Programmer's Guide to the DUL Facility

DICOM Communication Using the Upper Layer Protocol

Stephen M. Moore

Mallinckrodt Institute of Radiology
Electronic Radiology Laboratory
510 South Kingshighway Boulevard
St. Louis, Missouri 63110
314/362-6965 (Voice)
314/362-6971 (FAX)

Version 2.10.0

August 3, 1998

This document describes a subroutine library which allows DICOM applications to establish Associations and to exchange data as described in Part 8 of the DICOM Standard.

---

1 Introduction

This facility implements the DICOM Upper Layer (DUL) protocol as specified in:

ACR-NEMA V3.0 - Digital Image and Communications in Medicine (DICOM)
Part 8: Network Communication Support for Message Exchange.

This implementation is based on the draft of the standard dated March 27, 1992.

This facility contains functions in three general classes:

- Establish Associations
- Close Associations
- Read/Write Data

Associations are supported on TCP/IP using Berkeley-style sockets. The facility has place holders for other protocols (OSI), but no other protocols are implemented as of this writing.

1.1 Network Environments

This facility provides mechanisms for supporting different types of network environments (OSI, TCP/IP). However, this version only implements support for TCP/IP. Each application is required to identify its network environment and at that time provide information that will define the application as an Association Requestor, an Acceptor or both. Once the network environment has been established, an application can request or accept Associations.

1.2 Establishing Associations

An Association is established by the following sequence:

- Requestor initiates Association by choosing a network protocol (TCP/IP) and calling a Receiver at a known address.
- Receiver examines Requestor's call and determines if it wants to communicate with Requestor.
- Receiver replies to Requestor.
- Requestor examines reply to determine if it is appropriate.

The Requestor transmits an Associate Request (RQ) Protocol Data Unit (PDU) as part of the initialization sequence. This PDU contains a number of fixed and variable fields which define the type of Association the Requestor wishes to establish. The Receiver parses the Request PDU and responds with an Associate Request Accept PDU or with an Associate Request Reject PDU.

Once the Association is established, the Requestor and Acceptor communicate as peers. The "driver" of the conversation is determined by the type of Association that was established as defined by the Associate RQ PDU. The DICOM Standard allows either application to send

request messages once the Association is established. Which application sends a request message is determined by SCU/SCP roles and by the type of request message. Some request messages are sent by SCUs; other request messages are sent by SCPs.

1.3 Closing Associations

Associations can be closed or destroyed by either peer. There are several mechanisms for closing an Association:

- The application that initiated the Association asks that the Association be torn down gracefully by transmitting a Release Request. It is expected that the peer will acknowledge this request.
- Either application asks that the Association be torn down immediately by transmitting an Abort Request (and then closing the network connection).
- Either application abnormally terminates the network connection (for example, the application dies).

This facility provides the functions for the Release and Abort requests. It also notifies the application when the peer requests a Release or Abort or when the network connection is simply closed to allow the application to recover gracefully. Applications should always use the functions DUL_DropNetwork and DUL_DropAssociation to perform the final close function for the network or Association. The act of aborting or releasing an Association does not destroy the data structures associated with the Association. To maintain symmetry, every network environment or Association (identified by a DUL_NETWORKKEY or DUL_ASSOCIATIONKEY) should be destroyed with the appropriate drop function.

1.4 Transmitting Data

DICOM message fragments (part of DICOM COMMAND or DICOM DATASET) are transmitted between applications by including them in P-DATA PDUs. Each fragment is stored in a Presentation Data Value Item (PDV). A number of PDVs can be linked to form a list of PDVs which can be transmitted in a single P-DATA PDU. This facility provides the means for applications to send a list of PDVs to another application using an established Association and to read a list of PDVs. It enforces the rules which limit the total length of a list of PDVs that can be sent in any single P-DATA PDU.

1.5 Presentation Contexts/SOP Classes

Parts 4 and 8 of the DICOM Standard define how SOP classes are requested on an Association and the Presentation Context that is maintained for each requested SOP class. Presentation Contexts are identified by a simple numerical value when the Association is established and can be used to switch context during the period when the Association is established.

This facility allows the user to request multiple SOP classes by maintaining a List of Presentation Contexts. The requestor creates a new Presentation Context for each SOP class to be requested

and passes the list to the DUL facility. The DUL response marks each SOP Class as accepted or rejected. Likewise, an acceptor can support multiple SOP classes and accept more than one Presentation Context. The DUL facility presents an acceptor with a list of proposed Presentation Contexts. The acceptor marks the Presentation Contexts to accepted and rejected and instructs the DUL facility to send the appropriate response PDU.

1.6 Implementation Details

This facility is based on a finite state machine as defined in Part 8 of the DICOM V3 Standard. The machine is driven by requests from the application. The DUL facility does not use interrupts to sense when data arrives on the network. It only checks for network events when the application calls a DUL function. Thus, it is up to the application to call DUL functions to request data. This facility provides no means to interrupt the application to inform it of an event (such as data arriving on the network or the network closing). There may be some time between when an event occurs (e.g., the network closed by peer) and when the DUL facility notifies the user of the event.

This facility implements the DICOM Upper Layer (DUL) protocol as specified in:

ACR-NEMA V3.0 - Digital Image and Communications in Medicine (DICOM) Part 8: Network Communication Support for Message Exchange.

