PS3.19

DICOM PS3.19 ~~2026b~~2026c - Application Hosting

PS3.19: DICOM PS3.19 ~~2026b~~2026c - Application Hosting

Copyright © 2026 NEMA

A DICOM® publication

Table of Contents

Notice and Disclaimer 9
Foreword 11
1. Scope and Field of Application 13
2. Normative References 15
3. Definitions 17
4. Symbols and Abbreviations 19
5. Conventions 21
6. Application Hosting Overview 23
7. Hosted Application Life Cycle 25
7.1. Initialization 25
7.2. States 25
8. Interfaces 29
8.1. Application Interface 30
8.1.1. getState() : State 30
8.1.2. setState(newState : State) : boolean 30
8.1.3. bringToFront(requestedScreenArea : Rectangle) : boolean 31
8.2. Host Interface 31
8.2.1. generateUID() : UID 31
8.2.2. getAvailableScreen(appPreferredScreen : Rectangle) : Rectangle 31
8.2.3. getOutputLocation(preferredProtocols: ArrayOfString) : String 31
8.2.4. notifyStateChanged(state : State) : void 32
8.2.5. notifyStatus(status : Status) : void 32
8.3. DataExchange Interface 32
8.3.1. notifyDataAvailable(data : AvailableData, lastData : boolean) : boolean 36
8.3.2. getData(objectUUIDs : ArrayOfUUID, acceptableTransferSyntaxUIDs : ArrayOfUID, includeBulkData : boolean) : ArrayOfObjectLocator 36
8.3.3. getAsModels(objectUUIDs : ArrayOfUUID, classUID : UID, supportedInfosetTypes : ArrayOfMimeType) : ModelSetDescriptor 37
8.3.4. queryModel(models : ArrayOfUUID, xpaths : ArrayOfString) : ArrayOfQueryResult 37
8.3.5. queryInfoset(models : ArrayOfUUID, xpaths : ArrayOfString) : ArrayOfQueryResultInfoset 38
8.3.6. releaseData(objectUUIDs : ArrayOfUUID) : void 38
8.3.7. releaseModels(objectUUIDs : ArrayOfUUID) : void 38
9. Data Types and Structures 39
9.1. Arrayof[type] 39
9.2. AvailableData 39
9.2.1. ObjectDescriptor 39
9.2.2. Patient 39
9.2.3. Study 40
9.2.4. Series 40
9.3. MimeType 40
9.4. ModelSetDescriptor 40
9.5. ObjectLocator 41
9.6. QueryResult 41
9.7. QueryResultInfoset 41
9.8. Rectangle 41
9.9. State 42
9.10. Status 42
9.10.1. StatusType 42
9.11. UID 42
9.12. UUID 42
9.13. XPathNode 43
9.14. XPathNodeInfoset 43
9.15. XPathNodeType 43
10. Data Exchange Model Conventions 45
10.1. Coded Terminology 46
10.2. Person Name Components 47
A. Data Exchange Models 49

A.1. Native DICOM Model 49
A.1.1. Usage 49
A.1.2. Identification 49
A.1.3. Support 49
A.1.4. Information Model 50
A.1.5. Description 51
A.1.6. Schema 56
A.1.7. Examples 57
A.2. Abstract Multi-Dimensional Image Model 57
A.2.1. Usage 57
A.2.2. Identification 58
A.2.3. Support 58
A.2.4. Information Model 59
A.2.5. Description 59
A.2.6. Schema 64
A.2.7. Examples 66
A.2.7.1. Simple 3D Volume 66
A.2.7.2. Simple 4D Volume 67
A.2.7.3. 2D Ultrasound 67
A.2.7.4. 3D MR Metabolite Map - Single Component 68
A.2.7.5. 3D MR Metabolite Map - Multiple Component 68
B. Interface Definitions 69
B.1. Application Interface - Version 20100825 69
B.1.1. WSDL Definition of the Interface 69
B.1.2. Definition of Data Structures Used 74
B.1.2.1. Primary Definitions 74
B.1.2.2. Referenced Definitions 81
B.2. Host Interface - Version 20100825 82
B.2.1. WSDL Definition of the Interface 82
B.2.2. Definition of Data Structures Used 88
B.2.2.1. Primary Definitions 88
B.2.2.2. Referenced Definitions 97

