)
Application Setup Sequence	SQ	(300A,0230)
Application Setup Type	CS	(300A,0232)
Applicator Description	LO	(300A,010A)
Applicator ID	SH	(300A,0108)
Applicator Sequence	SQ	(300A,0107)
Applicator Type	CS	(300A,0109)
Approval Status	CS	(300E,0002)
Attached Contours	IS	(3006,0049)
Attenuation Correction Method	LO	(0054,1101)
Audio Comments	LT	(50xx,200E)
Audio Sample Data	OW or OB	(50xx,200C)
Audio Sample Format	US	(50xx,2002)
Audio Type	US	(50xx,2000)
Authorization Equipment Certification Number	LO	(0100,0426)
Average Pulse Width	DS	(0018,1154)
Axial Acceptance	DS	(0054,1200)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Axial Mash	IS	(0054,1201)
Axis Labels	SH	(50xx,0040)
Axis Units	SH	(50xx,0030)
Basic Color Image Sequence	SQ	(2020,0111)
Basic Grayscale Image Sequence	SQ	(2020,0110)
Beam Description	ST	(300A,00C3)
Beam Dose	DS	(300A,0084)
Beam Dose Specification Point	DS	(300A,0082)
Beam Limiting Device Angle	DS	(300A,0120)
Beam Limiting Device Angle Tolerance	DS	(300A,0046)
Beam Limiting Device Leaf Pairs Sequence	SQ	(3008,00A0)
Beam Limiting Device Position Sequence	SQ	(300A,011A)
Beam Limiting Device Position Tolerance	DS	(300A,004A)
Beam Limiting Device Rotation Direction	CS	(300A,0121)
Beam Limiting Device Sequence	SQ	(300A,00B6)
Beam Limiting Device Tolerance Sequence	SQ	(300A,0048)
Beam Meterset	DS	(300A,0086)
Beam Name	LO	(300A,00C2)
Beam Number	IS	(300A,00C0)
Beam Sequence	SQ	(300A,00B0)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Beam Stopper Position	CS	(3008,0230)
Beam Type	CS	(300A,00C4)
Beat Rejection Flag	CS	(0018,1080)
Billing Item Sequence	SQ	(0040,0296)
Billing Procedure Step Sequence	SQ	(0040,0320)
Billing Supplies and Devices Sequence	SQ	(0040,0324)
Bi-Plane Acquisition Sequence	SQ	(0028,5000)
Bits Allocated	US	(0028,0100)
Bits Stored	US	(0028,0101)
Block Data	DS	(300A,0106)
Block Divergence	CS	(300A,00FA)
Block Name	LO	(300A,00FE)
Block Number	IS	(300A,00FC)
Block Number of Points	IS	(300A,0104)
Block Sequence	SQ	(300A,00F4)
Block Thickness	DS	(300A,0100)
Block Transmission	DS	(300A,0102)
Block Tray ID	SH	(300A,00F5)
Block Type	CS	(300A,00F8)
Blue Palette Color Lookup Table Data	OW	(0028,1203)
Blue Palette Color Lookup Table Descriptor	US or SS	(0028,1103)
Body Part Examined	CS	(0018,0015)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Body Part Thickness	DS	(0018,11A0)
Border Density	CS	(2010,0100)
Bounding Box Annotation Units	CS	(0070,0003)
Bounding Box Bottom Right Hand Corner	FL	(0070,0011)
Bounding Box Text Horizontal Justification	CS	(0070,0012)
Bounding Box Top Left Hand Corner	FL	(0070,0010)
Brachy Accessory Device ID	SH	(300A,0263)
Brachy Accessory Device Name	LO	(300A,0266)
Brachy Accessory Device Nominal Thickness	DS	(300A,026A)
Brachy Accessory Device Nominal Transmission	DS	(300A,026C)
Brachy Accessory Device Number	IS	(300A,0262)
Brachy Accessory Device Sequence	SQ	(300A,0260)
Brachy Accessory Device Type	CS	(300A,0264)
Brachy Application Setup Dose	DS	(300A,00A4)
Brachy Application Setup Dose Specification Point	DS	(300A,00A2)
Brachy Control Point Delivered Sequence	SQ	(3008,0160)
Brachy Control Point Sequence	SQ	(300A,02D0)
Brachy Referenced Dose Reference Sequence	SQ	(300C,0055)
Brachy Treatment Technique	CS	(300A,0200)
Brachy Treatment Type	CS	(300A,0202)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Branch of Service	LO	(0010,1081)
Burned In Annotation	CS	(0028,0301)
Calculated Dose Reference Description	ST	(3008,0074)
Calculated Dose Reference Dose Value	DS	(3008,0076)
Calculated Dose Reference Number	IS	(3008,0072)
Calculated Dose Reference Sequence	SQ	(3008,0070)
Calibration Data Sequence	SQ	(0054,0306)
Calibration Image	CS	(0050,0004)
Cardiac Number of Images	IS	(0018,1090)
Cassette Orientation	CS	(0018,1402)
Cassette Size	CS	(0018,1403)
Center of Circular Collimator	IS	(0018,1710)
Center of Circular Shutter	IS	(0018,1610)
Center of Rotation Offset	DS	(0018,1145)
Certificate of Signer	OB	(0400,0115)
Certificate Type	CS	(0400,0110)
Certified Timestamp	OB	(0400,0310)
Certified Timestamp Type	CS	(0400,0305)
Channel Baseline	DS	(003A,0213)
Channel Definition Sequence	SQ	(003A,0200)
Channel Derivation Description	LO	(003A,020C)
Channel Label	SH	(003A,0203)
Channel Length	DS	(300A,0284)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Channel Maximum Value	OB or OW	(5400,0112)
Channel Minimum Value	OB or OW	(5400,0110)
Channel Number	IS	(300A,0282)
Channel Offset	DS	(003A,0218)
Channel Sample Skew	DS	(003A,0215)
Channel Sensitivity	DS	(003A,0210)
Channel Sensitivity Correction Factor	DS	(003A,0212)
Channel Sensitivity Units Sequence	SQ	(003A,0211)
Channel Sequence	SQ	(300A,0280)
Channel Shield ID	SH	(300A,02B3)
Channel Shield Name	LO	(300A,02B4)
Channel Shield Nominal Thickness	DS	(300A,02B8)
Channel Shield Nominal Transmission	DS	(300A,02BA)
Channel Shield Number	IS	(300A,02B2)
Channel Shield Sequence	SQ	(300A,02B0)
Channel Source Modifiers Sequence	SQ	(003A,0209)
Channel Source Sequence	SQ	(003A,0208)
Channel Status	CS	(003A,0205)
Channel Time Skew	DS	(003A,0214)
Channel Total Time	DS	(300A,0286)
Cine Rate	IS	(0018,0040)
Code Meaning	LO	(0008,0104)
Code Set Extension Creator UID	UI	(0008,010D)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Code Set Extension Flag	CS	(0008,010B)
Code Value	SH	(0008,0100)
Coding Scheme Designator	SH	(0008,0102)
Coding Scheme Version	SH	(0008,0103)
Coincidence Window Width	DS	(0054,1210)
Collation Flag	CS	(2000,0063)
Collimator Left Vertical Edge	IS	(0018,1702)
Collimator Lower Horizontal Edge	IS	(0018,1708)
Collimator Right Vertical Edge	IS	(0018,1704)
Collimator Shape	CS	(0018,1700)
Collimator Type	CS	(0018,1181)
Collimator Upper Horizontal Edge	IS	(0018,1706)
Collimator/grid Name	SH	(0018,1180)
Color Image Printing Flag	CS	(2000,0062)
Column Angulation	CS	(0018,1450)
Columns	US	(0028,0011)
Comments on Radiation Dose	ST	(0040,0310)
Comments on the Performed Procedure Step	ST	(0040,0280)
Comments on the Scheduled Procedure Step	LT	(0040,0400)
Compensator Columns	IS	(300A,00E8)
Compensator ID	SH	(300A,00E5)
Compensator Number	IS	(300A,00E4)
Compensator Pixel Spacing	DS	(300A,00E9)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Compensator Position	DS	(300A,00EA)
Compensator Rows	IS	(300A,00E7)
Compensator Sequence	SQ	(300A,00E3)
Compensator Thickness Data	DS	(300A,00EC)
Compensator Transmission Data	DS	(300A,00EB)
Compensator Type	CS	(300A,00EE)
Completion Flag	CS	(0040,A491)
Completion Flag Description	LO	(0040,A492)
Compression Force	DS	(0018,11A2)
Concept Code Sequence	SQ	(0040,A168)
Conceptname Code Sequence	SQ	(0040,A043)
Confidentiality Code	LO	(0040,1008)
Confidentiality Constraint on Patient Data Description	LO	(0040,3001)
Configuration Information	ST	(2010,0150)
Configuration Information Description	LT	(2010,0152)
Constraint Weight	DS	(300A,0021)
Content Date	DA	(0008,0023)
Content Sequence	SQ	(0040,A730)
Content Template Sequence	SQ	(0040,A504)
Content Time	TM	(0008,0033)
Context Group Local Version	DT	(0008,0107)
Context Group Version	DT	(0008,0106)
Context Identifier	CS	(0008,010F)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Continuity Of Content	CS	(0040,A050)
Contour Data	DS	(3006,0050)
Contour Geometric Type	CS	(3006,0042)
Contour Image Sequence	SQ	(3006,0016)
Contour Number	IS	(3006,0048)
Contour Offset Vector	DS	(3006,0045)
Contour Sequence	SQ	(3006,0040)
Contour Slab Thickness	DS	(3006,0044)
Contrast Allergies	LO	(0010,2110)
Contrast Flow Duration(s)	DS	(0018,1047)
Contrast Flow Rate(s)	DS	(0018,1046)
Contrast Frame Averaging	US	(0028,6112)
Contrast/Bolus Administration Route Sequence	SQ	(0018,0014)
Contrast/Bolus Agent	LO	(0018,0010)
Contrast/Bolus Agent Sequence	SQ	(0018,0012)
Contrast/Bolus Ingredient	CS	(0018,1048)
Contrast/Bolus Ingredient Concentration	DS	(0018,1049)
Contrast/Bolus Route	LO	(0018,1040)
Contrast/Bolus Start Time	TM	(0018,1042)
Contrast/Bolus Stop Time	TM	(0018,1043)
Contrast/Bolus Total Dose	DS	(0018,1044)
Contrast/Bolus Volume	DS	(0018,1041)
Control Point 3D Position	DS	(300A,02D4)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Control Point Delivery Sequence	SQ	(3008,0040)
Control Point Index	IS	(300A,0112)
Control Point Relative Position	DS	(300A,02D2)
Control Point Sequence	SQ	(300A,0111)
Conversion Type	CS	(0008,0064)
Convolution Kernel	SH	(0018,1210)
Coordinate Start Value	US	(50xx,0112)
Coordinate Step Value	US	(50xx,0114)
Coordinate System Axis Code Sequence	SQ	(0040,08DA)
Corrected Image	CS	(0028,0051)
Count Rate	IS	(0018,1243)
Country of Residence	LO	(0010,2150)
Counts Accumulated	IS	(0018,0070)
Counts Included	CS	(0054,1400)
Counts Source	CS	(0054,1002)
Cranial Thermal Index	DS	(0018,5026)
Creation Date	DA	(2100,0040)
Creation Time	TM	(2100,0050)
Cumulative Dose Reference Coefficient	DS	(300A,010C)
Cumulative Dose to Dose Reference	DS	(3008,0052)
Cumulative Meterset Weight	DS	(300A,0134)
Cumulative Time Weight	DS	(300A,02D6)
Current Fraction Number	IS	(3008,0022)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Current Patient Location	LO	(0038,0300)
Current Requested Procedure Evidence Sequence	SQ	(0040,A375)
Current Treatment Status	CS	(3008,0200)
Curve Activation Layer	CS	(50xx,1001)
Curve Data	OW or OB	(50xx,3000)
Curve Data Descriptor	US	(50xx,0110)
Curve Date	DA	(0008,0025)
Curve Description	LO	(50xx,0022)
Curve Dimensions	US	(50xx,0005)
Curve Label	LO	(50xx,2500)
Curve Number	IS	(0020,0024)
Curve Range	SH	(50xx,0106)
Curve Time	TM	(0008,0035)
Data Collection Diameter	DS	(0018,0090)
Data Elements Signed	AT	(0400,0020)
Data Information Sequence	SQ	(0054,0063)
Data Set Trailing Padding	OB	(FFFC,FFFC)
Data Value Representation	US	(50xx,0103)
Date	DA	(0040,A121)
Date of Last Calibration	DA	(0018,1200)
Date of Last Detector Calibration	DA	(0018,700C)
Date of Secondary Capture	DA	(0018,1012)
DateTime	DT	(0040,A120)
dB/dt	DS	(0018,1318)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Dead Time Correction Flag	CS	(0054,1401)
Dead Time Factor	DS	(0054,1324)
Decay Correction	CS	(0054,1102)
Decay Factor	DS	(0054,1321)
Decimate/Crop Result	CS	(2020,00A2)
Default Magnification Type	CS	(2010,00A6)
Default Printer Resolution ID	CS	(2010,0054)
Default Smoothing Type	CS	(2010,00A8)
Delivered Channel Total Time	DS	(3008,0134)
Delivered Meterset	DS	(3008,0044)
Delivered Number of Pulses	IS	(3008,0138)
Delivered Primary Meterset	DS	(3008,0036)
Delivered Pulse Repetition Interval	DS	(3008,013C)
Delivered Secondary Meterset	DS	(3008,0037)
Delivered Treatment Time	DS	(3008,003B)
Delivery Maximum Dose	DS	(300A,0023)
Delivery Warning Dose	DS	(300A,0022)
Depth of Scan Field	IS	(0018,5050)
Derivation Description	ST	(0008,2111)
Destination AE	AE	(2100,0140)
Detector Activation Offset From Exposure	DS	(0018,7016)
Detector Active Dimension(s)	DS	(0018,7026)
Detector Active Origin	DS	(0018,7028)
Detector Active Shape	CS	(0018,7024)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Detector Active Time	DS	(0018,7014)
Detector Binning	DS	(0018,701A)
Detector Conditions Nominal Flag	CS	(0018,7000)
Detector Configuration	CS	(0018,7005)
Detector Description	LT	(0018,7006)
Detector Element Physical Size	DS	(0018,7020)
Detector Element Size	DS	(0054,1203)
Detector Element Spacing	DS	(0018,7022)
Detector ID	SH	(0018,700A)
Detector Information Sequence	SQ	(0054,0022)
Detector Lines of Response Used	LO	(0054,1104)
Detector Mode	LT	(0018,7008)
Detector Primary Angle	DS	(0018,1530)
Detector Secondary Angle	DS	(0018,1531)
Detector Temperature	DS	(0018,7001)
Detector Time Since Last Exposure	DS	(0018,7012)
Detector Type	CS	(0018,7004)
Detector Vector	US	(0054,0020)
Device Description	LO	(0050,0020)
Device Diameter	DS	(0050,0016)
Device Diameter Units	CS	(0050,0017)
Device Length	DS	(0050,0014)
Device Sequence	SQ	(0050,0010)
Device Serial Number	LO	(0018,1000)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Device Volume	DS	(0050,0018)
Diaphragm Position	DS	(3002,0034)
Digital Image Format Acquired	LO	(0018,1023)
Digital Signature DateTime	DT	(0400,0105)
Digital Signature UID	UI	(0400,0100)
Digital Signatures Sequence	SQ	(FFFA,FFFA)
Digitizing Device Transport Direction	CS	(0018,2020)
Directory Record Sequence	SQ	(0004,1220)
Directory Record Type	CS	(0004,1430)
Discharge Date	DA	(0038,0030)
Discharge Diagnosis Code Sequence	SQ	(0038,0044)
Discharge Diagnosis Description	LO	(0038,0040)
Discharge Time	TM	(0038,0032)
Display Window Label Vector	SH	(0018,2006)
Displayed Area Bottom Right Hand Corner	SL	(0070,0053)
Displayed Area Selection Sequence	SQ	(0070,005A)
Displayed Area Top Left Hand Corner	SL	(0070,0052)
Distance Source to Detector	DS	(0018,1110)
Distance Source to Entrance	DS	(0040,0306)
Distance Source to Patient	DS	(0018,1111)