This implementation is based on the draft of the standard dated March 27, 1992.

This facility contains functions in three general classes:

- Establish Associations
- Close Associations
- Read/Write Data

Associations are supported on TCP/IP using Berkeley-style sockets. The facility has place holders for other protocols (OSI), but no other protocols are implemented as of this writing.

2 Data Structures

2.1 Network and Association Keys

This facility defines two keys or handles that are opaque to the programmer. These are defined in the “dulprotocol” include file as:

DUL_NETWORKKEY
DUL_ASSOCIATIONKEY

These keys are used when a network is initialized or an Association is established to describe the network environment or Association. They refer to structures maintained by the facility.

All functions in this facility require that the user declare variables as pointers to these keys:

```
DUL_NETWORKKEY *network

DUL_ASSOCIATIONKEY *association
```

These keys are then passed by reference to the functions in this facility:

```
DUL_Function(&network, &association)
```

This rule is used for all functions, whether the object is to be created or written by the function or merely referenced by the function.

2.2 Associate Service Parameters

The parameters needed to describe an Association are conveyed in a structure named DUL_ASSOCIATESERVICEPARAMETERS. The “C” declaration for this structure is:

```
typedef struct {
  char applicationContextName[DUL_LEN_NAME + 1];
  char callingAPTitle[DUL_LEN_TITLE + 1];
  char calledAPTitle[DUL_LEN_TITLE + 1];
  char respondingAPTitle[DUL_LEN_TITLE + 1];
  unsigned short result;
  unsigned short resultSource;
  unsigned short diagnostic;
  char callingPresentationAddress[64];
  char calledPresentationAddress[64];
  LST_HEAD *requestedPresentationContext;
  LST_HEAD *acknowledgedPresentationContext;
  unsigned short maximumOperationsInvoked;
  unsigned short maximumOperationsPerformed;
  char callingImplementationClassUID[DICOM_UI_LENGTH +1];
  char callingImplementationVersionName[16 + 1];
  char calledImplementationClassUID[DICOM_UI_LENGTH + 1];
  char calledImplementationVersionName[16 + 1];
  unsigned long peerMaxPDU;
}   DUL_ASSOCIATESERVICEPARAMETERS;
```

The fields are defined below. Definitions in italics are quoted directly from part 8 of the DICOM V3 draft.

applicationContextName	The application context name. This is taken directly from the Standard.
callingAPTitle	This parameter identifies the Application Process (AP) that shall contain the requestor of the A-ASSOCIATE service.
calledAPTitle	This parameter identifies the Application Process that shall contain the intended acceptor of the A-ASSOCIATE service.

respondingAPTitle	This parameter identifies the AP that shall contain the actual acceptor of the A-ASSOCIATE service.
maxPDU	The maximum length of the list of Presentation Data Values you expect to receive in a P-DATA PDU.
result	This corresponds to the result field in an A-ASSOCIATE-RJ PDU in the event that an Association Request is rejected.
resultSource	This corresponds to the Source field in an A-ASSOCIATE-RJ PDU in the event that an Association Request is rejected.
diagnostic	This corresponds to the Reason/Diag field in an A-ASSOCIATE-RJ PDU in the event that an Association Request is rejected.
callingPresentationAddress	This parameter shall contain a structured destination address unambiguous within the global network address structure. This shall be either an OSI Presentation Address or a TCP/IP address.
calledPresentationAddress	This parameter shall contain a structured destination address unambiguous within the global network address structure. This shall be either an OSI Presentation Address or a TCP/IP Address.
requestedPresentationContext	A list of DUL_PRESENTATIONCONTEXT items which are the requested Presentation Contexts. A requestor creates this list. The DUL routines parse Associate RQ PDUs and creates this list to present to acceptors.
acknowledgedPresentationContext	A list of DUL_PRESENTATIONCONTEXT items which are the acknowledged Presentation Contexts. An acceptor creates this list by examining the list of requested Presentation Contexts and adding an entry for each item in the requested list. The Presentation Context can be either accepted or rejected
maximumOperationsInvoked	A placeholder for a value which is used when the user negotiates asynchronous operations. Because our toolkit does not support this, the value is forced to 0 and ignored. See Part 8, sections D.3.3.3.1 and D.3.3.4.
maximumOperationsPerformed	A placeholder for a value which is used when the user negotiates asynchronous operations. Because our toolkit does not support this, the value is forced to 0 and ignored. See Part 8, sections D.3.3.3.1 and D.3.3.4
callingImplementationClassUID	... identifies in a unique manner a specific class of implementation (Part 7, Section D.3.3.2). A caller writing a requesting application must fill in a legal UID. The facility will parse an A-ASSOCIATE RQ and present this value to a caller writing an accepting application.
callingImplementationVersionName	...an option (which) is provided to convey an Implementation Version Name (Part 8, Secion D.3.3.2). This is an optional field. A caller writing a requesting application should fill in "" or a legal string. The facility will parse an A-ASSOCIATE RQ and present this value to a caller writing an accepting application.
calledImplementationClassUID	The Implementation Class UID of the receiving application. A caller writing a receiving application must fill this field with a legal UID. The facility will parse an A-ASSOCIATE RP and present this value to a caller writing a requesting application.

calledImplementationVersionName	The implementation version name of the receiving application. This is an optional field which should be set to "" or a legal string by the writer of a receiving application. If the value is present in the A-ASSOCIATE RP, the facility will parse the value and present it to the caller writing a requesting application.
peerMaxPDU	The maximum PDU value that was transmitted by the peer application during association negotiation, either in an A-ASSOCIATE RQ or A-ASSOCIATE RP message.