List of Figures

1-1. Interface between Hosted Application and Hosting System. 13
1-2. Illustration of Platform Independence via Hosted Application Architecture. 13
7.1-1. Hosted Application Initialization Sequence 25
7.2-1. State Diagram of Hosted Applications. 27
8-1. Diagram of the Interface Between the Hosting System and the Hosted Application 30
8.3-1. Example File-based Data Exchange Sequence 33
8.3-2. Example Model-based Data Exchange Sequence (continued on next page) 34
8.3-2b. Example Model-based Data Exchange Sequence (continued from previous page) 35
A.1.4-1. Native DICOM Model 50
A.2.4-1. Abstract Multi-Dimensional Image Model 59
A.2.7.1-1. Simple 3D Volume Example 66
A.2.7.2-1. Simple 4D Volume Example 67
A.2.7.3-1. 2D Ultrasound Example 67
A.2.7.4-1. Single Component 3D MR Metabolite Example 68
A.2.7.5-1. Multiple Component 3D MR Metabolite Map Example 68

List of Tables

7.2-1. States 25
7.2-2. Transitions Between States 26
10.1-1a. Basic Coded Terminology Macro 46
10.1-1b. Enhanced Coded Terminology Macro 46
10.1-1. Coded Terminology Macro 47
10.2-1. Person Name Components Macro 47
A.1.5-1. Native DICOM Model 51
A.1.5-2. DICOM Data Set Macro 51
A.2.5-1. Abstract Image Model 59
A.2.5-2. Dimensional Data Macro 63

Notice and Disclaimer

The information in this publication was considered technically sound by the consensus of persons engaged in the development and approval of the document at the time it was developed. Consensus does not necessarily mean that there is unanimous agreement among every person participating in the development of this document.

NEMA standards and guideline publications, of which the document contained herein is one, are developed through a voluntary consensus standards development process. This process brings together volunteers and/or seeks out the views of persons who have an interest in the topic covered by this publication. While NEMA administers the process and establishes rules to promote fairness in the development of consensus, it does not write the document and it does not independently test, evaluate, or verify the accuracy or completeness of any information or the soundness of any judgments contained in its standards and guideline publications.

NEMA disclaims liability for any personal injury, property, or other damages of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, application, or reliance on this document. NEMA disclaims and makes no guaranty or warranty, expressed or implied, as to the accuracy or completeness of any information published herein, and disclaims and makes no warranty that the information in this document will fulfill any of your particular purposes or needs. NEMA does not undertake to guarantee the performance of any individual manufacturer or seller's products or services by virtue of this standard or guide.

In publishing and making this document available, NEMA is not undertaking to render professional or other services for or on behalf of any person or entity, nor is NEMA undertaking to perform any duty owed by any person or entity to someone else. Anyone using this document should rely on his or her own independent judgment or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstances. Information and other standards on the topic covered by this publication may be available from other sources, which the user may wish to consult for additional views or information not covered by this publication.

NEMA has no power, nor does it undertake to police or enforce compliance with the contents of this document. NEMA does not certify, test, or inspect products, designs, or installations for safety or health purposes. Any certification or other statement of compliance with any health or safety-related information in this document shall not be attributable to NEMA and is solely the responsibility of the certifier or maker of the statement.

Foreword

This DICOM Standard was developed according to the procedures of the DICOM Standards Committee.

The DICOM Standard is structured as a multi-part document using the guidelines established in [ISO/IEC Directives, Part 2].

DICOM® is the registered trademark of the National Electrical Manufacturers Association for its standards publications relating to digital communications of medical information, all rights reserved.

HL7® and CDA® are the registered trademarks of Health Level Seven International, all rights reserved.

SNOMED®, SNOMED Clinical Terms®, SNOMED CT® are the registered trademarks of the International Health Terminology Standards Development Organisation (IHTSDO), all rights reserved.

LOINC® is the registered trademark of Regenstrief Institute, Inc, all rights reserved.

1 Scope and Field of Application

This Part of the DICOM Standard defines an interface between two software applications. One application, the Hosting System, provides the second application with data, such as a set of images and related data. The second application, the Hosted Application, analyzes that data, potentially returning the results of that analysis, for example in the form of another set of images and/or structured reports, to the first application. Such an Application Program Interface (API) differs in scope from other portions of the DICOM Standard in that it standardizes the data interchange between software components on the same system, instead of data interchange between different systems. Hosted Application programs written to that standardized interface can 'plug-into' (see Figure 1-1) Hosting Systems. The notion of software add-ons or 'plug-ins' is quite common in the computing world, and has been successfully employed to extend the capabilities of web browsers, media players, graphical editors, publishing programs, etc. A Hosting System implementer needs only to create the standardized API once in order to support a wide variety of add-on Hosted Applications.

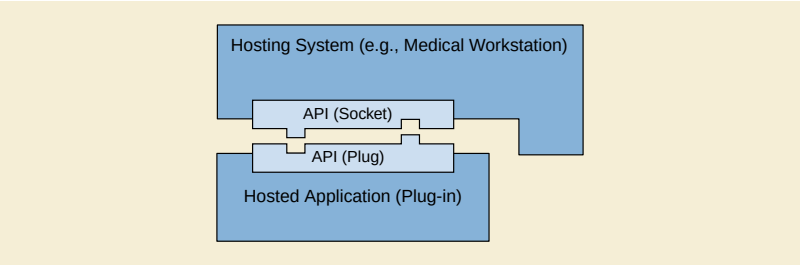


Figure 1-1. Interface between Hosted Application and Hosting System.