Distance Source to Support	DS	(0040,0307)
Distribution Address	LO	(4008,011A)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Distribution Name	PN	(4008,0119)
Doppler Correction Angle	FD	(0018,6034)
Doppler Sample Volume X Position	UL	(0018,6038)
Doppler Sample Volume Y Position	UL	(0018,603A)
Dose Calibration Factor	DS	(0054,1322)
Dose Comment	LO	(3004,0006)
Dose Grid Scaling	DS	(3004,000E)
Dose Rate Delivered	DS	(3008,0048)
Dose Rate Set	DS	(300A,0115)
Dose Reference Description	LO	(300A,0016)
Dose Reference Number	IS	(300A,0012)
Dose Reference Point Coordinates	DS	(300A,0018)
Dose Reference Sequence	SQ	(300A,0010)
Dose Reference Structure Type	CS	(300A,0014)
Dose Reference Type	CS	(300A,0020)
Dose Summation Type	CS	(3004,000A)
Dose Type	CS	(3004,0004)
Dose Units	CS	(3004,0002)
Dose Value	DS	(3004,0012)
DVH Data	DS	(3004,0058)
DVH Dose Scaling	DS	(3004,0052)
DVH Maximum Dose	DS	(3004,0072)
DVH Mean Dose	DS	(3004,0074)
DVH Minimum Dose	DS	(3004,0070)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
DVH Normalization Dose Value	DS	(3004,0042)
DVH Normalization Point	DS	(3004,0040)
DVH Number of Bins	IS	(3004,0056)
DVH Referenced ROI Sequence	SQ	(3004,0060)
DVH ROI Contribution Type	CS	(3004,0062)
DVH Sequence	SQ	(3004,0050)
DVH Type	CS	(3004,0001)
DVH Volume Units	CS	(3004,0054)
Echo Number(s)	IS	(0018,0086)
Echo Time	DS	(0018,0081)
Echo Train Length	IS	(0018,0091)
Effective Series Duration	DS	(0018,0072)
Empty Image Density	CS	(2010,0110)
End Cumulative Meterset Weight	DS	(300C,0009)
End Meterset	DS	(3008,007A)
Energy Window Information Sequence	SQ	(0054,0012)
Energy Window Lower Limit	DS	(0054,0014)
Energy Window Name	SH	(0054,0018)
Energy Window Number	US	(0054,0308)
Energy Window Range Sequence	SQ	(0054,0013)
Energy Window Upper Limit	DS	(0054,0015)
Energy Window Vector	US	(0054,0010)
Entrance Dose	US	(0040,0302)
Entrance Dose in mGy	DS	(0040,8302)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Estimated Radiographic Magnification Factor	DS	(0018,1114)
Ethnic Group	SH	(0010,2160)
Event Elapsed Time(s)	DS	(0008,2130)
Event Timer Name(s)	LO	(0008,2132)
Execution Status	CS	(2100,0020)
Execution Status Info	CS	(2100,0030)
Expected Completion Date and Time	DT	(0040,4011)
Exposed Area	US	(0040,0303)
Exposure	IS	(0018,1152)
Exposure Control Mode	CS	(0018,7060)
Exposure Control Mode Description	LT	(0018,7062)
Exposure Dose Sequence	SQ	(0040,030E)
Exposure in uAs	IS	(0018,1153)
Exposure Sequence	SQ	(3002,0030)
Exposure Status	CS	(0018,7064)
Exposure Time	IS	(0018,1150)
Exposure Time in μ S	DS	(0018,8150)
Exposures on Detector Since Last Calibration	IS	(0018,7010)
Exposures on Detector Since Manufactured	IS	(0018,7011)
Exposures on Plate	US	(0018,1404)
Failed SOP Instance UID List	UI	(0008,0058)
Failed SOP Sequence	SQ	(0008,1198)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Failure Reason	US	(0008,1197)
Field of View Dimension(s)	IS-2	(0018,1149)
Field of View Horizontal Flip	CS	(0018,7034)
Field of View Origin	DS	(0018,7030)
Field of View Rotation	DS	(0018,7032)
Field of View Shape	CS	(0018,1147)
File Meta Information Version	OB	(0002,0001)
File-set Consistency Flag	US	(0004,1212)
File-set Descriptor File ID	CS	(0004,1141)
File-set ID	CS	(0004,1130)
Filler Order Number / Imaging Service Request	LO	(0040,2017)
Film Box Content Sequence	SQ	(2130,0030)
Film Consumption Sequence	SQ	(0040,0321)
Film Destination	CS	(2000,0040)
Film Orientation	CS	(2010,0040)
Film Session Label	LO	(2000,0050)
Film Size ID	CS	(2010,0050)
Filter High Frequency	DS	(003A,0221)
Filter Low Frequency	DS	(003A,0220)
Filter Material	CS	(0018,7050)
Filter Thickness Maximum	DS	(0018,7054)
Filter Thickness Minimum	DS	(0018,7052)
Filter Type	SH	(0018,1160)
Final Cumulative Meterset Weight	DS	(300A,010E)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Final Cumulative Time Weight	DS	(300A,02C8)
First Treatment Date	DA	(3008,0054)
Fixation Device Description	ST	(300A,0196)
Fixation Device Label	SH	(300A,0194)
Fixation Device Position	SH	(300A,0198)
Fixation Device Sequence	SQ	(300A,0190)
Fixation Device Type	CS	(300A,0192)
Flip Angle	DS	(0018,1314)
Focal Distance	IS	(0018,1182)
Focal Spot(s)	DS	(0018,1190)
Focus Depth	DS	(0018,5012)
Fraction Group Number	IS	(300A,0071)
Fraction Group Sequence	SQ	(300A,0070)
Fraction Group Summary Sequence	SQ	(3008,0220)
Fraction Group Type	CS	(3008,0224)
Fraction Number	IS	(3002,0029)
Fraction Pattern	LT	(300A,007B)
Fraction Status Summary Sequence	SQ	(3008,0240)
Frame Delay	DS	(0018,1066)
Frame Increment Pointer	AT	(0028,0009)
Frame Label Vector	SH	(0018,2002)
Frame Numbers of Interest(FOI)	US	(0028,6020)
Frame of Reference Relationship Sequence	SQ	(3006,00C0)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Frame of Reference Transformation Comment	LO	(3006,00C8)
Frame of Reference Transformation Matrix	DS	(3006,00C6)
Frame of Reference Transformation Type	CS	(3006,00C4)
Frame of Reference UID	UI	(0020,0052)
Frame Primary Angle Vector	DS	(0018,2003)
Frame Reference Time	DS	(0054,1300)
Frame Secondary Angle Vector	DS	(0018,2004)
Frame Time	DS	(0018,1063)
Frame Time Vector	DS	(0018,1065)
Frame(s) of Interest Description	LO	(0028,6022)
Framing Type	LO	(0018,1064)
Gantry Angle	DS	(300A,011E)
Gantry Angle Tolerance	DS	(300A,0044)
Gantry Rotation Direction	CS	(300A,011F)
Gantry/Detector Slew	DS	(0018,1121)
Gantry/Detector Tilt	DS	(0018,1120)
Gated Information Sequence	SQ	(0054,0062)
General Purpose Performed Procedure Step Status	CS	(0040,4002)
General Purpose Scheduled Procedure Step Priority	CS	(0040,4003)
General Purpose Scheduled Procedure Step Status	CS	(0040,4001)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Generator Power	IS	(0018,1170)
Graphic Annotation Sequence	SQ	(0070,0001)
Graphic Annotation Units	CS	(0070,0005)
Graphic Data	FL	(0070,0022)
Graphic Dimensions	US	(0070,0020)
Graphic Filled	CS	(0070,0024)
Graphic Layer	CS	(0070,0002)
Graphic Layer Description	LO	(0070,0068)
Graphic Layer Order	IS	(0070,0062)
Graphic Layer Recommended Display Grayscale Value	US	(0070,0066)
Graphic Layer Recommended Display RGB Value	US	(0070,0067)
Graphic Layer Sequence	SQ	(0070,0060)
Graphic Object Sequence	SQ	(0070,0009)
Graphic Type	CS	(0070,0023)
Green Palette Color Lookup Table Data	OW	(0028,1202)
Green Palette Color Lookup Table Descriptor	US or SS	(0028,1102)
Grid	CS	(0018,1166)
Grid Absorbing Material	LT	(0018,7040)
Grid Aspect Ratio	IS	(0018,7046)
Grid Focal Distance	DS	(0018,704C)
Grid Frame Offset Vector	DS	(3004,000C)
Grid Period	DS	(0018,7048)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Grid Pitch	DS	(0018,7044)
Grid Spacing Material	LT	(0018,7041)
Grid Thickness	DS	(0018,7042)
Group Length	UL	(0002,0000)
Group Length	UL	(0004,0000)
Half Value Layer	DS	(0040,0314)
Hardcopy Creation Device ID	LO	(0018,1011)
Hardcopy Device Manufacturer's Model Name	LO	(0018,101B)
Hardcopy Device Manufacturer	LO	(0018,1017)
Hardcopy Device Software Version	LO	(0018,101A)
Heart Rate	IS	(0018,1088)
High Bit	US	(0028,0102)
High R-R Value	IS	(0018,1082)
High-Dose Technique Type	CS	(300A,00C7)
Histogram Bin Width	US	(0060,3008)
Histogram Data	UL	(0060,3020)
Histogram Explanation	LO	(0060,3010)
Histogram First Bin Value	US or SS	(0060,3004)
Histogram Last Bin Value	US or SS	(0060,3006)
Histogram Number of Bins	US	(0060,3002)
Histogram Sequence	SQ	(0060,3000)
Human Performer Code Sequence	SQ	(0040,4009)
Human Performer's Name	PN	(0040,4037)
Human Performer's Organization	LO	(0040,4036)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Icon Image Sequence	SQ	(0088,0200)
Identical Documents Sequence	SQ	(0040,A525)
Illumination	US	(2010,015E)
Image Area Dose Product	DS	(0018,115E)
Image Box Content Sequence	SQ	(2130,0040)
Image Box Presentation LUT Flag	CS	(2000,006A)
Image Center Point Coordinates Sequence	SQ	(0040,071A)
Image Comments	LT	(0020,4000)
Image Display Format	ST	(2010,0010)
Image Frame Origin	US	(60xx,0051)
Image Horizontal Flip	CS	(0070,0041)
Image ID	SH	(0054,0400)
Image Index	US	(0054,1330)
Image Laterality	CS	(0020,0062)
Image Orientation (Patient)	DS	(0020,0037)
Image Overlay Box Content Sequence	SQ	(2130,0060)
Image Overlay Flag	CS	(2000,0067)
Image Plane Pixel Spacing	DS	(3002,0011)
Image Position	US	(2020,0010)
Image Position (Patient)	DS	(0020,0032)
Image Rotation	US	(0070,0042)
Image Transformation Matrix	DS	(0018,5210)
Image Translation Vector	DS	(0018,5212)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Image Trigger Delay	DS	(0018,1067)
Image Type	CS	(0008,0008)
Imaged Nucleus	SH	(0018,0085)
Imager Pixel Spacing	DS	(0018,1164)
Images in Acquisition	IS	(0020,1002)
Imaging Device-Specific Acquisition Parameters	LO	(300A,00CC)
Imaging Frequency	DS	(0018,0084)
Imaging Service Request Comments	LT	(0040,2400)
Implant Present	CS	(0028,1300)
Implementation Class UID	UI	(0002,0012)
Implementation Version Name	SH	(0002,0013)
Impressions	ST	(4008,0300)
Input Availability Flag	CS	(0040,4020)
Input Information Sequence	SQ	(0040,4021)
Instance Availability	CS	(0008,0056)
Instance Creation Date	DA	(0008,0012)
Instance Creation Time	TM	(0008,0013)
Instance Creator UID	UI	(0008,0014)
Instance Number	IS	(0020,0013)
Institution Address	ST	(0008,0081)
Institution Code Sequence	SQ	(0008,0082)
Institution Name	LO	(0008,0080)
Institutional Department Name	LO	(0008,1040)
Intensifier Size	DS	(0018,1162)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Inter-marker Distance	DS	(0050,0019)
Interpretation Approval Date	DA	(4008,0112)
Interpretation Approval Time	TM	(4008,0113)
Interpretation Approver Sequence	SQ	(4008,0111)
Interpretation Author	PN	(4008,010C)
Interpretation Diagnosis Code Sequence	SQ	(4008,0117)
Interpretation Diagnosis Description	LT	(4008,0115)
Interpretation ID	SH	(4008,0200)
Interpretation ID Issuer	LO	(4008,0202)
Interpretation Recorded Date	DA	(4008,0100)
Interpretation Recorded Time	TM	(4008,0101)
Interpretation Recorder	PN	(4008,0102)
Interpretation Status ID	CS	(4008,0212)
Interpretation Text	ST	(4008,010B)
Interpretation Transcriber	PN	(4008,010A)
Interpretation Transcription Date	DA	(4008,0108)
Interpretation Transcription Time	TM	(4008,0109)
Interpretation Type ID	CS	(4008,0210)
Intervals Acquired	IS	(0018,1083)
Intervals Rejected	IS	(0018,1084)
Intervention Drug Code Sequence	SQ	(0018,0029)
Intervention Drug Dose	DS	(0018,0028)
Intervention Drug Information Sequence	SQ	(0018,0026)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Intervention Drug Name	LO	(0018,0034)
Intervention Drug Start Time	TM	(0018,0035)
Intervention Drug Stop Time	TM	(0018,0027)
Interventional Status	CS	(0018,0038)
Interventional Therapy Sequence	SQ	(0018,0036)
Inversion Time	DS	(0018,0082)
Isocenter Position	DS	(300A,012C)
Issue Date of Imaging Service Request	DA	(0040,2004)
Issue Time of Imaging Service Request	TM	(0040,2005)
Issuer of Admission ID	LO	(0038,0011)
Issuer of Patient ID	LO	(0010,0021)
Item	Note: The VR for Item does not exist. See PS 3.5 for explanation.	(FFFE,E000)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Item Delimitation Item	Note: The VR for Item Delimitation Item does not exist. See PS 3.5 for explanation.	(FFFE,E00D)
Item Number	IS	(0020,0019)
IVUS Acquisition	CS	(0018,3100)
IVUS Gated Rate	DS	(0018,3102)
IVUS Pullback Rate	DS	(0018,3101)
IVUS Pullback Start Frame Number	IS	(0018,3103)
IVUS Pullback Stop Frame Number	IS	(0018,3104)
KVP	DS	(0018,0060)
Largest Image Pixel Value	US or SS	(0028,0107)
Largest Image Pixel Value in Plane	US or SS	(0028,0111)
Largest Pixel Value in Series	US or SS	(0028,0109)
Last Menstrual Date	DA	(0010,21D0)
Laterality	CS	(0020,0060)
Leaf Position Boundaries	DS	(300A,00BE)
Leaf/Jaw Positions	DS	(300A,011C)
Lesion Number	IS	(0018,3105)
Lookup Table (LUT) Number	IS	(0020,0026)
Lossy Image Compression	CS	(0028,2110)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Lossy Image Compression Ratio	DS	(0028,2112)
Low R-R Value	IS	(0018,1081)
LUT Data	US or SS or OW	(0028,3006)
LUT Descriptor	US or SS	(0028,3002)
LUT Explanation	LO	(0028,3003)
MAC Algorithm	CS	(0400,0015)
MAC Calculation Transfer Syntax UID	UI	(0400,0010)
MAC ID number	US	(0400,0005)
MAC Parameters Sequence	SQ	(4FFE,0001)
Magnetic Field Strength	DS	(0018,0087)
Magnification Type	CS	(2010,0060)
Magnify to Number of Columns	US	(2040,0074)
Manufacturer	LO	(0008,0070)
Manufacturer's Model Name	LO	(0008,1090)
Mapping Resource	CS	(0008,0105)
Mask Frame Numbers	US	(0028,6110)
Mask Operation	CS	(0028,6101)
Mask Operation Explanation	ST	(0028,6190)
Mask Pointer(s)	US	(0028,6030)
Mask Sub-pixel Shift	FL	(0028,6114)
Mask Subtraction Sequence	SQ	(0028,6100)
Material ID	SH	(300A,00E1)