In general, the requestor allocates this structure and fills in some of the fields (see DUL_RequestAssociation). The DUL facility uses this structure to construct an A-ASSOCIATE RQ PDU and transmits it to an acceptor. The DUL facility used by the acceptor parses the RQ PDU and returns the structure to the acceptor application. The acceptor examines the structure, fills in its fields and accepts the Association (if appropriate).

Several constants define maximum length fields as specified by the DICOM V3 Part 8 draft. These are DUL_LEN_NAME and DUL_LEN_TITLE.

The presentation addresses defined by the draft standard are "OSI Presentation" or "TCP/IP" addresses. This version of the software only supports TCP/IP. This facility uses the name of the node for the unambiguous TCP/IP address. The facility also includes the port number with this address as part of the calledPresentationAddress. Thus, to call an acceptor at node "fred" which is listening on port 104, you would use the following for the calledPresentationAddress:

```
fred:104
```

When the facility accepts a TCP/IP connection, it finds the name of the node requesting the connection based on the IP address and places that name in the "callingPresentationAddress" field. In this implementation, any domain information is stripped. The decision to place the port number with the calledPresentationAddress allows applications to specify a port number and allows the DUL facility to ignore the mapping from Title/Pr Address to port number.

This facility treats all of the items above which are declared as character arrays as NULL terminated ASCII strings. This allows the facility and applications to use the normal run-time string functions. The mapping to fixed and variable-length fields in the Associate RQ and AC PDUs is handled by the DUL software.

2.3 Reject and Abort Parameters

The DUL facility implements the reject and abort services with a common data structure DUL_ABORT. This structure's members are:

```
unsigned char   result
unsigned char   source
unsigned char   reason
```

When an application requests a reject or abort, it uses the structure defined above and fills in the appropriate fields to define the reason for the reject or abort. Note that result is used for rejects and is ignored for aborts. The include file defines macro constants which should be used when writing or reading this structure.

2.4 Presentation Data Values

A Presentation Data Value encompasses a single fragment of a DICOM COMMAND or DATASET. A COMMAND is defined by the data elements in group 0000. A DATASET consists of data elements not in the COMMAND group. A single Presentation Data Value should not contain both COMMAND and DATASET information.

This facility uses the DUL_PDV structure to describe an individual Presentation Data Value:

```
unsigned long                   fragmentLength
DUL_PRESENTATIONCONTEXTID       presentationContextID
DUL_DATAPDV                     pdvType
BOOLEAN                         lastPDV
void                            *data
```

The fields are defined as follows:

fragmentLength	The length of the data fragment (in bytes).
presentationContextID	The presentation context ID for this Presentation Data Value.
pdvType	An enumerated type that is one of: DUL_COMMANDPDV DUL_DATASETPDV
lastPDV	A BOOLEAN flag indicating if this is the last fragment in a COMMAND or DATASET.
data	The pointer to the actual data.

A single P-DATA PDU may contain multiple PDVs. The DUL_PDVLIST structure defines a list of PDVs:

```
unsigned long                   count;
void                            *scratch;
unsigned long                   scratchLength;
DUL_ABORTITEMS                  abort;
DUL_PDV                         *pdv;
```

The fields in the structure are defined as follows:

count	The number of PDVs in this list of PDVs.
scratch	A buffer allocated by the application when reading PDVs. Please refer to DUL_ReadPDVs.
scratchLength	The length of the scratch buffer allocated by the user. Please refer to DUL_ReadPDVs.
abort	The structure which contains the information about an abort request. This is filled in by the DUL_ facility when a caller attempts to read PDVs and the peer aborts the Association.
pdv	The pointer to the first PDV in the sequential list of PDVs. Individual PDV items are allocated sequentially beginning with this item.

3 Include Files

All applications that use the DUL facility should include these files in the following order:

```
#include “dicom.h”
#include “condition.h”
#include “lst.h”
#include “dulprotocol.h”
```

4 Return Values

DUL_ABORTEDREQUEST	Request for Association was aborted during the setup process.
DUL_APABORT	Indication of Service Provider initiated abort. Abort initiated by DICOM DUL function, not by an application.
DUL_ASSOCIATIONPARAMETERFAILED	Failed to return a value from the association parameter structure.
DUL_ASSOCIATIONREJECTED	Request for Association rejected by peer.
DUL_CODINGERROR	Software failed internal checks. Indicates coding error on the part of the facility writer.
DUL_FSMERROR	DUL FSM had a run time error. Probably indicates that an event happened that was not expected given the current state.
DUL_ILLEGACCEPT	An application registered as a requestor tried to accept an association.
DUL_ILLEGALKEY	Illegal key (DUL_NETWORKKEY or DUL_ASSOCIATIONKEY) passed to DUL function.
DUL_ILLEGALPARAMETER	Caller passed an illegal parameter (the value was probably bad).