In the traditional 'plug-in' model, the 'plug-in' is dedicated to a particular host system (e.g., a web browsing program), and might not run under other host systems (e.g., other web browsing programs). PS3.19 defines a standardized API that may be implemented by any Hosting System. A 'plug-in' Hosted Application written to the standardized API would be able to run on any Hosting System that implements that standardized API (see Figure 1-2).

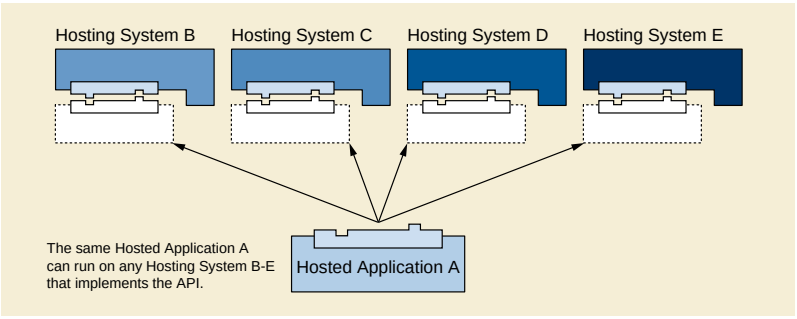


Figure 1-2. Illustration of Platform Independence via Hosted Application Architecture.

The design goals and assumptions for the API include:

- Language independence - the API is defined in such a way that programs written in any common programming language could utilize it.
- Platform independence - the API is defined in such a way that it is not dependent on any particular computing platform or operating system.
- Extensible - the API can be extended in a backward compatible fashion. Old applications still work even with new extensions in place, while new applications that are aware of the extensions can gain access to a richer set of functionality.
- Protected - the API design is consistent with later additions of mechanisms to protect intellectual property rights, and mechanisms that assure appropriate permissions and licenses are in place. The API should not interfere with common licensing systems.

- Secure - the Hosted Application's access to data on the Hosting System would be controlled via the API by the Hosting System. The Hosting System would be responsible for access controls and audit logging, since it is the one providing the data to the Hosted Application.
- Leverage Existing Technology - the API definition utilizes existing technology in common use, as far as practical, and does not define new methodologies.
- Simultaneous Launching - the Hosting System will be able to launch several instances of the same or of different Hosted Applications at the same time.
- Distributed Execution - although the API is designed for local execution, it does not prevent remote execution, where the Application is running on a different system from the Host.

PS3.19 specifies both the interactions and the Application Programming Interfaces (API) between Hosting Systems and Hosted Applications. PS3.19 also includes Normative and Informative Annexes that define the data models that are used by the API defined in this Part.

The API does not directly address workflow management, which is addressed by other DICOM Services.

2 Normative References

The following standards contain provisions that, through reference in this text, constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibilities of applying the most recent editions of the standards indicated below.

[ISO/IEC Directives, Part 2] ISO/IEC. 2021. 9.0. *Rules for the structure and drafting of International Standards*. http://www.iso.org/sites/directives/current/part2/index.xhtml .

[ISO 8822] ISO. 1988. *Information processing systems - Open Systems Interconnection - Connection oriented presentation service definition*.

[ISO/IEC 19757-2] ISO. 2008. *Information technology — Document Schema Definition Language (DSDL) — Part 2: Regular-grammar-based validation — RELAX NG*. http://www.iso.org/standard/52348.html .

[ISO/IEC 19757-3] ISO. 2016. *Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron*. http://www.iso.org/standard/55982.html .

[ITU-T X.667] ITU. 2012. *Information technology - Procedures for the operation of object identifier registration authorities: Generation of universally unique identifiers and their use in object identifiers*. http://www.itu.int/rec/T-REC-X.667/en .

[JIS X 0201] JIS. 1976. *Code for Information Interchange*.

[WSDL 1.1] W3C. 26 June 2007. *W3C Recommendation Web Services Description Language (WSDL) 1.1*. http://www.w3.org/TR/wsdl .

[XSD1] W3C. 28 October 2004. *XML Schema Part 1: Structures Second Edition. W3C Recommendation*. http://www.w3.org/TR/xmlschema-1/ .

[XPath 2.0] W3C. October 2016. *W3C Recommendation XML Path Language (XPath) 2.0*. http://www.w3.org/TR/xpath20/ .

[InfoSet] W3C. 4 February 2004. *W3C Recommendation XML Information Set*. http://www.w3.org/TR/xml-infoset/ .