Max Density	US	(2010,0130)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Maximum Collated Films	IS	(2010,0154)
Maximum Coordinate Value	US	(50xx,0105)
Maximum Memory Allocation	IS	(2000,0061)
Measured Dose Description	ST	(3008,0012)
Measured Dose Reference Number	IS	(3008,0064)
Measured Dose Reference Sequence	SQ	(3008,0010)
Measured Dose Type	CS	(3008,0014)
Measured Dose Value	DS	(3008,0016)
Measured Value Sequence	SQ	(0040,A300)
Measurement Units Code Sequence	SQ	(0040,08EA)
Measuring Units Sequence	SQ	(0040,0295)
Mechanical Index	DS	(0018,5022)
Media Installed Sequence	SQ	(2000,00A2)
Media Storage SOP Class UID	UI	(0002,0002)
Media Storage SOP Instance UID	UI	(0002,0003)
Medical Alerts	LO	(0010,2000)
Medical Record Locator	LO	(0010,1090)
Medium Type	CS	(2000,0030)
Memory Allocation	IS	(2000,0060)
Memory Bit Depth	US	(2000,00A0)
Meterset Exposure	DS	(3002,0032)
Military Rank	LO	(0010,1080)
Min Density	US	(2010,0120)
Minimum Coordinate Value	US	(50xx,0104)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Modalities in Study	CS	(0008,0061)
Modality	CS	(0008,0060)
Modality LUT Sequence	SQ	(0028,3000)
Modality LUT Type	LO	(0028,3004)
Modifier Code Sequence	SQ	(0040,A195)
Most Recent Treatment Date	DA	(3008,0056)
MR Acquisition Type	CS	(0018,0023)
MRDR Directory Record Offset	UL	(0004,1504)
Multiple Copies Flag	CS	(0040,4006)
Multiplex Group Label	SH	(003A,0020)
Multiplex Group Time Offset	DS	(0018,1068)
Name of Physician(s) Reading Study	PN	(0008,1060)
Names of Intended Recipients of Results	PN	(0040,1010)
Nominal Beam Energy	DS	(300A,0114)
Nominal Beam Energy Unit	CS	(300A,0015)
Nominal Interval	IS	(0018,1062)
Nominal Prior Dose	DS	(300A,001A)
Nominal Scanned Pixel Spacing	DS	(0018,2010)
Non-DICOM Output Code Sequence	SQ	(0040,4032)
Normalization Point	DS	(3004,0008)
Notch Filter Bandwidth	DS	(003A,0223)
Notch Filter Frequency	DS	(003A,0222)
Number of Averages	DS	(0018,0083)
Number of Beams	IS	(300A,0080)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Number of Blocks	IS	(300A,00F0)
Number of Boli	IS	(300A,00ED)
Number of Brachy Application Setups	IS	(300A,00A0)
Number of Channels	US	(50xx,2004)
Number of Compensators	IS	(300A,00E0)
Number of Contour Points	IS	(3006,0046)
Number of Control Points	IS	(300A,0110)
Number of Copies	IS	(2000,0010)
Number of Detectors	US	(0054,0021)
Number of Energy Windows	US	(0054,0011)
Number of Event Timers	IS	(0008,2129)
Number of Films	IS	(2100,0170)
Number of Fraction Pattern Digits Per Day	IS	(300A,0079)
Number of Fractions Delivered	IS	(3008,005A)
Number of Fractions Planned	IS	(300A,0078)
Number of Frames	IS	(0028,0008)
Number of Frames in Overlay	IS	(60xx,0015)
Number of Frames in Phase	US	(0054,0033)
Number of Frames in Rotation	US	(0054,0053)
Number of Graphic Points	US	(0070,0021)
Number of Leaf/Jaw Pairs	IS	(300A,00BC)
Number of Patient Related Instances	IS	(0020,1204)
Number of Patient Related Series	IS	(0020,1202)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Number of Patient Related Studies	IS	(0020,1200)
Number of Phase Encoding Steps	IS	(0018,0089)
Number of Phases	US	(0054,0031)
Number of Points	US	(50xx,0010)
Number of Pulses	IS	(300A,028A)
Number of References	UL	(0004,1600)
Number of Rotations	US	(0054,0051)
Number of R-R Intervals	US	(0054,0061)
Number of Samples	UL	(50xx,2006)
Number of Series Related Instances	IS	(0020,1209)
Number of Slices	US	(0054,0081)
Number of Stages	IS	(0008,2124)
Number of Study Related Instances	IS	(0020,1208)
Number of Study Related Series	IS	(0020,1206)
Number of Table Break Points	UL	(0018,6050)
Number of Table Entries	UL	(0018,6056)
Number of Temporal Positions	IS	(0020,0105)
Number of Time Slices	US	(0054,0101)
Number of Time Slots	US	(0054,0071)
Number of Tomosynthesis Source Images	IS	(0018,1495)
Number of Triggers in Phase	US	(0054,0211)
Number of Views in Stage	IS	(0008,212A)
Number of Waveform Channels	US	(003A,0005)
Number of Waveform Samples	UL	(003A,0010)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Number of Wedges	IS	(300A,00D0)
Numeric Value	DS	(0040,A30A)
Observation DateTime	DT	(0040,A032)
Observation Number	IS	(3006,0082)
Occupation	SH	(0010,2180)
Offset of Referenced Lower-Level Directory Entity	UL	(0004,1420)
Offset of the First Directory Record of the Root Directory Entity	UL	(0004,1200)
Offset of the Last Directory Record of the Root Directory Entity	UL	(0004,1202)
Offset of the Next Directory Record	UL	(0004,1400)
Operators' Name	PN	(0008,1070)
Order Callback Phone Number	SH	(0040,2010)
Order Entered By	PN	(0040,2008)
Order Enterer's Location	SH	(0040,2009)
Organ at Risk Full-volume Dose	DS	(300A,002A)
Organ at Risk Limit Dose	DS	(300A,002B)
Organ at Risk Maximum Dose	DS	(300A,002C)
Organ at Risk Overdose Volume Fraction	DS	(300A,002D)
Organ Dose	DS	(0040,0316)
Organ Exposed	CS	(0040,0318)
Original Image Sequence	SQ	(2130,00C0)
Originator	AE	(2100,0070)
Other Magnification Types Available	CS	(2010,00A7)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Other Media Available Sequence	SQ	(2000,00A4)
Other Patient IDs	LO	(0010,1000)
Other Patient Names	PN	(0010,1001)
Other Smoothing Types Available	CS	(2010,00A9)
Other Study Numbers	IS	(0020,1070)
Output Information Sequence	SQ	(0040,4033)
Output Power	SH	(0018,5000)
Overlay Activation Layer	CS	(60xx,1001)
Overlay Background Density	CS	(2040,0082)
Overlay Bit Position	US	(60xx,0102)
Overlay Bits Allocated	US	(60xx,0100)
Overlay Columns	US	(60xx,0011)
Overlay Data	OB or OW	(60xx,3000)
Overlay Date	DA	(0008,0024)
Overlay Description	LO	(60xx,0022)
Overlay Foreground Density	CS	(2040,0080)
Overlay Label	LO	(60xx,1500)
Overlay Magnification Type	CS	(2040,0060)
Overlay Number	IS	(0020,0022)
Overlay or Image Magnification	CS	(2040,0072)
Overlay Origin	SS	(60xx,0050)
Overlay Pixel Data Sequence	SQ	(2040,0020)
Overlay Plane Origin	US	(60xx,0052)
Overlay Planes	US	(60xx,0012)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Overlay Rows	US	(60xx,0010)
Overlay Smoothing Type	CS	(2040,0070)
Overlay Subtype	LO	(60xx,0045)
Overlay Time	TM	(0008,0034)
Overlay Type	CS	(60xx,0040)
Override Parameter Pointer	AT	(3008,0062)
Override Reason	ST	(3008,0066)
Override Sequence	SQ	(3008,0060)
Owner ID	SH	(2100,0160)
Page Number Vector	IS	(0018,2001)
Palette Color Lookup Table UID	UI	(0028,1199)
Partial View	CS	(0028,1350)
Partial View Description	ST	(0028,1351)
Patient Additional Position	LO	(300A,0184)
Patient Comments	LT	(0010,4000)
Patient Gantry Relationship Code Sequence	SQ	(0054,0414)
Patient ID	LO	(0010,0020)
Patient Orientation	CS	(0020,0020)
Patient Orientation Code Sequence	SQ	(0054,0410)
Patient Orientation Modifier Code Sequence	SQ	(0054,0412)
Patient Position	CS	(0018,5100)
Patient Setup Number	IS	(300A,0182)
Patient Setup Sequence	SQ	(300A,0180)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Patient State	LO	(0038,0500)
Patient Support Angle	DS	(300A,0122)
Patient Support Angle Tolerance	DS	(300A,004C)
Patient Support Rotation Direction	CS	(300A,0123)
Patient Transport Arrangements	LO	(0040,1004)
Patient's Name	PN	(0010,0010)
Patient's Primary Language Code Modifier Sequence	SQ	(0010,0102)
Patient's Primary Language Code Sequence	SQ	(0010,0101)
Patient's Telephone Numbers	SH	(0010,2154)
Patient's Address	LO	(0010,1040)
Patient's Age	AS	(0010,1010)
Patient's Birth Date	DA	(0010,0030)
Patient's Birth Name	PN	(0010,1005)
Patient's Birth Time	TM	(0010,0032)
Patient's Institution Residence	LO	(0038,0400)
Patient's Insurance Plan Code Sequence	SQ	(0010,0050)
Patient's Mother's Birth Name	PN	(0010,1060)
Patient's Religious Preference	LO	(0010,21F0)
Patient's Sex	CS	(0010,0040)
Patient's Size	DS	(0010,1020)
Patient's Weight	DS	(0010,1030)
Pause Between Frames	IS	(0054,0038)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Percent Phase Field of View	DS	(0018,0094)
Percent Sampling	DS	(0018,0093)
Performed Location	SH	(0040,0243)
Performed Procedure Code Sequence	SQ	(0040,A372)
Performed Procedure Step Description	LO	(0040,0254)
Performed Procedure Step End Date	DA	(0040,0250)
Performed Procedure Step End Time	TM	(0040,0251)
Performed Procedure Step ID	SH	(0040,0253)
Performed Procedure Step Start Date	DA	(0040,0244)
Performed Procedure Step Start Time	TM	(0040,0245)
Performed Procedure Step Status	CS	(0040,0252)
Performed Procedure Type Description	LO	(0040,0255)
Performed Processing Applications Code Sequence	SQ	(0040,4007)
Performed Protocol Code Sequence	SQ	(0040,0260)
Performed Series Sequence	SQ	(0040,0340)
Performed Station AE Title	AE	(0040,0241)
Performed Station Class Code Sequence	SQ	(0040,4029)
Performed Station Geographic Location Code Sequence	SQ	(0040,4030)
Performed Station Name	SH	(0040,0242)
Performed Station Name Code Sequence	SQ	(0040,4028)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Performed Workitem Code Sequence	SQ	(0040,4019)
Performing Physician's Name	PN	(0008,1050)
Person Name	PN	(0040,A123)
Pertinent Other Evidence Sequence	SQ	(0040,A385)
Phase Delay	IS	(0054,0036)
Phase Encoding Direction	CS	(0018,1312)
Phase Information Sequence	SQ	(0054,0032)
Phase Vector	US	(0054,0030)
Phosphor Type	LO	(0018,1261)
Photometric Interpretation	CS	(0028,0004)
Phototimer Setting	DS	(0018,7065)
Physical Delta X	FD	(0018,602C)
Physical Delta Y	FD	(0018,602E)
Physical Units X Direction	US	(0018,6024)
Physical Units Y Direction	US	(0018,6026)
Physician Approving Interpretation	PN	(4008,0114)
Physician(s) of Record	PN	(0008,1048)
Pixel Aspect Ratio	IS	(0028,0034)
Pixel Bandwidth	DS	(0018,0095)
Pixel Component Data Type	US	(0018,604E)
Pixel Component Mask	UL	(0018,6046)
Pixel Component Organization	US	(0018,6044)
Pixel Component Physical Units	US	(0018,604C)
Pixel Component Range Start	UL	(0018,6048)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Pixel Component Range Stop	UL	(0018,604A)
Pixel Data	OW or OB	(7FE0,0010)
Pixel Intensity Relationship	CS	(0028,1040)
Pixel Intensity Relationship Sign	SS	(0028,1041)
Pixel Padding Value	US or SS	(0028,0120)
Pixel Representation	US	(0028,0103)
Pixel Spacing	DS	(0028,0030)
Pixel Spacing Sequence	SQ	(0040,08D8)
Placer Order Number / Imaging Service Request	LO	(0040,2016)
Planar Configuration	US	(0028,0006)
Planes	US	(0028,0012)
Planned Verification Image Sequence	SQ	(300A,00CA)
Plate ID	LO	(0018,1004)
Plate Type	SH	(0018,1260)
Polarity	CS	(2020,0020)
Position Reference Indicator	LO	(0020,1040)
Positioner Motion	CS	(0018,1500)
Positioner Primary Angle	DS	(0018,1510)
Positioner Primary Angle Increment	DS	(0018,1520)
Positioner Secondary Angle	DS	(0018,1511)
Positioner Secondary Angle Increment	DS	(0018,1521)
Positioner Type	CS	(0018,1508)
Postprocessing Function	LO	(0018,5021)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Predecessor Documents Sequence	SQ	(0040,A360)
Preferred Playback Sequencing	US	(0018,1244)
Pregnancy Status	US	(0010,21C0)
Pre-Medication	LO	(0040,0012)
Prescription Description	ST	(300A,000E)
Presentation Creation Date	DA	(0070,0082)
Presentation Creation Time	TM	(0070,0083)
Presentation Creator's Name	PN	(0070,0084)
Presentation Description	LO	(0070,0081)
Presentation Intent Type	CS	(0008,0068)
Presentation Label	CS	(0070,0080)
Presentation LUT Content Sequence	SQ	(2130,0080)
Presentation LUT Flag	CS	(2000,0069)
Presentation LUT Sequence	SQ	(2050,0010)
Presentation LUT Shape	CS	(2050,0020)
Presentation Pixel Aspect Ratio	IS	(0070,0102)
Presentation Pixel Magnification Ratio	FL	(0070,0103)
Presentation Pixel Spacing	DS	(0070,0101)
Presentation Size Mode	CS	(0070,0100)
Primary Anatomic Structure Modifier Sequence	SQ	(0008,2230)
Primary Anatomic Structure Sequence	SQ	(0008,2228)
Primary Dosimeter Unit	CS	(300A,00B3)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Primary(Prompts) Counts Accumulated	IS	(0054,1310)
Print Job Description Sequence	SQ	(2120,0050)
Print Job ID	SH	(2100,0010)
Print Management Capabilities Sequence	SQ	(2130,0010)
Print Priority	CS	(2000,0020)
Print Queue ID	SH	(2110,0099)
Printer Characteristics Sequence	SQ	(2130,0015)
Printer Configuration Sequence	SQ	(2000,001E)
Printer Name	LO	(2110,0030)
Printer Pixel Spacing	DS	(2010,0376)
Printer Resolution ID	CS	(2010,0052)
Printer Status	CS	(2110,0010)
Printer Status Info	CS	(2110,0020)
Printing Bit Depth	US	(2000,00A1)
Private Coding Scheme Creator UID	UI	(0008,010C)
Private Information	OB	(0002,0102)
Private Information Creator UID	UI	(0002,0100)
Private Record UID	UI	(0004,1432)
Procedure Code Sequence	SQ	(0008,1032)
Processing Function	LO	(0018,5020)
Projection Eponymous Name Code Sequence	SQ	(0018,5104)
Proposed Study Sequence	SQ	(2130,00A0)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Protocol Name	LO	(0018,1030)
Pulse Repetition Frequency	UL	(0018,6032)