DUL_ILLEGALPDU	Illegal or ill-formed PDU received from peer.
DUL_ILLEGALPDULENGTH	DUL facility detected a PDU from the network with an illegal length. Probably indicates the length was greater than the max value established during association negotiation.
DUL_ILLEGALREJECTREASON	Caller attempted to reject Association with illegal Reason parameter.
DUL_ILLEGALREJECTRESULT	Caller attempted to reject Association with illegal Result parameter.
DUL_ILLEGALREQUEST	Illegal request made by caller. For example, caller attempted to write PDU before an Association was established.
DUL_ILLEGALSERVICEPARAMETER	Function detected illegal service parameter. Nominally sensed when requesting or accepting Associations.
DUL_INCORRECTBUFFERLENGTH	Caller allocated incorrect buffer length when requesting data (for example, allocated too many or too few bytes for an integer).
DUL_INSUFFICIENTBUFFERLENGTH	Caller allocated insufficient buffer length to hold requested data.
DUL_LISTCREATEFAILED	A DUL function failed to create a new list.
DUL_LISTERROR	An error occurred in a DUL function when trying to perform a LST function
DUL_KEYCREATEFAILURE	Function failed to create a key, likely due to memory allocation failure.
DUL_MALLOCERROR	Function failed to allocate memory.
DUL_NETWORKCLOSED	Network connection is closed. No more communication possible.
DUL_NETWORKINITIALIZED	Attempt to initialize a network environment twice.
DUL_NOASSOCIATIONREQUEST	No outstanding requests for Associations for this network listener.
DUL_NOCONNECTION	Network connection refused during Association request (remote node did not respond to connect request).
DUL_NORMAL	Normal return from DUL function.
DUL_NOPDVS	No PDVs available in current buffer.
DUL_NULLKEY	Caller passed NULL key to DUL function.
DUL_READTIMEOUT	Network code timed-out when waiting for data.
DUL_PCTRANSLATIONFAILURE	Presentation context translation failure.
DUL_PDATAPDUARRIVED	P-DATA PDU arrived from connected peer.
DUL_PEERABORTEDASSOCIATION	Peer application aborted Association.
DUL_PEERDROPPEDASSOCIATION	Peer application dropped Association with notification.
DUL_PEERILLEGALXFERSYNTAXCOUNT	Peer supplied illegal number of transfer syntaxes.
DUL_PEERREQUESTEDRELEASE	Peer application requested release.
DUL_READTIMEOUT	Time-out when waiting for packet to arrive from network.
DUL_RELEASECONFIRMED	Peer confirmed Release Request.
DUL_REQUESTASSOCIATIONFAILED	Request to establish Association failed.
DUL_TCPINITERROR	TCP Network initialization error.
DUL_TCPIOERROR	TCP error when reading from or writing to network.
DUL_UNEXPECTEDPDU	Unexpected PDU received from peer.
DUL_UNKNOWNHOST	User attempted to make a connection to an unknown host.

DUL_UNKNOWNREMOTENODE	Association Request received from unknown remote node.
DUL_UNRECOGNIZEDAE	Unrecognized Application Entity type (not AE_REQUESTOR or AE_ACCEPTOR).
DUL_UNRECOGNIZEDPDUTYPE	Unrecognized PDU type received from peer.
DUL_UNSUPPORTEDNETWORK	User attempted to initialize (or use) an unsupported network type. This version of the code only implements a TCP/IP environment.
DUL_UNSUPPORTEDPEERPROTOCOL	DUL facility detected a PDU from a peer with an unsupported protocol.
DUL_WRONGASSOCIATIONSTATE	DICOM finite state machine in the wrong state for request made or received an unexpected event.
DUL_WRONGDATATYPE	Caller specified wrong data type for an attribute when trying to obtain data from DUL facility.

5 DUL Routines

This section provides detailed documentation for each DUL facility routine.

DUL_AbortAssociation

Name

DUL_AbortAssociation - abort an Association by sending an A-ABORT PDU and waiting for the network to close.

Synopsis

CONDITION DUL_AbortAssociation(ASSOCIATION_KEY **association)

association The handle for the Association to be aborted.

Description

DUL_AbortAssociation is called by an application to abort an Association. This function causes an A-ABORT PDU to be sent to the connected peer. The function then waits for the network connection to close or for the network timer to expire. After the network is closed or the timer expires, this function closes the applications connection to the network.

After the Association is aborted, the caller must destroy the reference to association by calling DUL_DropAssociation.

Notes

The DICOM Upper Layer protocol states that the user of the DUL service cannot specify a significant value for the reason field in the A-ABORT PDU. Therefore, this function provides no interface for providing the caller a means to specify the reason for the abort request.

Return Values

DUL_NORMAL
DUL_NULLKEY
DUL_ILLEGALKEY
DUL_NETWORKCLOSED

DUL_AcknowledgeAssociationRQ

Name

DUL_AcknowledgeAssociationRQ - acknowledge an Associationrequest and to establish an Association.

Synopsis

CONDITION DUL_AcknowledgeAssociationRQ(DUL_ASSOCIATIONKEY **association,
DUL_ASSOCIATESERVICEPARAMS *params)

association Caller's handle to Associationwhich is to be acknowledged (and therefore established).

params Parameters which describe the type of service to be handled by this Association.

Description

DUL_AcknowledgeAssociationRQ is used by an acceptor to acknowledge an Association Request. It uses the service parameters supplied in params to construct an A-ASSOCIATE AC PDU. This PDU is transmitted to the requestor, thereby establishing an Association. Once the Association is established, either application can transmit data (depending on the type of Association established).