Pulse Repetition Interval	DS	(300A,028C)
Purpose of Reference Code Sequence	SQ	(0040,A170)
PVC Rejection	LO	(0018,1085)
Quality Control Image	CS	(0028,0300)
Quantity	DS	(0040,0294)
Quantity Sequence	SQ	(0040,0293)
Query/Retrieve Level	CS	(0008,0052)
Queue Status	CS	(2120,0010)
R Wave Pointer	US	(0028,6040)
Radial Position	DS	(0018,1142)
Radiation Machine Name	SH	(3002,0020)
Radiation Machine SAD	DS	(3002,0022)
Radiation Machine SSD	DS	(3002,0024)
Radiation Mode	CS	(0018,115A)
Radiation Setting	CS	(0018,1155)
Radiation Type	CS	(300A,00C6)
Radionuclide Code Sequence	SQ	(0054,0300)
Radionuclide Half Life	DS	(0018,1075)
Radionuclide Positron Fraction	DS	(0018,1076)
Radionuclide Total Dose	DS	(0018,1074)
Radiopharmaceutical	LO	(0018,0031)
Radiopharmaceutical Code Sequence	SQ	(0054,0304)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Radiopharmaceutical Information Sequence	SQ	(0054,0016)
Radiopharmaceutical Route	LO	(0018,1070)
Radiopharmaceutical Specific Activity	DS	(0018,1077)
Radiopharmaceutical Start Time	TM	(0018,1072)
Radiopharmaceutical Stop Time	TM	(0018,1073)
Radiopharmaceutical Volume	DS	(0018,1071)
Radius of Circular Collimator	IS	(0018,1712)
Radius of Circular Shutter	IS	(0018,1612)
Randoms Correction Method	CS	(0054,1100)
Reason for Study	LO	(0032,1030)
Reason for the Imaging Service Request	LO	(0040,2001)
Reason for the Requested Procedure	LO	(0040,1002)
Receiving Coil	SH	(0018,1250)
Recommended Display Frame Rate	IS	(0008,2144)
Recommended Viewing Mode	CS	(0028,1090)
Reconstruction Diameter	DS	(0018,1100)
Reconstruction Method	LO	(0054,1103)
Record In-use Flag	US	(0004,1410)
Recorded Block Sequence	SQ	(3008,00D0)
Recorded Brachy Accessory Device Sequence	SQ	(3008,0120)
Recorded Channel Sequence	SQ	(3008,0130)
Recorded Channel Shield Sequence	SQ	(3008,0150)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Recorded Compensator Sequence	SQ	(3008,00C0)
Recorded Source Applicator Sequence	SQ	(3008,0140)
Recorded Source Sequence	SQ	(3008,0100)
Recorded Wedge Sequence	SQ	(3008,00B0)
Rectification Type	CS	(0018,1156)
Red Palette Color Lookup Table Data	OW	(0028,1201)
Red Palette Color Lookup Table Descriptor	US or SS	(0028,1101)
Reference Air Kerma Rate	DS	(300A,022A)
Reference Image Number	IS	(300A,00C8)
Reference Pixel Physical Value X	FD	(0018,6028)
Reference Pixel Physical Value Y	FD	(0018,602A)
Reference Pixel X ₀	SL	(0018,6020)
Reference Pixel Y ₀	SL	(0018,6022)
Reference to Recorded Sound	LO	(4008,0103)
Referenced Basic Annotation Box Sequence	SQ	(2010,0520)
Referenced Beam Number	IS	(300C,0006)
Referenced Beam Sequence	SQ	(300C,0004)
Referenced Block Number	IS	(300C,00E0)
Referenced Bolus Sequence	SQ	(300C,00B0)
Referenced Brachy Accessory Device Number	IS	(3008,0122)
Referenced Brachy Application Setup Number	IS	(300C,000C)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Referenced Brachy Application Setup Sequence	SQ	(300C,000A)
Referenced Calculated Dose Reference Number	IS	(3008,0092)
Referenced Calculated Dose Reference Sequence	SQ	(3008,0090)
Referenced Channel Shield Number	IS	(3008,0152)
Referenced Compensator Number	IS	(300C,00D0)
Referenced Content Item Identifier	UL	(0040,DB73)
Referenced Control Point Index	IS	(300C,00F0)
Referenced Curve Sequence	SQ	(0008,1145)
Referenced Datetime	DT	(0040,A13A)
Referenced Dose Reference Number	IS	(300C,0051)
Referenced Dose Reference Sequence	SQ	(300C,0050)
Referenced Dose Sequence	SQ	(300C,0080)
Referenced File ID	CS	(0004,1500)
Referenced Film Box Sequence	SQ	(2000,0500)
Referenced Film Session Sequence	SQ	(2010,0500)
Referenced Fraction Group Number	IS	(300C,0022)
Referenced Fraction Group Sequence	SQ	(300C,0020)
Referenced Fraction Number	IS	(3008,0223)
Referenced Frame Number	IS	(0008,1160)
Referenced Frame Numbers	US	(0040,A136)
Referenced Frame of Reference Sequence	SQ	(3006,0010)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Referenced Frame of Reference UID	UI	(3006,0024)
Referenced General Purpose Scheduled Procedure Step Sequence	SQ	(0040,4016)
Referenced General Purpose Scheduled Procedure Step Transaction UID	UI	(0040,4023)
Referenced Image Box Sequence	SQ	(2010,0510)
Referenced Image Sequence	SQ	(0008,1140)
Referenced Instance Sequence	SQ	(0008,114A)
Referenced Interpretation Sequence	SQ	(4008,0050)
Referenced Measured Dose Reference Number	IS	(3008,0082)
Referenced Measured Dose Reference Sequence	SQ	(3008,0080)
Referenced Non-Image Composite SOP Instance Sequence	SQ	(0040,0220)
Referenced Overlay Group	US	(50xx,2610)
Referenced Overlay Plane Groups	US	(2040,0011)
Referenced Overlay Plane Sequence	SQ	(2040,0010)
Referenced Overlay Sequence	SQ	(0008,1130)
Referenced Overlay Sequence	SQ	(50xx,2600)
Referenced Patient Alias Sequence	SQ	(0038,0004)
Referenced Patient Sequence	SQ	(0008,1120)
Referenced Patient Setup Number	IS	(300C,006A)
Referenced Presentation LUT Sequence	SQ	(2050,0500)
Referenced Print Job Sequence	SQ	(2100,0500)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Referenced Print Job Sequence	SQ	(2120,0070)
Referenced Procedure Step Sequence	SQ	(0040,0330)
Referenced Reference Image Number	IS	(300C,0007)
Referenced Reference Image Sequence	SQ	(300C,0042)
Referenced Request Sequence	SQ	(0040,A370)
Referenced Results Sequence	SQ	(0008,1100)
Referenced ROI Number	IS	(3006,0084)
Referenced RT Plan Sequence	SQ	(300C,0002)
Referenced Sample Positions	UL	(0040,A132)
Referenced Series Sequence	SQ	(0008,1115)
Referenced SOP Class UID	UI	(0008,1150)
Referenced SOP Class UID in File	UI	(0004,1510)
Referenced SOP Instance UID	UI	(0008,1155)
Referenced SOP Instance UID in File	UI	(0004,1511)
Referenced SOP Sequence	SQ	(0008,1199)
Referenced Source Applicator Number	IS	(3008,0142)
Referenced Source Number	IS	(300C,000E)
Referenced Stored Print Sequence	SQ	(2000,0510)
Referenced Structure Set Sequence	SQ	(300C,0060)
Referenced Study Component Sequence	SQ	(0008,1111)
Referenced Study Sequence	SQ	(0008,1110)
Referenced Time Offsets	DS	(0040,A138)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Referenced Tolerance Table Number	IS	(300C,00A0)
Referenced Transfer Syntax UID in File	UI	(0004,1512)
Referenced Treatment Record Sequence	SQ	(3008,0030)
Referenced Verification Image Sequence	SQ	(300C,0040)
Referenced Visit Sequence	SQ	(0008,1125)
Referenced Waveform Channels	US	(0040,A0B0)
Referenced Waveform Sequence	SQ	(0008,113A)
Referenced Wedge Number	IS	(300C,00C0)
Referring Physician's Name	PN	(0008,0090)
Referring Physician's Telephone Numbers	SH	(0008,0094)
Referring Physician's Address	ST	(0008,0092)
Reflected Ambient Light	US	(2010,0160)
Region Data Type	US	(0018,6014)
Region Flags	UL	(0018,6016)
Region Location Max X ₁	UL	(0018,601C)
Region Location Max Y ₁	UL	(0018,601E)
Region Location Min X ₀	UL	(0018,6018)
Region Location Min Y ₀	UL	(0018,601A)
Region of Residence	LO	(0010,2152)
Region Spatial Format	US	(0018,6012)
Related Frame of Reference UID	UI	(3006,00C2)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Related RT ROI Observations Sequence	SQ	(3006,00A0)
Relationship Type	CS	(0040,A010)
Relative X-ray Exposure	IS	(0018,1405)
Relevant Information Sequence	SQ	(0040,4022)
Repeat Fraction Cycle Length	IS	(300A,007A)
Repetition Time	DS	(0018,0080)
Reported Values Origin	CS	(3002,000A)
Reporting Priority	SH	(0040,1009)
Representative Frame Number	US	(0028,6010)
Reprojection Method	CS	(0054,1004)
Request Attributes Sequence	SQ	(0040,0275)
Requested Contrast Agent	LO	(0032,1070)
Requested Decimate/Crop Behavior	CS	(2020,0040)
Requested Image Size	DS	(2020,0030)
Requested Image Size Flag	CS	(2020,00A0)
Requested Procedure Code Sequence	SQ	(0032,1064)
Requested Procedure Comments	LT	(0040,1400)
Requested Procedure Description	LO	(0032,1060)
Requested Procedure ID	SH	(0040,1001)
Requested Procedure Location	LO	(0040,1005)
Requested Procedure Priority	SH	(0040,1003)
Requested Resolution ID	CS	(2020,0050)
Requested Subsequent Workitem Code Sequence	SQ	(0040,4031)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Requesting Physician	PN	(0032,1032)
Requesting Service	LO	(0032,1033)
Rescale Intercept	DS	(0028,1052)
Rescale Slope	DS	(0028,1053)
Rescale Type	LO	(0028,1054)
Residual Syringe Counts	IS	(0054,0017)
Resulting General Purpose Performed Procedure Steps Sequence	SQ	(0040,4015)
Results Comments	ST	(4008,4000)
Results Distribution List Sequence	SQ	(4008,0118)
Results ID	SH	(4008,0040)
Results ID Issuer	LO	(4008,0042)
Retrieve AE Title	AE	(0008,0054)
Review Date	DA	(300E,0004)
Review Time	TM	(300E,0005)
Reviewer Name	PN	(300E,0008)
ROI Area	IS	(60xx,1301)
ROI Contour Sequence	SQ	(3006,0039)
ROI Description	ST	(3006,0028)
ROI Display Color	IS	(3006,002A)
ROI Generation Algorithm	CS	(3006,0036)
ROI Generation Description	LO	(3006,0038)
ROI Interpreter	PN	(3006,00A6)
ROI Mean	DS	(60xx,1302)
ROI Name	LO	(3006,0026)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
ROI Number	IS	(3006,0022)
ROI Observation Description	ST	(3006,0088)
ROI Observation Label	SH	(3006,0085)
ROI Physical Properties Sequence	SQ	(3006,00B0)
ROI Physical Property	CS	(3006,00B2)
ROI Physical Property Value	DS	(3006,00B4)
ROI Standard Deviation	DS	(60xx,1303)
ROI Volume	DS	(3006,002C)
Rotation Direction	CS	(0018,1140)
Rotation Information Sequence	SQ	(0054,0052)
Rotation of Scanned Film	DS	(0018,2030)
Rotation Vector	US	(0054,0050)
Route of Admissions	LO	(0038,0016)
Rows	US	(0028,0010)
R-R Interval Vector	US	(0054,0060)
RT Beam Limiting Device Type	CS	(300A,00B8)
RT Dose ROI Sequence	SQ	(3004,0010)
RT Image Description	ST	(3002,0004)
RT Image Label	SH	(3002,0002)
RT Image Name	LO	(3002,0003)
RT Image Orientation	DS	(3002,0010)
RT Image Plane	CS	(3002,000C)
RT Image Position	DS	(3002,0012)
RT Image SID	DS	(3002,0026)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
RT Plan Date	DA	(300A,0006)
RT Plan Description	ST	(300A,0004)
RT Plan Geometry	CS	(300A,000C)
RT Plan Label	SH	(300A,0002)
RT Plan Name	LO	(300A,0003)
RT Plan Relationship	CS	(300A,0055)
RT Plan Time	TM	(300A,0007)
RT Referenced Series Sequence	SQ	(3006,0014)
RT Referenced Study Sequence	SQ	(3006,0012)
RT Related ROI Sequence	SQ	(3006,0030)
RT ROI Identification Code Sequence	SQ	(3006,0086)
RT ROI Interpreted Type	CS	(3006,00A4)
RT ROI Observations Sequence	SQ	(3006,0080)
RT ROI Relationship	CS	(3006,0033)
Safe Position Exit Date	DA	(3008,0162)
Safe Position Exit Time	TM	(3008,0164)
Safe Position Return Date	DA	(3008,0166)
Safe Position Return Time	TM	(3008,0168)
Sample Rate	UL	(50xx,2008)
Samples per Pixel	US	(0028,0002)
Sampling Frequency	DS	(003A,001A)
SAR	DS	(0018,1316)
Scan Arc	DS	(0018,1143)
Scan Length	IS	(0018,1302)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Scan Options	CS	(0018,0022)
Scan Velocity	DS	(0018,1300)
Scanning Sequence	CS	(0018,0020)
Scatter Correction Method	LO	(0054,1105)
Scatter Fraction Factor	DS	(0054,1323)
Scheduled Admission Date	DA	(0038,001A)
Scheduled Admission Time	TM	(0038,001B)
Scheduled Discharge Date	DA	(0038,001C)
Scheduled Discharge Time	TM	(0038,001D)
Scheduled Human Performers Sequence	SQ	(0040,4034)
Scheduled Patient Institution Residence	LO	(0038,001E)
Scheduled Performing Physician's Name	PN	(0040,0006)
Scheduled Procedure Step Description	LO	(0040,0007)
Scheduled Procedure Step End Date	DA	(0040,0004)
Scheduled Procedure Step End Time	TM	(0040,0005)
Scheduled Procedure Step ID	SH	(0040,0009)
Scheduled Procedure Step Location	SH	(0040,0011)
Scheduled Procedure Step Sequence	SQ	(0040,0100)
Scheduled Procedure Step Start Date	DA	(0040,0002)
Scheduled Procedure Step Start Date and Time	DT	(0040,4005)
Scheduled Procedure Step Start Time	TM	(0040,0003)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Scheduled Procedure Step Status	CS	(0040,0020)
Scheduled Processing Applications Code Sequence	SQ	(0040,4004)
Scheduled Protocol Code Sequence	SQ	(0040,0008)
Scheduled Station AE Title	AE	(0040,0001)
Scheduled Station Class Code Sequence	SQ	(0040,4026)
Scheduled Station Geographic Location Code Sequence	SQ	(0040,4027)
Scheduled Station Name	SH	(0040,0010)
Scheduled Station Name Code Sequence	SQ	(0040,4025)
Scheduled Step Attributes Sequence	SQ	(0040,0270)
Scheduled Study Location	LO	(0032,1020)
Scheduled Study Location AE Title(s)	AE	(0032,1021)
Scheduled Study Start Date	DA	(0032,1000)
Scheduled Study Start Time	TM	(0032,1001)
Scheduled Study Stop Date	DA	(0032,1010)
Scheduled Study Stop Time	TM	(0032,1011)
Scheduled Workitem Code Sequence	SQ	(0040,4018)
Secondary Capture Device ID	LO	(0018,1010)
Secondary Capture Device Manufacturer	LO	(0018,1016)
Secondary Capture Device Manufacturer's Model Name	LO	(0018,1018)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Secondary Capture Device Software Version(s)	LO	(0018,1019)
Secondary Counts Accumulated	IS	(0054,1311)
Secondary Counts Type	CS	(0054,1220)
Segmented Blue Palette Color Lookup Table Data	OW	(0028,1223)
Segmented Green Palette Color Lookup Table Data	OW	(0028,1222)
Segmented Red Palette Color Lookup Table Data	OW	(0028,1221)
Sensitivity	DS	(0018,6000)
Sequence Delimitation Item	Note: The VR for Sequence Delimitation Item does not exist. See PS 3.5 for explanation.	(FFFE,E0DD)
Sequence Name	SH	(0018,0024)
Sequence of Ultrasound Regions	SQ	(0018,6011)
Sequence Variant	CS	(0018,0021)
Series Date	DA	(0008,0021)
Series Description	LO	(0008,103E)
Series in Study	IS	(0020,1000)
Series Instance UID	UI	(0020,000E)
Series Number	IS	(0020,0011)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Series Time	TM	(0008,0031)
Series Type	CS	(0054,1000)
Setup Device Description	ST	(300A,01BA)
Setup Device Label	SH	(300A,01B8)
Setup Device Parameter	DS	(300A,01BC)
Setup Device Sequence	SQ	(300A,01B4)
Setup Device Type	CS	(300A,01B6)
Setup Reference Description	ST	(300A,01D0)
Setup Technique	CS	(300A,01B0)
Setup Technique Description	ST	(300A,01B2)
Shielding Device Description	ST	(300A,01A6)
Shielding Device Label	SH	(300A,01A4)
Shielding Device Position	SH	(300A,01A8)
Shielding Device Sequence	SQ	(300A,01A0)
Shielding Device Type	CS	(300A,01A2)
Shutter Left Vertical Edge	IS	(0018,1602)
Shutter Lower Horizontal Edge	IS	(0018,1608)
Shutter Overlay Group	US	(0018,1623)
Shutter Presentation Value	US	(0018,1622)
Shutter Right Vertical Edge	IS	(0018,1604)
Shutter Shape	CS	(0018,1600)
Shutter Upper Horizontal Edge	IS	(0018,1606)
Signature	OB	(0400,0120)
Skip Beats	IS	(0018,1086)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Slice Location	DS	(0020,1041)
Slice Location Vector	DS	(0018,2005)
Slice Sensitivity Factor	DS	(0054,1320)
Slice Thickness	DS	(0018,0050)
Slice Vector	US	(0054,0080)
Slide Identifier	LO	(0040,06FA)
Smallest Image Pixel Value	US or SS	(0028,0106)
Smallest Image Pixel Value in Plane	US or SS	(0028,0110)
Smallest Pixel Value in Series	US or SS	(0028,0108)
Smoking Status	CS	(0010,21A0)
Smoothing Type	CS	(2010,0080)
Soft Tissue Thermal Index	DS	(0018,5027)
Soft Tissue-focus Thermal Index	DS	(0018,5028)
Soft Tissue-surface Thermal Index	DS	(0018,5029)
Softcopy VOI LUT Sequence	SQ	(0028,3110)
Software Version(s)	LO	(0018,1020)
SOP Authorization Comment	LT	(0100,0424)
SOP Authorization Date and Time	DT	(0100,0420)
SOP Class UID	UI	(0008,0016)
SOP Classes Supported	UI	(0008,115A)
SOP Instance Status	CS	(0100,0410)
SOP Instance UID	UI	(0008,0018)
Source Application Entity Title	AE	(0002,0016)
Source Applicator ID	SH	(300A,0291)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Source Applicator Length	DS	(300A,0296)
Source Applicator Manufacturer	LO	(300A,0298)
Source Applicator Name	LO	(300A,0294)
Source Applicator Number	IS	(300A,0290)
Source Applicator Step Size	DS	(300A,02A0)
Source Applicator Type	CS	(300A,0292)
Source Applicator Wall Nominal Thickness	DS	(300A,029C)
Source Applicator Wall Nominal Transmission	DS	(300A,029E)
Source Encapsulation Nominal Thickness	DS	(300A,0222)
Source Encapsulation Nominal Transmission	DS	(300A,0224)
Source Image Sequence	SQ	(0008,2112)
Source Isotope Half Life	DS	(300A,0228)
Source Isotope Name	LO	(300A,0226)
Source Manufacturer	LO	(300A,0216)
Source Movement Type	CS	(300A,0288)
Source Number	IS	(300A,0212)
Source Sequence	SQ	(300A,0210)
Source Serial Number	LO	(3008,0105)
Source to Beam Limiting Device Distance	DS	(300A,00BA)
Source to Block Tray Distance	DS	(300A,00F6)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Source to Compensator Tray Distance	DS	(300A,00E6)
Source to Reference Object Distance	DS	(3002,0028)
Source to Surface Distance	DS	(300A,0130)
Source to Wedge Tray Distance	DS	(300A,00DA)
Source Type	CS	(300A,0214)
Source Waveform Sequence	SQ	(003A,020A)
Source-Axis Distance	DS	(300A,00B4)
Spacing Between Slices	DS	(0018,0088)
Spatial Resolution	DS	(0018,1050)
Special Needs	LO	(0038,0050)
Specific Character Set	CS	(0008,0005)
Specific Character Set of File-set Descriptor File	CS	(0004,1142)
Specified Channel Total Time	DS	(3008,0132)
Specified Meterset	DS	(3008,0042)
Specified Number of Pulses	IS	(3008,0136)
Specified Primary Meterset	DS	(3008,0032)
Specified Pulse Repetition Interval	DS	(3008,013A)
Specified Secondary Meterset	DS	(3008,0033)
Specified Treatment Time	DS	(3008,003A)
Specimen Accession Number	LO	(0040,050A)
Specimen Identifier	LO	(0040,0551)
Specimen Sequence	SQ	(0040,0550)
Specimen Type Code Sequence	SQ	(0040,059A)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Stage Code Sequence	SQ	(0040,000A)
Stage Name	SH	(0008,2120)
Stage Number	IS	(0008,2122)
Start Angle	DS	(0054,0200)
Start Cumulative Meterset Weight	DS	(300C,0008)
Start Meterset	DS	(3008,0078)
Start Trim	IS	(0008,2142)
Station Name	SH	(0008,1010)
Steering Angle	FD	(0018,6036)
Stop Trim	IS	(0008,2143)
Storage Media File-set ID	SH	(0088,0130)
Storage Media File-set UID	UI	(0088,0140)
Structure Set Date	DA	(3006,0008)
Structure Set Description	ST	(3006,0006)
Structure Set Label	SH	(3006,0002)
Structure Set Name	LO	(3006,0004)
Structure Set ROI Sequence	SQ	(3006,0020)
Structure Set Time	TM	(3006,0009)
Study Arrival Date	DA	(0032,1040)
Study Arrival Time	TM	(0032,1041)
Study Comments	LT	(0032,4000)
Study Completion Date	DA	(0032,1050)
Study Completion Time	TM	(0032,1051)
Study Component Status ID	CS	(0032,1055)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Study Date	DA	(0008,0020)
Study Description	LO	(0008,1030)
Study ID	SH	(0020,0010)
Study ID Issuer	LO	(0032,0012)
Study Instance UID	UI	(0020,000D)
Study Priority ID	CS	(0032,000C)
Study Read Date	DA	(0032,0034)
Study Read Time	TM	(0032,0035)
Study Status ID	CS	(0032,000A)
Study Time	TM	(0008,0030)
Study Verified Date	DA	(0032,0032)
Study Verified Time	TM	(0032,0033)
Supported Image Display Formats Sequence	SQ	(2000,00A8)
Surface Entry Point	DS	(300A,012E)
Synchronization Channel	US	(0018,106C)
Synchronization Frame of Reference UID	UI	(0020,0200)
Synchronization Trigger	CS	(0018,106A)
Syringe Counts	IS	(0018,1045)
Table Angle	DS	(0018,1138)
Table Height	DS	(0018,1130)
Table Lateral Increment	DS	(0018,1136)
Table Longitudinal Increment	DS	(0018,1137)
Table Motion	CS	(0018,1134)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Table of Parameter Values	FL	(0018,605A)
Table of Pixel Values	UL	(0018,6058)
Table of X Break Points	UL	(0018,6052)
Table of Y Break Points	FD	(0018,6054)
Table Top Eccentric Angle	DS	(300A,0125)
Table Top Eccentric Angle Tolerance	DS	(300A,004E)
Table Top Eccentric Axis Distance	DS	(300A,0124)
Table Top Eccentric Rotation Direction	CS	(300A,0126)
Table Top Lateral Position	DS	(300A,012A)
Table Top Lateral Position Tolerance	DS	(300A,0053)
Table Top Lateral Setup Displacement	DS	(300A,01D6)
Table Top Longitudinal Position	DS	(300A,0129)
Table Top Longitudinal Position Tolerance	DS	(300A,0052)
Table Top Longitudinal Setup Displacement	DS	(300A,01D4)
Table Top Vertical Position	DS	(300A,0128)
Table Top Vertical Position Tolerance	DS	(300A,0051)
Table Top Vertical Setup Displacement	DS	(300A,01D2)
Table Traverse	DS	(0018,1131)
Table Type	CS	(0018,113A)
Table Vertical Increment	DS	(0018,1135)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Target Maximum Dose	DS	(300A,0027)
Target Minimum Dose	DS	(300A,0025)
Target Prescription Dose	DS	(300A,0026)
Target Underdose Volume Fraction	DS	(300A,0028)
Template Extension Creator UID	UI	(0040,DB0D)
Template Extension Flag	CS	(0040,DB0B)
Template Extension Organization UID	UI	(0040,DB0C)
Template Identifier	CS	(0040,DB00)
Template Local Version	DT	(0040,DB07)
Template Name	LO	(300A,0244)
Template Number	IS	(300A,0240)
Template Type	SH	(300A,0242)
Template Version	DT	(0040,DB06)
Temporal Position Identifier	IS	(0020,0100)
Temporal Range Type	CS	(0040,A130)
Temporal Resolution	DS	(0020,0110)
Text Object Sequence	SQ	(0070,0008)
Text String	LO	(2030,0020)
Text Value	UT	(0040,A160)
Therapy Description	CS	(0018,0039)
Therapy Type	CS	(0018,0037)
Thermal Index	DS	(0018,5024)
TID Offset	SS	(0028,6120)
Time	TM	(0040,A122)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Time Distribution Protocol	CS	(0018,1802)
Time of Last Calibration	TM	(0018,1201)
Time of Last Detector Calibration	TM	(0018,700E)
Time of Secondary Capture	TM	(0018,1014)
Time Slice Vector	US	(0054,0100)
Time Slot Information Sequence	SQ	(0054,0072)
Time Slot Time	DS	(0054,0073)
Time Slot Vector	US	(0054,0070)
Time Source	SH	(0018,1801)
Timezone Offset From UTC	SH	(0008,0201)
TM-Line Position X ₀	UL	(0018,603C)
TM-Line Position X ₁	UL	(0018,6040)
TM-Line Position Y ₀	UL	(0018,603E)
TM-Line Position Y ₁	UL	(0018,6042)
Tolerance Table Label	SH	(300A,0043)
Tolerance Table Number	IS	(300A,0042)
Tolerance Table Sequence	SQ	(300A,0040)
Tomo Angle	DS	(0018,1470)
Tomo Class	CS	(0018,1491)
Tomo Layer Height	DS	(0018,1460)
Tomo Time	DS	(0018,1480)
Tomo Type	CS	(0018,1490)
Topic Author	LO	(0088,0910)
Topic Key Words	LO	(0088,0912)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Topic Subject	ST	(0088,0906)
Topic Title	LO	(0088,0904)
Total Block Tray Factor	DS	(300A,00F2)
Total Compensator Tray Factor	DS	(300A,00E2)
Total Number of Exposures	US	(0040,0301)
Total Reference Air Kerma	DS	(300A,0250)
Total Time	UL	(50xx,200A)
Total Time of Fluoroscopy	US	(0040,0300)
Transaction UID	UI	(0008,1195)
Transducer Data	LO	(0018,5010)
Transducer Frequency	UL	(0018,6030)
Transducer Orientation Modifier Sequence	SQ	(0008,2246)
Transducer Orientation Sequence	SQ	(0008,2244)
Transducer Position Modifier Sequence	SQ	(0008,2242)
Transducer Position Sequence	SQ	(0008,2240)
Transducer Type	CS	(0018,6031)
Transfer Syntax UID	UI	(0002,0010)
Transfer Tube Length	DS	(300A,02A4)
Transfer Tube Number	IS	(300A,02A2)
Transmitting Coil	SH	(0018,1251)
Transverse Mash	IS	(0054,1202)
Treatment Control Point Date	DA	(3008,0024)
Treatment Control Point Time	TM	(3008,0025)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Treatment Date	DA	(3008,0250)
Treatment Delivery Type	CS	(300A,00CE)
Treatment Intent	CS	(300A,000A)
Treatment Machine Name	SH	(300A,00B2)
Treatment Machine Sequence	SQ	(300A,0206)
Treatment Protocols	LO	(300A,0009)
Treatment Session Application Setup Sequence	SQ	(3008,0110)
Treatment Session Beam Sequence	SQ	(3008,0020)
Treatment Sites	LO	(300A,000B)
Treatment Status Comment	ST	(3008,0202)
Treatment Summary Calculated Dose Reference Sequence	SQ	(3008,0050)
Treatment Summary Measured Dose Reference Sequence	SQ	(3008,00E0)
Treatment Termination Code	SH	(3008,002B)
Treatment Termination Status	CS	(3008,002A)
Treatment Time	TM	(3008,0251)
Treatment Verification Status	CS	(3008,002C)
Trigger Sample Position	UL	(0018,106E)
Trigger Source or Type	LO	(0018,1061)
Trigger Time	DS	(0018,1060)
Trigger Time Offset	DS	(0018,1069)
Trigger Vector	IS	(0054,0210)
Trigger Window	IS	(0018,1094)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Trim	CS	(2010,0140)
Type of Data	CS	(50xx,0020)
Type of Detector Motion	CS	(0054,0202)
Type of Filters	LO	(0018,1161)
UID	UI	(0040,A124)
Ultrasound Color Data Present	US	(0028,0014)
Unformatted Text Value	ST	(0070,0006)
Units	CS	(0054,1001)
Value Type	CS	(0040,A040)
Variable Flip Angle Flag	CS	(0018,1315)
Verification DateTime	DT	(0040,A030)
Verification Flag	CS	(0040,A493)
Verifying Observer Identification Code Sequence	SQ	(0040,A088)
Verifying Observer Name	PN	(0040,A075)
Verifying Observer Sequence	SQ	(0040,A073)
Verifying Organization	LO	(0040,A027)
Vertices of the Polygonal Collimator	IS	(0018,1720)
Vertices of the Polygonal Shutter	IS	(0018,1620)
Video Image Format Acquired	SH	(0018,1022)
View Code Sequence	SQ	(0054,0220)
View Modifier Code Sequence	SQ	(0054,0222)
View Name	SH	(0008,2127)
View Number	IS	(0008,2128)
View Position	CS	(0018,5101)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
Visit Comments	LT	(0038,4000)
Visit Status ID	CS	(0038,0008)
VOI LUT Sequence	SQ	(0028,3010)
Waveform Bits Allocated	US	(5400,1004)
Waveform Bits Stored	US	(003A,021A)
Waveform Channel Number	IS	(003A,0202)
Waveform Data	OB or OW	(5400,1010)
Waveform Originality	CS	(003A,0004)
Waveform Padding Value	OB or OW	(5400,100A)
Waveform Sample Interpretation	CS	(5400,1006)
Waveform Sequence	SQ	(5400,0100)
Wedge Angle	IS	(300A,00D5)
Wedge Factor	DS	(300A,00D6)
Wedge ID	SH	(300A,00D4)
Wedge Number	IS	(300A,00D2)
Wedge Orientation	DS	(300A,00D8)
Wedge Position	CS	(300A,0118)
Wedge Position Sequence	SQ	(300A,0116)
Wedge Sequence	SQ	(300A,00D1)
Wedge Type	CS	(300A,00D3)
Whole Body Technique	CS	(0018,1301)
Window Center	DS	(0028,1050)
Window Center & Width Explanation	LO	(0028,1055)
Window Width	DS	(0028,1051)