This function is to be called after the application has received an Association Request via the function DUL_ReceiveAssociationRQ. The caller must use the same params buffer initialized by the call to DUL_ReceiveAssociationRQ. The caller is responsible for supplying these fields:

respondingAPTitle
maxPDUReceive
acknowledgedPresentationContext

The caller may offer a different application context by modifying the applicationContextName field of params.

Notes

The applicationContextName, abstractSyntax and transferSyntax fields of params are based on the ISO 8824 standard (as stated in Part 8 of the DICOM V3 draft). These identifiers are encoded as ASCII numeric strings with "." separators.

The maxPDUReceive field allows the application to define the maximum length of a list of PDVs it will receive in a single P-DATA PDU. This value is not necessarily the same as the value the peer has specified for its maximum length.

The acknowledgedPresentationContext is the list of Presentation Contexts that are either accepted or rejected.

Return Values

DUL_NORMAL
DUL_NULLKEY
DUL_ILLEGALKEY
DUL_ILLEGALREQUEST

DUL_AcknowledgeRelease

Name

DUL_AcknowledgeRelease - acknowledge a release request from a peer application by sending an A-RELEASE RP PDU.

Synopsis

CONDITION DUL_AcknowledgeRelease(DUL_ASSOCIATIONKEY**association)

association Caller's handle to Association which is currently active.

Description

DUL_AcknowledgeRelease is used by an application to acknowledge a peer application's request to release an active association. This function formulates and transmits an A-RELEASE RP PDU to the peer application and closes the network transport.

Notes

This function does not destroy the association key nor does it release any of the lists which are maintained in the association parameters for this association. The key and service parameters should be cleaned by calls to DUL_DropAssociation and DUL_ClearServiceParameters.

Return Values

```
DUL_NORMAL
DUL_NULLKEY
DUL_ILLEGALKEY
```

DUL_ClearServiceParameters

Name

DUL_ClearServiceParameters - clear the lists and other parameters created when an association is established.

Synopsis

CONDITION DUL_ClearServiceParameters(DUL_ASSOCIATESERVICEPARAMETERS *params)

params Pointer to caller allocated structure to be cleared.

Description

DUL_ClearServiceParameters is used by an application after an association is terminated (normally or abnormally). When an association is created, the DUL facility fills a parameters structure with information that describes the association. If this structure is to be reused for another association, these parameters should be cleared. This keeps the DUL facility from being fooled by parameters from a previous association. This is also good practice because it frees up memory that was allocated by the process of creating an association.

Return Values

DUL_NORMAL

DUL_Debug

Name

DUL_Debug - turn off or on debugging output from DUL facility.

Synopsis

void DUL_Debug(BOOLEAN flag)

flag Flag indicating if DUL facility is run in debug mode (TRUE) or silent mode (FALSE).

Description

DUL_Debug sets a flag in the DUL facility which puts it in the debug mode or silent mode. When in the debug mode, information of use to developers is printed to standard error.

DUL_DefaultServiceParameters

Name

DUL_DefaultServiceParameters - load a set of default parameters into a caller's service parameters structure.

Synopsis

void	DUL_DefaultServiceParameters(DUL_ASSOCIATESERVICEPARAMETERS *params)
params	Pointer to a structure in caller's space that will be loaded by this function.

Description

DUL_DefaultServiceParameters is used by applications that want to load a default set of parameters into a DUL_ASSOCIATESERVICE PARAMETERS structure. Such defaults include the maximum PDU length, the application context name, the implementation class as well as other values. The user will still need to fill in some values that cannot be loaded as defaults.

Return Values

None

DUL_DropAssociation

Name

DUL_DropAssociation - drop an Association without notifying the peer application and destroy the caller's reference to the Association.

Synopsis

CONDITION DUL_DropAssociation(DUL_ASSOCIATIONKEY **association)

association Caller's handle to the Association to be dropped.

Description

This function drops an Association without notifying the peer application and destroys the caller's Association Key. The caller's reference to the Association Key is also destroyed to prevent misuse.

This function is called after the application senses an abort or release from its peer or after the application has successfully aborted or released an Association.

Notes

The "normal" use of this function is after an Association has been released or aborted. The function can also be used to force the network connection to close without formally notifying the peer. This feature can also be used to test DUL implementations to determine how they handle network disruption.

This function is also used in the Unix environment for applications that wish to use the "fork" system call. If an application forks with an established Association, both the parent and child will have a handle that can be used to communicate with the peer. One of either the parent or child should call DUL_DropAssociation to destroy the Association and let the other process communicate with the peer.

Return Values

```
DUL_NULLKEY
DUL_ILLEGALKEY
DUL_NORMAL
```

DUL_DropNetwork

Name

DUL_DropNetwork - drop the network connection established by DUL_InitializeNetwork

Synopsis

CONDITION DUL_DropNetwork(DUL_NETWORKKEY **network)

network Caller's handle to the network environment.

Description

DUL_DropNetwork is used to release any resources associated with the network that were obtained when calling DUL_InitializeNetwork or other DUL network functions. This is a shutdown procedure that should be called as your application is exiting or when you have decided you want to perform no further DUL-based network activity.

When DUL_DropNetwork has disposed of network resources, it writes a NULL value into the caller's network handle to prevent future use by the caller.

Notes

This function is not called to drop individual associations. The effect may be to drop associations, but that is not the intent of this function.

Return Values

```
DUL_NULLKEY
DUL_ILLEGALKEY
DUL_NORMAL
```

DUL_DumpParams

Name

DUL_Dump_Params - dump the contents of a DUL_ASSOCIATESERVICEPARAMETERS structure.

Synopsis

void DUL_DumpParams(DUL_ASSOCIATESERVICEPARAMETERS *params)

params Pointer to caller's structure which contains parameters used to establish an Association.

Description

DUL_DumpParams examines the contents of a DUL_ASSOCIATESERVICEPARAMETERS structure and dumps the information to the standard output.

Notes

This is most often used as a debugging tool.

Return Values

None

DUL_InitializeNetwork

Name

DUL_InitializeNetwork - initialize network environment and return a handle which describes the environment.

Synopsis

CONDITION DUL_InitializeNetwork(char *networkType, char *mode, void *param,
int timeout, unsigned long options, DUL_NETWORKKEY **networkKey)

networkType	One of a set of predefined constants which describe the type of network environment requested.
mode	NULL-terminated ASCII string identifying the application as a requestor or acceptor. Caller should use one of the constants DUL_AEREQUESTOR, DUL_AEACCEPTOR, or DUL_AEBOTH.
param	A parameter which is specific to the network type, which may be needed to initialize the network.
timeout	Length of time (in seconds) for TIMER time-out. If 0, the function will use a default value.
options	Bitmask which describes options to be used when initializing the network.
networkKey	The handle created by this function and returned to the caller to access this network environment.

Description

DUL_InitializeNetwork identifies the network environment requested by the application and defines the mode of the application (requestor, acceptor or both). Network setup is performed by this function (such as setting up a socket to listen for inbound requests). After setup, the function creates a handle and returns it to the caller for use in establishing Associations.

networkType defines the type of network requested. The caller should use one of the constants defined in the include file for this facility. The only network type that is supported by this version of the software isDUL_NETWORK_TCP.

param is a network and mode-specific argument needed to initialize the network. If the network type is TCP/IP and the application is an acceptor, param is the address of an integer containing the port number used to accept TCP connections.

timeout is used to set a global value for timeouts for network operations for the DUL routines. This timeout may be overridden in individual functions by passing an explicit timeout parameter.

options defines special options which are used in the network environment. The caller should mathematically OR the desired options. Two types of options are defined in this version of the software. The caller

must use the flag DUL_ORDERBIGENDIAN defined in the DUL include file. This defines that the DUL protocol will be implemented using big-endian byte ordering of data on the network.

The second option controls how the DUL facility returns remote host names to callers (for example, when your application is accepting Associations from remote nodes). If the caller specifies no option, the DUL facility truncates remote host names after the machine name (wuerl.wustl.edu gets truncated to wuerl). If the caller specifies DUL_FULLDOMAINNAME, the DUL facility will not truncate the domain name. The actual name that gets returned will depend on how your system resolves host names and how data is entered in your host table. The effect of this option is seen when the caller invokes DUL_ReceiveAssociationRQ.

Notes

As a side effect, this function initializes the state table for the finite state machine that controls the DUL software.

Return Values

```
DUL_UNSUPPORTEDNETWORK
DUL_UNRECOGNIZEDAE
DUL_KEYCREATEFAILURE
DUL_NETWORKINITIALIZED
DUL_NORMAL
```

DUL_MakePresentationCtx

Name

DUL_MakePresentationCtx - create one presentation context for inclusion in an Association Request or Association Acknowledgement

Synopsis

CONDITION DUL_MakePresentationCtx(DUL_PRESENTATIONCONTEXT **ctx,
DUL_SC_ROLE proposedSCRole, DUL_SC_ROLE acceptedSCRole,
DUL_PRESENTATIONCONTEXTID ctxID, unsigned char result,
char *abstractSyntax, char *transferSyntax, ...)

ctx — Address of caller's presentation context pointer. This function allocates memory for a presentation context structure and places the address of the structure in the caller's ctx pointer.

proposedSCRole — The proposed service class role for this presentation context. If a user is requesting an association, the user can propose any role (as listed below). If the user is accepting an Association, the user should copy the proposed value from the Association Request into this argument. Legal values for this argument are:

```
DUL_SC_ROLE_DEFAULT
DUL_SC_ROLE_SCU
DUL_SC_ROLE_SCP
DUL_SC_ROLE_SCUSCP
```

acceptedSCRole — The service class role that is accepted by an application operating as an Association acceptor. Legal values are listed above for the proposedSCRole argument. Applications that are requesting Associations should use DUL_SC_ROLE_DEFAULT.

ctxID — A user-defined presentation context ID that identifies this presentation context. Context IDs are odd numbers between 1 and 255. An application that is an Association acceptor must use the same presentation context ID as proposd by the Association requestor.

result — For Association acceptors, a value which indicates if the presentation context was accepted or rejected. Legal values are:

```
DUL_PRESENTATION_ACCEPT
DUL_PRESENTATION_REJECT_USER
DUL_PRESENTATION_REJECT_NOREASON
DUL_PRESENTATION_REJECT_ABSTRACT_SYNTAX
DUL_PRESENTATION_REJECT_TRANSFER_SYNTAX
```

abstractSyntax — The unique identifier (defined by NEMA or privately) that identifies the proposed SOP class.