Table A-1: DICOM Attributes (Continued)

Attribute Name	VR	Tag
X Focus Center	DS	(0018,1183)
X offset in Slide Coordinate System	DS	(0040,072A)
X-Ray Image Receptor Angle	DS	(3002,000E)
X-Ray Image Receptor Translation	DS	(3002,000D)
X-Ray Output	DS	(0040,0312)
X-ray Tube Current	IS	(0018,1151)
X-Ray Tube Current in μ A	DS	(0018,8151)
Y Focus Center	DS	(0018,1184)
Y offset in Slide Coordinate System	DS	(0040,073A)
Z offset in Slide Coordinate System	DS	(0040,074A)
Zoom Center	DS	(0028,0032)
Zoom Factor	DS	(0028,0031)

Table A-1: DICOM Attributes (Continued)

Value Representations

The following Value Representations describe the data type and format of each DICOM attribute. These are defined in Section 6.2 of *Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding*.

VR	Definition	Details
AE	<p>Application Entity</p> <p>A string of characters with leading and trailing spaces (20H) being non-significant. The value made of 16 spaces, meaning “no application name specified”, shall not be used.</p>	<p>Character Repertoire</p> <p>Default Character Repertoire excluding control characters LF, FF, CR and ESC.</p> <p>Length</p> <p>16 bytes maximum</p> <p>IDL Data Type</p> <p>STRING</p>
AS	<p>Age String</p> <p>A string of characters with one of the following formats -- nnnD, nnnW, nnnM, nnnY; where nnn shall contain the number of days for D, weeks for W, months for M, or years for Y.</p> <p>Example -</p> <p>“018M” would represent an age of 18 months.</p>	<p>Character Repertoire</p> <p>“0”-“9”, “D”, “W”, “M”, “Y” of Default Character Repertoire</p> <p>Length</p> <p>4 bytes fixed</p> <p>IDL Data Type</p> <p>STRING</p>

Table A-2: DICOM Value Representations (VR) Types

VR	Definition	Details
AT	<p>Attribute Tag Ordered pair of 16-bit unsigned integers that is the value of a Data Element Tag.</p> <p>Example - A Data Element Tag of (0018,00FF) would be encoded as a series of 4 bytes in a Little-Endian Transfer Syntax as 18H,00H,FFH,00H and in a Big-Endian Transfer Syntax as 00H,18H,00H,FFH.</p> <p>Note - The encoding of an AT value is exactly the same as the encoding of a Data Element Tag as defined in Section 7.</p>	<p>Character Repertoire Not applicable</p> <p>Length 4 bytes fixed</p> <p>IDL Data Type ULONG</p>
CS	<p>Code String A string of characters with leading or trailing spaces (20H) being non-significant.</p>	<p>Character Repertoire Uppercase characters, “0”-”9”, the SPACE character, and underscore “_”, of the Default Character Repertoire</p> <p>Length 16 byte maximum</p> <p>IDL Data Type STRING</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
DA	<p>Date</p> <p>A string of characters of the format <code>yyyymmdd</code>; where <code>yyyy</code> shall contain year, <code>mm</code> shall contain the month, and <code>dd</code> shall contain the day. This conforms to the ANSI HISPP MSDS Date common data type.</p> <p>Example -</p> <p>“19930822” would represent August 22, 1993.</p> <p>Note -</p> <ol style="list-style-type: none"> For reasons of backward compatibility with versions of this standard prior to V3.0, it is recommended that implementations also support a string of characters of the format <code>yyyy.mm.dd</code> for this VR. See also DT VR in this table. <p>Note - For reasons specified in the previous column, implementations may wish to support the “.” character as well.</p>	<p>Character Repertoire</p> <p>“0”-“9” of Default Character Repertoire</p> <p>Length</p> <p>8 bytes fixed</p> <p>Note - For reasons specified in the previous columns, implementations may also wish to support a 10 byte fixed length as well.</p> <p>IDL Data Type</p> <p>STRING</p>
DL	<p>Delimitation</p>	
DS	<p>Decimal String</p> <p>A string of characters representing either a fixed point number or a floating point number. A fixed point number shall contain only the characters 0-9 with an optional leading “+” or “-” and an optional “.” to mark the decimal point. A floating point number shall be conveyed as defined in ANSI X3.9, with an “E” or “e” to indicate the start of the exponent. Decimal Strings may be padded with leading or trailing spaces. Embedded spaces are not allowed.</p>	<p>Character Repertoire</p> <p>“0” - “9”, “+”, “-”, “E”, “e”, “.” of Default Character Repertoire</p> <p>Length</p> <p>16 bytes maximum</p> <p>IDL Data Type</p> <p>STRING</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
DT	<p>DATE TIME</p> <p>The Date Time common data type. Indicates a concatenated date-time ASCII string in the format: YYYYMMDDHHMMSS.FFFFFFFF&ZZZZ The components of this string, from left to right, are YYYY = Year, MM = Month, DD = Day, HH = Hour, MM = Minute, SS = Second, FFFFFFFF = Fractional Second, & = “+” or “-”, and ZZZZ = Hours and Minutes of offset. &ZZZZ is an optional suffix for plus/minus offset from Coordinated Universal Time. A component that is omitted from the string is termed a null component. Trailing null components of Date Time are ignored. Non-trailing null components are prohibited, given that the optional suffix is not considered as a component.</p> <p>Note - For reasons of backward compatibility with versions of this standard prior to V3.0, many existing DICOM Data Elements use the separate DA and TM VRs. Standard and Private Data Elements defined in the future should use DT, when appropriate, to be more compliant with ANSI HISPP MSDS.</p>	<p>Character Repertoire “0” - “9”, “+”, “-”, “.” of Default Character Repertoire</p> <p>Length 26 bytes maximum</p> <p>IDL Data Type STRING</p>
FL	<p>Floating Point Single</p> <p>Single precision binary floating point number represented in IEEE 754:1985 32-bit Floating Point Number Format.</p>	<p>Character Repertoire Not applicable</p> <p>Length 4 bytes fixed</p> <p>IDL Data Type FLOAT</p>
FD	<p>Floating Point Double</p> <p>Double precision binary floating point number represented in IEEE 754:1985 64-bit Floating Point Number Format.</p>	<p>Character Repertoire Not applicable</p> <p>Length 8 bytes fixed</p> <p>IDL Data Type DOUBLE</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
IS	<p>Integer String</p> <p>A string of characters representing an Integer in base-10 (decimal), shall contain only the characters 0 - 9, with an optional leading “+” or “-”. It may be padded with leading and/or trailing spaces. Embedded spaces are not allowed. The integer, n, represented shall be in the range:</p> $-2^{31} \leq n \leq (2^{31} - 1).$	<p>Character Repertoire “0”-“9”, “+”, “-” of Default Character Repertoire</p> <p>Length 12 bytes maximum</p> <p>IDL Data Type STRING</p>
LO	<p>Long String</p> <p>A character string that may be padded with leading and/or trailing spaces. The character code 5CH (the BACKSLASH “\” in ISO-IR 6) shall not be present, as it is used as the delimiter between values in multiple valued Default Character Repertoire and/or as defined by (0008,0005).</p>	<p>Character Repertoire 64 chars</p> <p>Length 64 chars maximum</p> <p>Note - The length of the VRs for which the Character Repertoire can be extended or replaced are expressly specified in characters rather than bytes because the mapping from a character to the number of bytes used for that character’s encoding may be dependent on the character set used.</p> <p>IDL Data Type STRING</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
LT	<p>Long Text</p> <p>A character string that may contain one or more paragraphs. It may contain the Graphic Character set and the Control Characters, CR, LF, FF, and ESC. It may be padded with trailing spaces, which may be ignored, but leading spaces are considered to be significant. Data Elements with this VR shall not be multi-valued and therefore character code 5CH (the BACKSLASH “\” in ISO-IR 6) may be used. Default Character Repertoire and/or as defined by (0008,0005).</p>	<p>Character Repertoire 10240 chars</p> <p>Length 10240 chars maximum</p> <p>Note - The length of the VRs for which the Character Repertoire can be extended or replaced are expressly specified in characters rather than bytes because the mapping from a character to the number of bytes used for that character’s encoding may be dependent on the character set used.</p> <p>IDL Data Type STRING</p>
OB	<p>Other Byte String</p> <p>A string of bytes where the encoding of the contents is specified by the negotiated Transfer Syntax. OB is a VR which is insensitive to Little/Big Endian byte ordering (see Section 7.3 of <i>Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding</i>). The string of bytes shall be padded with a single trailing NULL byte value (00H) when necessary to achieve even length.</p>	<p>Character Repertoire Not applicable</p> <p>Length See Transfer Syntax definition</p> <p>IDL Data Type BYTE</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
OF	<p>Other Float String</p> <p>A string of 32-bit IEEE 754:1985 floating point words. OF is a VR which requires byte swapping within each 32-bit word when changing between Little Endian and Big Endian byte ordering (see Section 7.3 of <i>Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding</i>).</p>	<p>Character Repertoire Not applicable</p> <p>Length 2³²-4 maximum</p> <p>IDL Data Type FLOAT</p>
OW	<p>Other Word String</p> <p>A string of 16-bit words where the encoding of the contents is specified by the negotiated Transfer Syntax. OW is a VR which requires byte swapping within each word when changing between Little Endian and Big Endian byte ordering (see Section 7.3 of <i>Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding</i>).</p>	<p>Character Repertoire Not applicable</p> <p>Length See Transfer Syntax definition</p> <p>IDL Data Type INT</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
PN	<p>Person Name</p> <p>A character string encoded using a 5 component convention. The character code 5CH (the BACKSLASH “\” in ISO-IR 6) shall not be present, as it is used as the delimiter between values in multiple valued data elements. The string may be padded with trailing spaces. The five components in their order of occurrence are: family name complex, given name complex, middle name, name prefix, name suffix. Any of the five components may be an empty string. The component delimiter shall be the caret “^” character (5EH). Delimiters are required for interior null components. Trailing null components and their delimiters may be omitted. Multiple entries are permitted in each component and are encoded as natural text strings, in the format preferred by the named person. This conforms to the ANSI HISPP MSDS Person Name common data type. This group of five components is referred to as a Person Name component group. For the purpose of writing names in ideographic characters and in phonetic characters, up to 3 groups of components (see Annex H examples 1 and 2) may be used. The delimiter for component groups shall be the equals character “=” (3DH). The three component groups of components in their order of occurrence are: a single-byte character representation, an ideographic representation, and a phonetic representation. Any component group may be absent, including the first component group. In this case, the person name may start with one or more “=” delimiters. Delimiters are required for interior null component groups. Trailing null component groups and their delimiters may be omitted. Precise semantics are defined for each component group. See section 6.2.1 of <i>Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding</i>.</p>	<p>Character Repertoire</p> <p>Default Character Repertoire and/or as defined by (0008,0005) excluding Control Characters LF, FF, and CR but allowing Control Character ESC.</p> <p>Length</p> <p>64 chars maximum per component group</p> <p>Note - The length of VRs for which the Character Repertoire can be extended or replaced are expressly specified in characters rather than bytes because the mapping from a character to the number of bytes used for that character’s encoding may be dependent on the character set used.</p> <p>IDL Data Type</p> <p>STRING</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
PN	<p>Person Name (continued)</p> <p>Examples:</p> <p>Rev. John Robert Quincy Adams, B.A. M.Div. “Adams^John Robert Quincy^^Rev.^B.A. M.Div.” [One family name; three given names; no middle name; one prefix; two suffixes.]</p> <p>Susan Morrison-Jones, Ph.D., Chief Executive Officer “Morrison-Jones^Susan^^Ph.D., Chief Executive Officer” [Two family names; one given name; no middle name; no prefix; two suffixes.]</p> <p>John Doe “Doe^John” [One family name; one given name; no middle name, prefix, or suffix. Delimiters have been omitted for the three trailing null components.] (for examples of the encoding of Person Names using multi-byte character sets see Annex H of <i>Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding</i>)</p> <p>Note -</p> <ol style="list-style-type: none"> 1. This five component convention is also used by HL7 as defined in ASTM E-1238-91 and further specialized by the ANSI MSDS. 2. In typical American and European usage the first occurrence of “given name” would represent the “first name”. The second and subsequent occurrences of the “given name” would typically be treated as a middle name(s). The “middle name” component is retained for the purpose of backward compatibility with existing standards. 3. The “Degree” component present in ASTM E-1238-91 is absorbed into the “Suffix” component. 	<p>Character Repertoire</p> <p>Default Character Repertoire and/or as defined by (0008,0005) excluding Control Characters LF, FF, and CR but allowing Control Character ESC.</p> <p>Length</p> <p>64 chars maximum per component group</p> <p>Note - The length of VRs for which the Character Repertoire can be extended or replaced are expressly specified in characters rather than bytes because the mapping from a character to the number of bytes used for that character’s encoding may be dependent on the character set used.</p> <p>IDL Data Type</p> <p>STRING</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
SH	<p>Short String</p> <p>A character string that may be padded with leading and/or trailing spaces. The character code 05CH (the BACKSLASH “\” in ISO-IR 6) shall not be present, as it is used as the delimiter between values for multiple data elements. The string shall not have Control Characters except ESC. Default Character Repertoire and/or as defined by (0008,0005).</p>	<p>Character Repertoire 16 chars</p> <p>Length 16 chars maximum</p> <p>Note - The length of VRs for which the Character Repertoire can be extended or replaced are expressly specified in characters rather than bytes because the mapping from a character to the number of bytes used for that character’s encoding may be dependent on the character set used.</p> <p>IDL Data Type STRING</p>
SL	<p>Signed Long</p> <p>Signed binary integer 32 bits long in 2’s complement form. Represents an integer, n, in the range:</p> $- 2^{31} \leq n \leq (2^{31} - 1)$	<p>Character Repertoire Not applicable</p> <p>Length 4 bytes fixed</p> <p>IDL Data Type LONG</p>
SQ	<p>Sequence of Items</p> <p>Value is a Sequence of zero or more Items, as defined in Section 7.5 of <i>Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding</i>.</p>	<p>Character Repertoire Not applicable</p> <p>Length Not applicable</p> <p>IDL Data Type LONG</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
SS	Signed Short Signed binary integer 16 bits long in 2's complement form. Represents an integer n in the range: $-2^{15} \leq n \leq (2^{15} - 1)$	Character Repertoire Not applicable Length 2 bytes fixed IDL Data Type INT
ST	Short Text A character string that may contain one or more paragraphs. It may contain the Graphic Character set and the Control Characters, CR, LF, FF, and ESC. It may be padded with trailing spaces, which may be ignored, but leading spaces are considered to be significant. Data Elements with this VR shall not be multi-valued and therefore character code 5CH (the BACKSLASH “\” in ISO-IR 6) may be used. Default Character Repertoire and/or as defined by (0008,0005).	Character Repertoire 1024 chars Length 1024 chars maximum Note - The length of Value Representations for which the Character Repertoire can be extended or replaced are expressly specified in characters rather than bytes because the mapping from a character to the number of bytes used for that character's encoding may be dependent on the character set used. IDL Data Type STRING

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
TM	<p>Time</p> <p>A string of characters of the format <i>hhmmss.frac</i>; where <i>hh</i> contains hours (range “00” - “23”), <i>mm</i> contains minutes (range “00” - “59”), <i>ss</i> contains seconds (range “00” - “59”), and <i>frac</i> contains a fractional part of a second as small as 1 millionth of a second (range “000000” - “999999”). A 24 hour clock is assumed. Midnight can be represented by only “0000” since “2400” would violate the hour range. The string may be padded with trailing spaces. Leading and embedded spaces are not allowed. One or more of the components <i>mm</i>, <i>ss</i>, or <i>frac</i> may be unspecified as long as every component to the right of an unspecified component is also unspecified. If <i>frac</i> is unspecified the preceding “.” may not be included. <i>Frac</i> shall be held to six decimal places or less to ensure its format conforms to the ANSI HISPP MSDS Time common data type.</p> <p>Examples -</p> <ol style="list-style-type: none"> “070907.0705” represents a time of 7 hours, 9 minutes and 7.0705 seconds. “1010” represents a time of 10 hours, and 10 minutes. “021” is an invalid value. <p>Note -</p> <ol style="list-style-type: none"> For reasons of backward compatibility with versions of this standard prior to V3.0, it is recommended that implementations also support a string of characters of the format <i>hh:mm:ss.frac</i> for this VR. See also DT VR in this table. 	<p>Character Repertoire</p> <p>“0” - “9”, “.” of Default Character Repertoire</p> <p>Length</p> <p>16 bytes maximum</p> <p>IDL Data Type</p> <p>STRING</p>

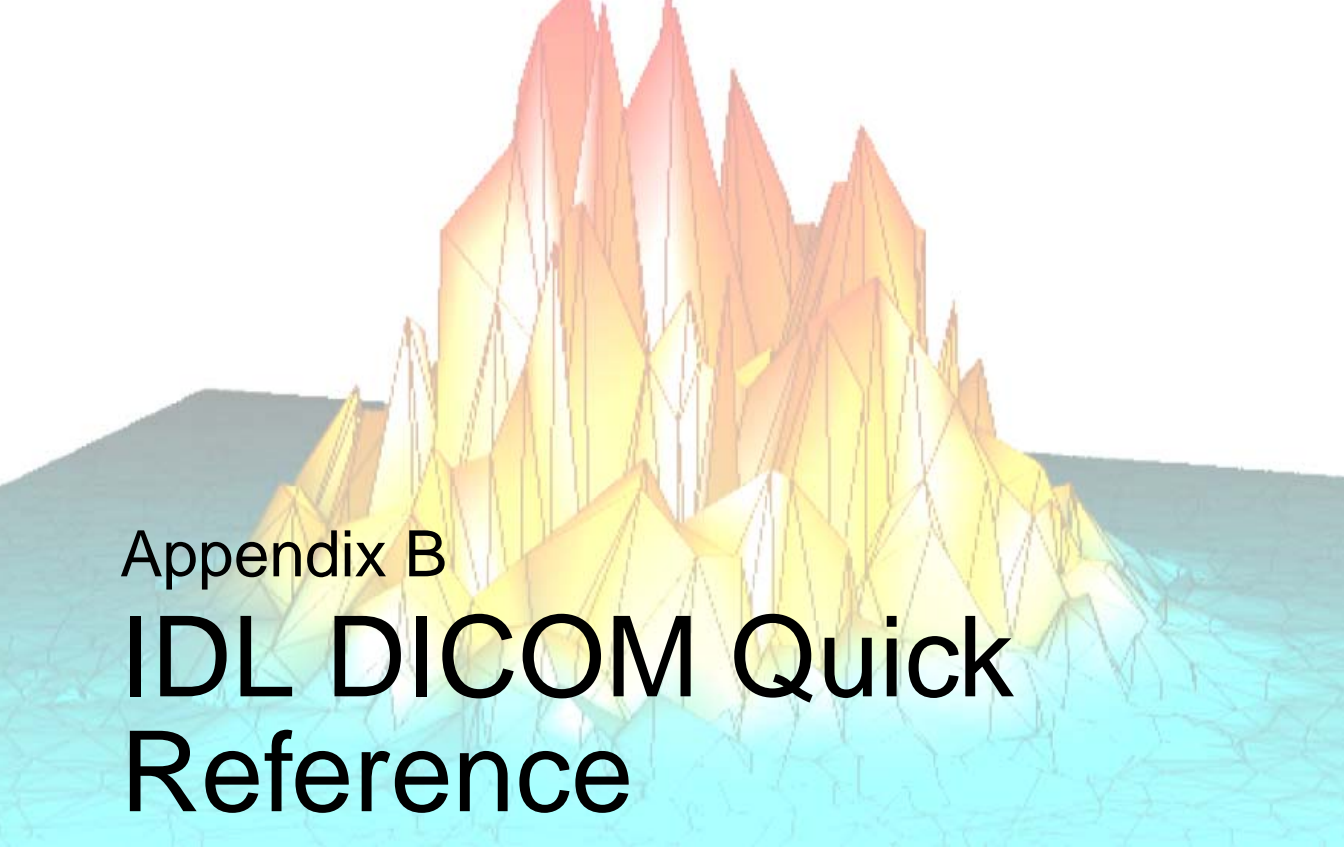
Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
UI	<p>Unique Identifier</p> <p>A character string containing a UID that is used to uniquely identify a wide variety of items. The UID is a series of numeric components separated by the period “.” character. If a Value Field containing one or more UIDs is an odd number of bytes in length, the Value Field shall be padded with a single trailing NULL (00H) character to ensure that the Value Field is an even number of bytes in length. See Section 9 and Annex B of <i>Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding</i> for a complete specification and examples.</p>	<p>Character Repertoire “0” - “9”, “.” of Default Character Repertoire</p> <p>Length 64 bytes maximum</p> <p>IDL Data Type STRING</p>
UL	<p>Unsigned Long</p> <p>Unsigned binary integer 32 bits long. Represents an integer n in the range:</p> $0 \leq n < 2^{32}$	<p>Character Repertoire Not applicable</p> <p>Length 4 bytes fixed</p> <p>IDL Data Type ULONG</p>
UN	<p>Unknown</p> <p>A string of bytes where the encoding of the contents is unknown (see Section 6.2.2 of <i>Digital Imaging and Communications in Medicine (DICOM) - Part 5: Data Structures and Encoding</i>).</p>	<p>Character Repertoire Not applicable</p> <p>Length Any length valid for any of the other DICOM VRs</p> <p>IDL Data Type BYTE</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)

VR	Definition	Details
US	<p>Unsigned Short</p> <p>Unsigned binary integer 16 bits long. Represents integer n in the range:</p> $0 \leq n < 2^{16}$	<p>Character Repertoire Not applicable</p> <p>Length 2 bytes fixed</p> <p>IDL Data Type UINT</p>
UT	<p>Unlimited Text</p> <p>A character string that may contain one or more paragraphs. It may contain the Graphic Character set and the Control Characters, CR, LF, FF, and ESC. It may be padded with trailing spaces, which may be ignored, but leading spaces are considered to be significant. Data Elements with this VR shall not be multi-valued and therefore character code 5CH (the BACKSLASH “\” in ISO-IR 6) may be used.</p> <p>The text will be interpreted as specified by Specific Character Set (0008,0005).</p>	<p>Character Repertoire Default Character Repertoire and/or as defined by (0008,0005).</p> <p>Length $2^{32} - 2$</p> <p>Note - Limited only by the size of the maximum unsigned integer representable in a 32 bit VL field minus one, since FFFFFFFFH is reserved.</p> <p>IDL Data Type STRING</p>

Table A-2: DICOM Value Representations (VR) Types (Continued)



Appendix B

IDL DICOM Quick Reference

Alphabetical Listing

DICOMEX_GETCONFIGFILEPATH - Returns the location of the local or system configuration file associated with the DICOM Network Services utility.

Result =
DICOMEX_GETCONFIGFILEPATH([, /SYSTEM])

DICOMEX_GETSTORSCPDIR - Returns the location of the directory associated with the Storage SCP Service of the DICOM Network Services utility.

Result = DICOMEX_GETSTORSCPDIR()

DICOMEX_NET - Launches the DICOM Network Service utility.

DICOMEX_NET [, /SYSTEM]

IDLffDicomEx object - This object allows you to read and write DICOM files. It includes the ability to create a new DICOM file, clone and modify an existing DICOM file, or access elements from a file in read-only mode depending on how the IDLffDicomEx object is created. No superclasses. No subclasses.

Properties:

[, BITS_ALLOCATED{Get, Set}=integer]
 [, BITS_STORED{Get, Set}=integer]
 [, COLUMNS{Get, Set}=integer]
 [, FILENAME{Get}=string]
 [, HIGH_BIT{Get, Set}=integer]
 [, IMAGE_TYPE{Get, Set}=(ORIGINAL | DERIVED)
 { PRIMARY | SECONDARY }{modality specific
 characteristics}{implementation specific characteristics}]
 [, INSTANCE_NUMBER{Get, Set}=string]
 [, MODALITY{Get, Set}=string]
 [, NO_PIXEL_DATA{Get, Init}=integer]
 [, NUMBER_OF_FRAMES{Get, Set}=string]
 [, PHOTOMETRIC_INTERPRETATION{Get,
 Set}={MONOCHROME1 | MONOCHROME2 |
 PALETTECOLOR | RGB | HSV | CMYK}]
 [, PIXEL_ASPECT_RATIO{Get, Set}=string]
 [, PIXEL_MAX{Get, Set}=integer]
 [, PIXEL_MIN{Get, Set}=integer]
 [, PIXEL_REPRESENTATION{Get, Set}={0 | 1}]
 [, PIXEL_SPACING{Get, Set}=string]
 [, PLANAR_CONFIGURATION{Get, Set}={0 | 1}]
 [, ROWS{Get, Set}=integer]
 [, SAMPLES_PER_PIXEL{Get, Set}={1 | 3 | 4}]
 [, SOP_CLASS_UID{Get, Set}=string]
 [, SOP_INSTANCE_UID{Get, Set}=string]
 [, TRANSFER_SYNTAX{Get}=string]

IDLffDicomEx::AddGroup - Creates a group within a sequence.

Result = *Obj*->[IDLffDicomEx::]AddGroup (*DicomTag*
 [, PARENTSEQID=integer])

IDLffDicomEx::AddPrivateGroup - Creates a group within a private sequence.

Result = *Obj*->[IDLffDicomEx::]AddPrivateGroup
 (*PrivateCode*, *Group*, *Element*
 [, PARENTSEQID=integer])

IDLffDicomEx::AddPrivateSequence - Creates a new private sequence.

Result = *Obj*->[IDLffDicomEx::]AddPrivateSequence
 (*PrivateCode*, *Group*, *Element*
 [, PARENTSEQID=integer])

IDLffDicomEx::AddSequence - Creates a new sequence.

Result = *Obj*->[IDLffDicomEx::]AddSequence (*DicomTag*
 [, PARENTSEQID=integer])

IDLffDicomEx::ChangeTransferSyntax - Changes the transfer syntax of the IDLffDicomEx object and its associated pixel data. This allows you to change the compression setting of the pixel data and ensures that the transfer syntax value and pixel data are synchronized.

Obj->[IDLffDicomEx::]ChangeTransferSyntax,
NewSyntaxUID [, /LOSSY]

IDLffDicomEx::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, *Obj*
 or
Obj->[IDLffDicomEx::]Cleanup (Only in subclass'
 Cleanup method.)

IDLffDicomEx::Commit - Writes changes to the object to the underlying DICOM file.

Obj->[IDLffDicomEx::]Commit

IDLffDicomEx::CopyTags - Copies specified tags from the source object to the destination object beginning with the DICOM attribute tag specified by the start tag and copying up to the stop tag.

Obj->[IDLffDicomEx::]CopyTags, *DestinationObject*,
Start_Tag, *Stop_Tag*

IDLffDicomEx::EnumerateTags - Returns an array of structures representing the contents of the DICOM file.

Result = *Obj*->[IDLffDicomEx::]EnumerateTags
 ([, START_TAG=string] [, STOP_TAG=string]
 [, COUNT=variable] [, FILENAME=string] [, /QUIET])

IDLffDicomEx::GetDescription - Returns the description associated with a standard, public DICOM attribute.

Result = *Obj*->[IDLffDICOM::]GetDescription(
DicomTag)

IDLffDicomEx::GetPixelData - Returns pixel data from the DICOM image file.

Result = *Obj*->[IDLffDicomEx::]
 GetPixelData([FRAME=integer] [, /ORDER]
 [, COUNT=variable])

IDLffDicomEx::GetPrivateValue - Returns the value of a private DICOM attribute.

Result = *Obj*->[IDLffDicomEx::]GetPrivate Value
 (*PrivateCode*, *Group*, *Element* [, SEQID=integer]
 [, COUNT=variable])

- IDLffDicomEx::GetPrivateValueCount** - Returns the number of values contained in the value field of a private DICOM attribute.
Result = Obj->[IDLffDicomEx::]GetPrivateValueCount (PrivateCode, Group, Element [, SEQID=integer])
- IDLffDicomEx::GetPrivateValueLength** - Returns the length of all values or of a specified value (in bytes) in a private DICOM attribute.
Result = Obj->[IDLffDicomEx::]GetPrivateValueLength (PrivateCode, Group, Element [, SEQID=integer] [, VALUEINDEX=integer])
- IDLffDicomEx::GetPrivateVR** - Returns the value representation (VR) of a private DICOM attribute.
Result = Obj->[IDLffDicomEx::]GetPrivateVR (PrivateCode, Group, Element [, SEQID=integer])
- IDLffDicomEx::GetProperty** - Retrieves the value of an IDLffDicomEx property.
Obj->[IDLffDicomEx::]GetProperty [, PROPERTY=variable]
- IDLffDicomEx::GetValue** - Returns the value of a DICOM attribute specified by a standard DICOM attribute tag.
Result = Obj->[IDLffDicomEx::]GetValue(DicomTag [, SEQID=integer] [, COUNT=variable])
- IDLffDicomEx::GetValueCount** - Returns the number of values in a public DICOM attribute specified by a standard DICOM attribute tag.
Result = Obj->[IDLffDicomEx::]GetValueCount (DicomTag [, SEQID=integer])
- IDLffDicomEx::GetValueLength** - Returns the length of all values or of a specified value (in bytes) in a DICOM attribute specified by a standard DICOM attribute tag.
Result = Obj->[IDLffDicomEx::]GetValueLength (DicomTag [, SEQID=integer] [, VALUEINDEX=integer])
- IDLffDicomEx::GetVR** - Returns the value representation (VR) of a DICOM attribute.
Result = Obj->[IDLffDicomEx::]GetVR(DicomTag [, SEQID=integer])
- IDLffDicomEx::Init** - Initializes a IDLffDicomEx object. This method is called when the object is created via OBJ_NEW. The IDLffDicomEx object allows you to read and write DICOM files or create a new DICOM file based on keyword settings.
Obj = OBJ_NEW('IDLffDicomEx', FileName [, CLONE=string] [, /CREATE] [, SOP_CLASS=string] [, /NON_CONFORMING])
 or
Result = Obj->[IDLffDicomEx::]Init(FileName [, CLONE=string] [, /CREATE] [, SOP_CLASS=string] [, /NON_CONFORMING]) (Only in a subclass' Init method.)
- IDLffDicomEx::QueryPrivateValue** - Checks a DICOM file for the presence of a specified private attribute. This method allows you to verify the presence of a tag prior to calling a method that requires a DICOM attribute to be present in order to succeed.
Result = Obj->[IDLffDicomEx::]QueryPrivateValue (PrivateCode, Group, Element [, SEQID=integer])
- IDLffDicomEx::QueryValue** - Checks a DICOM file for the presence of a specified attribute. This method allows you to verify the presence of a tag prior to calling a method that requires a DICOM attribute to be present in order to succeed.
Result = Obj->[IDLffDicomEx::]QueryValue(DicomTag [, SEQID=integer])
- IDLffDicomEx::SetPixelData** - Writes pixel data to the DICOM image file.
Obj->[IDLffDicomEx::]SetPixelData, PixelData [, FRAME=integer] [, /ORDER] [, BITS_ALLOCATED=integer] [, COLUMNS=integer] [, NUMBER_OF_FRAMES=string] [, PHOTOMETRIC_INTERPRETATION={MONOCHROME1 | MONOCHROME2 | PALETTECOLOR | RGB | HSV | CMYK}] [, PIXEL_REPRESENTATION={0 | 1}] [, PLANAR_CONFIGURATION={0 | 1}] [, ROWS=integer] [, SAMPLES_PER_PIXEL={1 | 3 | 4}]
- IDLffDicomEx::SetPrivateValue** - Allows you to add and alter private attributes including items contained in private sequences.
Obj->[IDLffDicomEx::]SetPrivateValue, PrivateCode, Group, Element, VR [, Value] [, SEQID=integer] [, /CLEAR] [, /REMOVE] [, /BLOCKREMOVE]
- IDLffDicomEx::SetProperty** - Specifies a property value.
Obj->[IDLffDicomEx::]SetProperty [, PROPERTY=value]
- IDLffDicomEx::SetValue** - Allows you to add and alter attributes including items contained in sequences.
Obj->[IDLffDicomEx::]SetValue, DicomTag [, VR] [, Value] [, SEQID=integer] [, /CLEAR] [, /REMOVE]
- IDLffDicomExCfg object** - This object allows you to set and retrieve the values of IDL DICOM Network Service configuration parameters. No superclasses. No subclasses. No properties.
- IDLffDicomExCfg::Cleanup** - Performs all cleanup on the object, including closing the associated configuration file.
 OBJ_DESTROY, Obj
 or
Obj->[IDLffDicomExCfg::]Cleanup()
(In a lifecycle method only.)
- IDLffDicomExCfg::Commit** - Commits changes made to the object and saves them to the underlying configuration file. Changes are not applied to the current configuration or saved in the configuration file until this method is called.
Obj->[IDLffDicomExCfg::]Commit

IDLffDicomExCfg::Echo - Lets you test the network connection to a remote machine that supports Query SCP or Storage SCP service types.

Result =

Obj->[IDLffDicomExCfg::]Echo(*ApplicationEntityName* [, COUNT=*variable*])

IDLffDicomExCfg::GetApplicationEntities - Returns an array of structures containing the definitions of all the application entities defined or all the application entities of a specified service type.

Result =

Obj->[IDLffDicomExCfg::]GetApplicationEntities([, COUNT=*variable*] [, /SERVICE_TYPE=*string*])

IDLffDicomExCfg::GetApplicationEntity - Returns a structure containing the parameters for a specific Application Entity.

Result =

Obj->[IDLffDicomExCfg::]GetApplicationEntity(*ApplicatonEntityName*)

IDLffDicomExCfg::GetServiceLists - Returns an array of strings containing the names of the service lists that can be assigned to an Application Entity.

Result = Obj->[IDLffDicomExCfg::]GetServiceLists([, COUNT=*variable*])

IDLffDicomExCfg::GetServiceTypes - Returns an array of strings containing the list of service types that can be assigned to an Application Entity.

Result = Obj->[IDLffDicomExCfg::]GetServiceTypes([, COUNT=*variable*])

IDLffDicomExCfg::GetValue - Returns the value for the specified parameter.

Result = Obj->[IDLffDicomExCfg::]GetValue(*ConfigurationParameterName*)

IDLffDicomExCfg::Init - Initializes the IDLffDicomExCfg object. The IDLffDicomExCfg object allows you to set and retrieve the values of IDL DICOM Network Service configuration parameters.

Obj = OBJ_NEW('IDLffDicomExCfg' [, /SYSTEM])

or

Result = Obj->[IDLffDicomExCfg::]Init([, /SYSTEM])
(*In a lifecycle method only.*)

IDLffDicomExCfg::IsDirty - Is used to determine if configuration values have been changed since the configuration file was read into the configuration object.

Result = Obj->[IDLffDicomExCfg::]IsDirty()

IDLffDicomExCfg::RemoveApplicationEntity - Marks a specified Application Entity for removal from the collection of application entities.

Obj->[IDLffDicomExCfg::]RemoveApplicationEntity(*ApplicationEntityName*)

IDLffDicomExCfg::SetApplicationEntity - Defines a new Application Entity or modifies the definition of an existing Application Entity.

Obj->[IDLffDicomExCfg::]SetApplicationEntity(*ApplicationEntityName*, *AET*, *Hostname*, *Port*, *ServiceListName*, *ServiceType*)

IDLffDicomExCfg::SetValue - Sets the value for the specified parameter.