transferSyntax — One or more unique identifiers that identify the transfer syntax that is being proposed by an Association initiator. If the application is an Association acceptor, can only be a single UID. This set of transfer syntaxes must be terminated with the value NULL,

e.g.,DICOM_TRANSFERLITTLEENDIAN,
DICOM_TRANSFERLITTLEENDIANEXPLICIT, NULL

Description

DUL_MakePresentationCtx is used to construct a single presentation context by a DICOM Association Requestor or by an Association Acceptor. The caller supplies parameters which are used to construct a presentation context that can be stored in a DUL_ASSOCIATESERVERPARAMETERS structure. DUL_MakePresentationCtx allocates memory for a structure, fills in the structure and returns the address of the allocated structure to the caller.

Return Values

```
DUL_NORMAL
DUL_LISTCREATEFAILED
DUL_LISTERROR
DUL_MALLOCERROR
```

DUL_NextPDV

Name

DUL_NextPDV - return the next PDV that has been read from the network to the caller

Synopsis

CONDITION DUL_NextPDV(DUL_ASSOCIATIONKEY **association, DUL_PDV *pdv)

association	Caller's handle for the Association.
pdv	Pointer to DUL_PDV structure allocated by caller. This function fills in this structure upon successful completion.

Description

DUL_NextPDV returns the next PDV that has already by read from the network. The caller allocates a PDV structure which is filled in by DUL_NextPDV. If there is no PDV available, the function returns DUL_NOPDVS.

Notes

The behavior of this function and DUL_ReadPDVs is confusing. Users who want a PDV should call this function to see if a PDV exists in the current buffer. If no PDV exists, the user should call DUL_ReadPDVs to read another set of PDVs from the network and then call DUL_NextPDV again to actually retrieve the PDV from the buffer.

Return Values

```
DUL_NORMAL
DUL_NULLKEY
DUL_ILLEGALKEY
DUL_NOPDVS
```

DUL_ReadPDVs

Name

DUL_ReadPDVs - read the next available set of PDVs in one PDU.

Synopsis

CONDITION DUL_ReadPDVs(DUL_ASSOCIATIONKEY **association,
DUL_PDVLIST *pdvList, DUL_BLOCKOPTIONS block, int timeout)

association	Caller's handle for the Association used to read the PDVs.
pdvList	No longer used
block	Flag indicating how read for PDU should be done (blocking/non blocking) timeout Timeout in seconds if reading in non-blocking mode.

Description

DUL_ReadPDVs reads the next set of PDVs available on an association by reading the next PDU. The PDVs are maintained with the association key and are accessed by the user through DUL_NextPDV. That is, this call does not return a PDV directly to the caller.

The function can operate in blocking or non blocking mode when receiving data from the network. If the caller wishes not to block, the timeout argument defines how long DUL_ReadPDVs should wait before returning without having read a PDU.

Notes

The relationship between this function and DUL_NextPDV is more difficult than it should be.

The pdvList parameter is a remnant from previous code and should be eliminated in the future.

Return Values

```
DUL_NORMAL
DUL_NULLKEY
DUL_ILLEGALKEY
DUL_ILLEGALREQUEST
DUL_NETWORKCLOSED
DUL_PEERREQUESTEDRELEASE
DUL_PEERABORTEDASSOCIATION
DUL_PEERDROPPEDASSOCIATION
```

DUL_ReceiveAssociationRQ

Name

DUL_ReceiveAssociationRQ - read the next outstanding Association Request PDU for the purpose of accepting an Association.

Synopsis

```
CONDITION DUL_ReceiveAssociationRQ(DUL_NETWORKKEY **network,
                DUL_BLOCKOPTIONS block,
                DUL_ASSOCIATESERVICEPARAMETERS *params,
                DUL_ASSOCIATIONKEY  **association)
```

network	Caller's handle returned by call to DUL_InitializeNetwork. This handle describes the network environment.
block	Flag containing options for how long the function waits in the event there are no outstanding requests.
params	Pointer to area allocated by caller to hold parameters written by this function which describe the Association the requestor would like to establish.
association	Caller handle for association that is created by this function.

Description

DUL_ReceiveAssociationRQ is used to allow the caller to receive the next outstanding Association Request on an initialized network (network). When an Association Request is received, this function fills in part of the params structure allocated by the caller and returns to the caller. The function also creates a handle (association) that is used to identify this Association Request. This handle is to be used to accept or reject this Association Request.

block is a variable of type DUL_BLOCKOPTIONS (defined in the DUL include file). This variable can take on one of several values to indicate how the function should operate in the event that no Association request is immediately available. Legal values are:

DUL_BLOCKBlock indefinitely
DUL_NOBLOCKReturn immediately

When an Association Request is received, DUL_ReceiveAssociationRQ fills in the following fields in params:

applicationContextName	calledAPTitle
callingAPTitle	maxPDUsend
callingPresentationAddress	requestedPresentationContext

Return Values

DUL_NORMAL	DUL_NULLKEY
DUL_ILLEGALKEY	DUL_ILLEGALREQUEST
DUL_UNSUPPORTEDNETWORK	DUL_UNKNOWNREMOTENODE
DUL_NETWORKCLOSED	DUL_ABORTEDREQUEST

DUL_RejectAssociationRQ

Name

DUL_RejectAssociationRQ - reject an Association Request made by a requestor.