Obj->[IDLffDicomExCfg::]SetValue(*ConfigurationParameterName*, *Value*)

IDLffDicomExCfg::StorageScpService - Used to start, stop or check the status of the local IDL Storage SCP service.

Result = Obj->[IDLffDicomExCfg::]StorageScpService(*Command*)

IDLffDicomExQuery object - This object allows you to query the contents of a remote DICOM node that has been configured to work with IDL's DICOM Network Service feature, and to retrieve DICOM files available on that node. No superclasses. No subclasses.

Properties:

[, ACCESSION_NUMBER{Get, Set}=*string*]
[, CALLBACK_DATA{Get, Set}=*any type*]
[, CALLBACK_FUNCTION{Get, Set}=*string*]
[, FAMILY_NAME{Get, Set}=*string*]
[, GIVEN_NAME{Get, Set}=*string*]
[, INSTANCE_NUMBER{Get, Set}=*string*]
[, MIDDLE_NAME{Get, Set}=*string*]
[, MODALITY{Get, Set}=*string*]
[, PATIENT_ID{Get, Set}=*string*]
[, PATIENT_NAME{Get, Set}=*string*]
[, PREFIX{Get, Set}=*string*]
[, QUERY_LEVEL{Get, Set}=*integer*]
[, QUERY_MODEL{Get, Set}=*integer*]
[, QUERY_SCP{Get, Set}=*string*]
[, SERIES_INSTANCE_UID{Get, Set}=*string*]
[, SERIES_NUMBER{Get, Set}=*string*]
[, SOP_INSTANCE_UID{Get, Set}=*string*]
[, STORAGE_SCP{Get, Set}=*string*]
[, STUDY_DATE{Get, Set}=*string*]
[, STUDY_ID{Get, Set}=*string*]
[, STUDY_INSTANCE_UID{Get, Set}=*string*]
[, STUDY_TIME{Get, Set}=*string*]
[, SUFFIX{Get, Set}=*string*]

IDLffDicomExQuery::Cleanup - Performs all cleanup on the object.

OBJ_DESTROY, *Obj*

or

Obj->[IDLffDicomExQuery::]Cleanup()

(*In a lifecycle method only.*)

IDLffDicomExQuery::ClearProperties - Used to reset the values of all properties of the IDLffDicomExQuery object to their default values.

Obj->[IDLffDicomExQuery::]ClearProperties

IDLffDicomExQuery::GetProperty - Retrieves the value of an IDLffDicomExQuery property.

Obj->[IDLffDicomExQuery::]GetProperty([, *PROPERTY=variable*])

IDLffDicomExQuery::Init - Initializes the IDLffDicomExQuery object. The IDLffDicomExQuery object allows you query or retrieve DICOM files from a remote node configured as a Query SCP available in IDL's DICOM Network Service configuration.

```
Obj = OBJ_NEW('IDLffDicomExQuery')
or
Result = Obj->[IDLffDicomExQuery::]Init( )
(In a lifecycle method only.)
```

IDLffDicomExQuery::Query - Used to query a remote Query/Retrieve SCP Application Entity.

```
Result = Obj->[IDLffDicomExQuery::]Query(
[, COUNT=variable] )
```

IDLffDicomExQuery::QueryModelsSupported - Used to determine what Query/Retrieve services are supported by Query/Retrieve SCP Application Entity defined by the QUERY_SCP property.

```
Result =
Obj->[IDLffDicomExQuery::]QueryModelsSupported(
[, COUNT=variable] )
```

IDLffDicomExQuery::Retrieve - Moves a DICOM file or files from a source node to a destination node.

```
Result = Obj->[IDLffDicomExQuery::]Retrieve(
[, COUNT=variable] [, PATIENT_ID=string]
[, STUDY_INSTANCE_UID=string]
[, SERIES_INSTANCE_UID=string]
[, SOP_INSTANCE_UID=string] )
```

IDLffDicomExQuery::SetProperty - Sets the value of a property or group of properties for the IDLffDicomExQuery object.

```
Obj->[IDLffDicomExQuery::]SetProperty
[, PROPERTY=value]
```

IDLffDicomExStorScu object - This object implements a local DICOM Storage SCU (Service Class User) that allows you to transmit files to a remote destination that is identified as a DICOM Storage SCP (Service Class Provider). No superclasses. No subclasses.

Properties:

```
[, CALLBACK_DATA{Get, Set}=any type]
[, CALLBACK_FUNCTION{Get, Set}=string]
[, STORAGE_SCP{Get, Set}=string]
```

IDLffDicomExStorScu::Cleanup - Performs all cleanup on the object.

```
OBJ_DESTROY, Obj
or
Obj->[IDLffDicomExStorScu::]Cleanup( )
(In a lifecycle method only.)
```

IDLffDicomExStorScu::ClearProperty - Resets the values of all properties of the IDLffDicomExQuery object to their default values.

```
Obj->[IDLffDicomExStorScu::]ClearProperties
```

IDLffDicomExStorScu::GetProperty - Retrieves the value of an IDLffDicomExStorScu property.

```
Obj->[IDLffDicomExStorScu::]GetProperty
[, PROPERTY=variable]
```

IDLffDicomExStorScu::Init - Initializes the IDLffDicomExStorScu object. The IDLffDicomExStorScu object implements a local DICOM Storage SCU that allows you to transmit files to a remote node that is configured as a DICOM Storage SCP available in IDL's DICOM Network Service configuration.

```
Obj = OBJ_NEW('IDLffDicomExStorScu')
or
Result = Obj->[IDLffDicomExStorScu::]Init( )
(In a lifecycle method only.)
```

IDLffDicomExStorScu::Send - Transfers DICOM files to a remote Storage SCP node.

```
Result = Obj->[IDLffDicomExStorScu::]Send(Files)
```

IDLffDicomExStorScu::SetProperty - Sets the value of a property or group of properties for the IDLffDicomExStorScu object.

```
Obj->[IDLffDicomExStorScu::]SetProperty
[, PROPERTY=value]
```




Index

A

- AddGroup method
 - IDLffDicomEx, [86](#)
- AddPrivateGroup method
 - IDLffDicomEx, [93](#)
- AddPrivateSequence method
 - IDLffDicomEx, [100](#)
- AddSequence method
 - IDLffDicomEx, [104](#)
- Application Entity
 - about, [25](#)
 - configuring
 - new, [26](#)
 - service list name, [28](#)
 - service list type, [28](#)
 - title, [27](#)
 - default entities, [25](#)
 - defining Storage SCP AE, [44](#)
 - deleting, [29](#)
 - Echo SCU, [48](#)
 - modifying, [28](#)
 - network service entities, [21](#)
 - Query Retrieve SCU, [30](#)
 - Query SCP, [30](#)
 - Query SCU, [33](#)
 - service class provider (SCP), [25](#)
 - service class user (SCU), [25](#)
 - Storage SCP
 - local, [22](#)
 - remote, [43](#)
 - Storage SCU, [43](#)
- attributes
 - DICOM
 - group, element, [67](#)
 - list of, [300](#)

- private
 - described, 67
 - querying, 176
 - reading, 136
 - writing, 191
- public
 - described, 67
 - querying, 179
 - reading, 152
 - writing, 201
- structure of, 67
- viewing, 121

C

ChangeTransferSyntax method
IDLffDicomEx, 108

classes

- file format
IDLffDicomEx, 64

cloning

- DICOM file, 164

Commit method

- IDLffDicomEx, 115

Control Port, 23

copying

- DICOM attributes, 116
- DICOM file, 164

CopyTags method

- IDLffDicomEx, 116

creating

- Application Entity, 26
- DICOM file, 165

D

depot directory, 50

DICOM

- Application Entities. *See* Application Entity attribute

- defined structure, 67
- list, 300

DICOM Network Services utility, 13

IDLffDicomEx object, 64

IDLffDicomExCfg object, 212

IDLffDicomExQuery object, 239

IDLffDicomExStorScu object, 283

licensing requirements, 11

Network Services

- about, 13
- querying for files, 30
- receiving files, 22
- sending files, 43
- supported, 14
- troubleshooting
 - echo operation, 48
 - retrieval operation, 40
 - send operation, 47
 - Storage SCP Service, 50
- utility, 18

Read/Write

- assigning SOP class, 171
- cloning file, 164
- committing file changes, 115
- copying attributes, 116
- creating file, 165
- getting pixel data, 129
- non-standard files, 168
- object class, 64
- read-only file, 163
- recovering files, 168
- RGB pixel data, 186
- setting pixel data, 182
- transfer syntax, 68

standards web site, 300

Storage SCP Service, 49

VR (value representation) list, 372

DICOM Network Services utility, 30

- about, 13

- configuring Query SCU, 33

- creating custom query, 36

- Echo SCU, 48
- license key, 11
- local versus system mode, 18
- locating configuration files, 58
- locating Storage SCP directory, 60
- managing Storage SCP Service, 51
- Query Retrieve SCU tab, 33
- starting, 18
- Storage SCU tab, 43
- DICOM object
 - IDLffDICOMEx, 64
- DICOMEX_GETCONFIGFILEPATH, 58
- DICOMEX_GETSTORSCPDIR, 60
- dicomex_importimage_doc.pro, 186
- DICOMEX_NET, 62

E

- Echo SCU, 48
- EnumerateTags method
 - IDLffDicomEx, 121
- examples
 - DICOM
 - dicomex_importimage_doc.pro, 186
 - filter_clonedicom_doc.pro, 132

F

- file
 - compression (DICOM), 68
- filter_clonedicom_doc.pro, 132

G

- GetDescription method
 - IDLffDicomEx, 127
- GetPixelData method
 - IDLffDicomEx, 129
- GetPrivateValue method
 - IDLffDicomEx, 136

- GetPrivateValueCount method
 - IDLffDicomEx, 140
- GetPrivateValueLength method
 - IDLffDicomEx, 143
- GetPrivateVR method
 - IDLffDicomEx, 148
- GetValue method
 - IDLffDicomEx, 152
- GetValueCount method
 - IDLffDicomEx, 155
- GetValueLength method
 - IDLffDicomEx, 157
- GetVR method
 - IDLffDicomEx, 161
- graphics
 - image file formats
 - DICOM
 - reading, 163
 - writing, 165
 - groups, definition in DICOM, 67

/

- IDLffDicomEx
 - class, 64
 - license key, 11
 - methods, 65
 - AddGroup, 86
 - AddPrivateGroup, 93
 - AddPrivateSequence, 100
 - AddSequence, 104
 - ChangeTransferSyntax, 108
 - Cleanup, 114
 - Commit, 115
 - CopyTags, 116
 - EnumerateTags, 121
 - GetDescription, 127
 - GetPixelData, 129
 - GetPrivateValue, 136
 - GetPrivateValueCount, 140
 - GetPrivateValueLength, 143

- GetPrivateVR, 148
- GetProperty, 151
- GetValue, 152
- GetValueCount, 155
- GetValueLength, 157
- GetVR, 161
- Init, 163
- QueryPrivateValue, 176
- QueryValue, 179
- SetPixelData, 182
- SetPrivateValue, 191
- SetProperty, 199
- SetValue, 201
- properties, 64
- supported platforms, 11

IDLffDicomExCfg

- class, 212
- methods, 212
 - Cleanup, 216
 - Commit, 217
 - Echo, 218
 - GetApplicationEntities, 220
 - GetApplicationEntity, 222
 - GetServiceLists, 223
 - GetServiceTypes, 224
 - GetValue, 225
 - Init, 227
 - IsDirty, 230
 - RemoveApplicationEntity, 231
 - SetApplicationEntity, 232
 - SetValue, 234
 - StorageScpService, 236
- supported platforms, 11

IDLffDicomExQuery

- class, 239
- methods, 240
 - Cleanup, 260
 - ClearProperties, 261
 - GetProperty, 262
 - Init, 263
 - Query, 265

- QueryModelsSupported, 270
- Retrieve, 273
- SetProperty, 282
- properties, 242
- supported platforms, 11

IDLffDicomExStorScu

- class, 283
- methods, 283
 - Cleanup, 289
 - ClearProperties, 290
 - GetProperty, 291
 - Init, 292
 - Send, 294
 - SetProperty, 298
- properties, 285
- supported platforms, 11

input/output

- DICOM files, 64

IP address, 16

L

- Largest Image Pixel Value, 79
- licensing
 - DICOM Network Services, 11
 - DICOM Read/Write, 11
 - IDLffDicomEx object, 11

M

- Macintosh
 - transfer syntax limits, 68
- metadata
 - DICOM file, 168
- methods
 - IDLffDicomEx
 - AddGroup, 86
 - AddPrivateGroup, 93
 - AddPrivateSequence, 100
 - AddSequence, 104

- ChangeTransferSyntax, 108
 - Cleanup, 114
 - Commit, 115
 - CopyTags, 116
 - EnumerateTags, 121
 - GetDescription, 127
 - GetPixelData, 129
 - GetPrivateValue, 136
 - GetPrivateValueCount, 140
 - GetPrivateValueLength, 143
 - GetPrivateVR, 148
 - GetProperty, 151
 - GetValue, 152
 - GetValueCount, 155
 - GetValueLength, 157
 - GetVR, 161
 - Init, 163
 - QueryPrivateValue, 176
 - QueryValue, 179
 - SetPixelData, 182
 - SetPrivateValue, 191
 - SetProperty, 199
 - SetValue, 201
 - IDLffDicomExCfg
 - Cleanup, 216
 - Commit, 217
 - Echo, 218
 - GetApplicationEntities, 220
 - GetApplicationEntity, 222
 - GetServiceLists, 223
 - GetServiceTypes, 224
 - GetValue, 225
 - Init, 227
 - IsDirty, 230
 - RemoveApplicationEntity, 231
 - SetApplicationEntity, 232
 - SetValue, 234
 - StorageScpService, 236
 - IDLffDicomExQuery
 - Cleanup, 260
 - ClearProperties, 261
 - GetProperty, 262
 - Init, 263
 - Query, 265
 - QueryModelsSupported, 270
 - Retrieve, 273
 - SetProperty, 282
 - IDLffDicomExStorScu
 - Cleanup, 289
 - ClearProperties, 290
 - GetProperty, 291
 - Init, 292
 - Send, 294
 - SetProperty, 298
 - multi-frame image
 - getting pixel data, 129
 - setting pixel data, 182
- ## P
- pixels
 - data
 - compression, 108
 - RGB, 186
 - preamble, 168
 - private DICOM attributes, 67
 - properties
 - IDLffDicomEx, 64
 - IDLffDicomExQuery, 242
 - IDLffDicomExStorScu, 285
 - public DICOM attributes, 67
- ## Q
- Query models, 36
 - Query SCP, 30
 - Query SCU, 33
 - query/retrieve (DICOM)
 - attribute matching, 37
 - configure query, 34
 - creating custom query, 36

- define remote machine, [30](#)
- overview, [33](#)
- query results, [35](#)
- retrieve files, [38](#)
- SCU configuration, [30](#), [33](#)
- supported query models, [36](#)
- wildcards, [37](#)

querying

- DICOM files, [33](#)

QueryPrivateValue method

- IDLffDicomEx, [176](#)

QueryValue method

- IDLffDicomEx, [179](#)

R

reading

- DICOM files, [163](#)

read-only

- DICOM file, [163](#)

RGB images

- DICOM pixel data, [186](#)

S

sequence

- described, [67](#)

private

- accessing nested, [144](#)

- adding, [100](#)

- adding repeating tags, [93](#)

public

- accessing nested, [159](#)

- adding, [104](#)

- adding repeating tags, [86](#)

- VR definition, [381](#)

service class provider (SCP), [25](#)

service class user (SCU), [25](#)

SetPixelData method

- IDLffDicomEx, [182](#)

SetPrivateValue method

- IDLffDicomEx, [191](#)

SetValue method

- IDLffDicomEx, [201](#)

Smallest Image Pixel Value, [80](#)

SOP class, DICOM file, [171](#)

starting

- DICOM Network Services utility, [18](#)

Storage SCP Service

- about, [49](#)

- depot directory, [50](#)

- image storage directory, [22](#)

- log files, [50](#)

- permissions, [49](#)

- start and stop, [51](#)

storscp.log, [50](#)

supported platforms

- DICOM toolkit, [11](#)

T

TCP/IP ports, [16](#)

transfer syntax, [68](#)

- changing, [108](#)

- Macintosh limitations, [68](#)

- support, [68](#)

troubleshooting

- DICOM Network Services

- echo operation, [48](#)

- retrieval operation, [40](#)

- send operation, [47](#)

- Storage SCP Service, [50](#)

V

Value Representations (VR)

- described, [67](#)

- list, [372](#)

W

in DICOM query, [37](#)

wildcards