Synopsis

CONDITION DUL_RejectAssociationRQ(DUL_ASSOCIATIONKEY **association,
DUL_ABORTITEMS *abortItems)

association — Caller's handle to Association which is to be rejected. This handle would have been returned by DUL_ReceiveAssociationRQ.

abortItems — Pointer to a structure which gives the reason for rejecting the Association.

Description

DUL_RejectAssociationRQ constructs an A-ASSOCIATE-RJ PDU and transmits it to the requesting application, rejecting the Association. After the ASSOCIATE RJ PDU is transmitted, the function waits for the network (i.e., socket) to close or until the network timer expires.

The caller defines the nature of the reject by filling in the result and reason fields of abortItems. The result field of abortItems should take on one of the following constants:

```
DUL_REJ_RSLTPERMANENT
DUL_REJ_RSLTTRANSIENT
```

The reason field of abortItems should take on one of these constants:

```
DUL_REJ_SU_NOREASON
DUL_REJ_SU_UNSUP_APP_CTX
DUL_REJ_SU_UNRECOG_CALLINGAP
DUL_REJ_SU_UNRECOG_CALLEDAP
```

These constants are defined in "dulprotocol.h". They correspond to definitions which are found in Table 9.3.4-1 of Part 8 of the Standard.

Return Values

```
DUL_NORMAL
DUL_NULLKEY
DUL_ILLEGALKEY
DUL_ILLEGALREQUEST
DUL_ILLEGALREJECTREASON
DUL_NETWORKCLOSED
DUL_ILLEGALREJECTRESULT
```

DUL_ReleaseAssociation

Name

DUL_ReleaseAssociation - request the orderly release of an Association.

Synopsis

CONDITION DUL_ReleaseAssociation(DUL_ASSOCIATIONKEY **association)

association Caller's handle to Association to be released.

Description

DUL_ReleaseAssociation is used by an application to perform an orderly shutdown of an Association. This function sends a RELEASE-RQ PDU to the connected peer and waits for a reply. If the reply is a RELEASE-RP PDU, the Association is released. If the next PDU is something other than a RELEASE-RP PDU, the function informs the caller of the event. The caller may be forced to use other functions to release the Association.

Notes

This function should only be called by the requestor of an Association. The acceptor of an Association is not supposed to request a release.

Return Values

```
DUL_NORMAL
DUL_NULLKEY
DUL_ILLEGALKEY
DUL_PEERREQUESTEDRELEASE
DUL_NETWORKCLOSED
DUL_ILLEGALREQUEST
```

DUL_RequestAssociation

Name

DUL_RequestAssociation - request an Association with another node.

Synopsis

```
CONDITION DUL_RequestAssociation(DUL_NETWORKKEY **network,
                DUL_ASSOCIATESERVICEPARAMETERS *params,
                DUL_ASSOCIATIONKEY  **association)
```

network	Caller's handle which describes the network environment.
params	Pointer to list of parameters which describe the type of Association requested by the caller.
association	Handle created by this function and returned to the caller which describes this Association.

Description

DUL_RequestAssociation establishes a network connection with a "listening" application (an acceptor) and sends an A-ASSOCIATE RQ PDU for the purpose of establishing an Association. The caller is responsible for supplying an initialized network environment (network) and a list of parameters (params) which describe the desired Association. The function sends the A-ASSOCIATE RQ PDU to the acceptor and waits for a response. If the response is an ACCEPT, the Association is established and the function fills in the fields of params that indicate the type of Association offered by the acceptor. The function will also create a handle used by the application to reference this Association (association) and will return it to the caller.

If the Association is rejected, the function fills in the reason for the rejection by filling in the result, resultSource and diagnostic fields of params.

Notes

If the Association is accepted, the params structure will contain a list called acknowledgedPresentationContext. This is a list of all of the Presentation Context items received from the peer application. The result field in each presentation context item will indicate if the Presentation Context was accepted or rejected by the peer.

Return Values

```
DUL_NORMAL
DUL_NULLKEY
DUL_ILLEGALKEY
DUL_ILLEGALREQUEST
DUL_ASSOCIATIONREJECTED
DUL_NOCONNECTION
```

DUL_WritePDVs

Name

DUL_WritePDVs - write a list of PDVs on an active Association.

Synopsis

CONDITION DUL_WritePDVs(DUL_ASSOCIATIONKEY **association, DUL_PDVLIST *pdvList)

association	Caller's handle to an active Association.
pdvList	Pointer to structure which describes the list of PDVs to be written on the active Association.

Description

DUL_WritePDVs writes a list of PDVs on an active Association (association). The caller's pdvList arguments points to the list of PDVs to be written.

Notes

The peer application specifies the maximum length of a list of PDVs when the Association is established. This function will segment the caller's list of PDVs and send as many P-DATA PDUs as are needed to satisfy the maximum length constraint. Therefore, a call to DUL_WritePDVs may result in multiple P-DATA PDUs being transmitted to the peer application.

Return Values

```
DUL_NORMAL
DUL_NULLKEY
DUL_ILLEGALKEY
DUL_ILLEGALREQUEST
DUL_NETWORKCLOSED
```