[RFC2045] IETF. November 1996. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. http://www.rfc-editor.org/info/rfc2045 .

[RFC2046] IETF. November 1996. *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. http://www.rfc-editor.org/info/rfc2046 .

[RFC2048] IETF. November 1996. *Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*. http://www.rfc-editor.org/info/rfc2048 .

[RFC3240] IETF. February 2002. *Digital Imaging and Communications in Medicine (DICOM) - Application/dicom MIME Sub-type Registration*. http://www.rfc-editor.org/info/rfc3240 .

[RFC3986] IETF. *Uniform Resource Identifiers (URI): Generic Syntax*. http://www.rfc-editor.org/info/rfc3986 .

3 Definitions

For the purposes of this Standard the following definitions apply.

3.1 Presentation Service Definitions

This Part of the Standard makes use of the following terms defined in [ISO 8822]:

Transfer Syntax	See [ISO 8822].
Transfer Syntax Name	See [ISO 8822].

3.2 XML Infoset Definitions

This Part of the Standard makes use of the following terms defined in [InfoSet]:

Note

1. The concept of an XML Attribute is quite distinct from that of a DICOM Attribute.
2. To avoid confusion with the DICOM terms with similar names, the text of the DICOM Standard will use XML Element and XML Attribute when referring to these XML Infoset concepts. The appearance of Element or Attribute without the term XML in front of them generally refers to the DICOM concepts instead of the XML Infoset concepts.

XML Attribute	See [InfoSet].
XML Element	See [InfoSet].
XML Infoset	See [InfoSet].

3.3 DICOM Introduction and Overview Definitions

This Part of the Standard makes use of the following terms defined in PS3.1:

Attribute	See Attribute in PS3.1.
Service-Object Pair Class (SOP Class)	See Service-Object Pair Class in PS3.1.

3.4 DICOM Information Object Definition

This Part of the Standard makes use of the following term defined in PS3.3:

Attribute Tag	See Attribute Tag in PS3.3.

3.5 DICOM Data Structures and Encoding

This Part of the Standard makes use of the following terms defined in PS3.5:

Data Element	See Data Element in PS3.5.
Data Element Tag	See Data Element Tag in PS3.5.
Data Element Type	See Data Element Type in PS3.5.
Data Set	See Data Set in PS3.5.
Defined Term	See Defined Term in PS3.5.

Enumerated Value	See Enumerated Value in PS3.5.
Sequence of Items	See Sequence of Items in PS3.5.
Unique Identifier (UID)	See Unique Identifier in PS3.5.
Value Multiplicity (VM)	See Value Multiplicity in PS3.5.
Value Representation (VR)	See Value Representation in PS3.5.

3.6 Codes and Controlled Terminology Definitions:

This Part of the Standard makes use of the following terms defined in PS3.16:

Baseline Context Group Identifier (BCID)	See Baseline Context Group Identifier in PS3.16.
Defined Context Group Identifier (DCID)	See Defined Context Group Identifier in PS3.16.
Context Group	See Context Group in PS3.16.
Context Group Version	See Context Group Version in PS3.16.
Context ID (CID)	See Context ID in PS3.16.
Mapping Resource	See Mapping Resource in PS3.16.
DICOM Content Mapping Resource (DCMR)	See DICOM Content Mapping Resource in PS3.16.
Value Set	See Value Set in PS3.16.
Coding Scheme	See Coding Scheme in PS3.16.

3.7 Application Hosting Definitions

The following definitions are commonly used in this Part of the Standard:

Application Programming Interface (API)	A set of interface methods that Hosted Applications and Hosting Systems use to communicate with each other.
Hosted Application	An application launched and controlled by a Hosting System. The Hosted Application may utilize services offered by the Hosting System.
Hosting System	The application used to launch and control Hosted Applications. The Hosting System provides a variety of services such as DICOM object retrieval and storage for the Hosted Application. The Hosting System provides the infrastructure in which the Hosted Application runs and interacts with the external environment. This includes network access, database and security.

3.8 DICOM Service Class Definitions

This Part of the Standard makes use of the following terms defined in PS3.4:

Service-Object Pair Instance (SOP Instance)	See Service-Object Pair Instance in PS3.4.

4 Symbols and Abbreviations

The following symbols and abbreviations are used in this Part of the Standard.

ACR American College of Radiology

ASCII American Standard Code for Information Interchange

ANSI American National Standards Institute

API Application Programming Interface

BCID Baseline Context Group Identifier

CID Context ID

DCID Defined Context Group Identifier

DCMR DICOM Content Mapping Resource

DICOM Digital Imaging and Communications in Medicine

DSDL Document Schema Definition Languages

IEC International Electrotechnical Commission

IOD Information Object Definition

IANA Internet Assigned Numbers Authority

ISO International Standards Organization

LUT Lookup Table

MIME Multipurpose Internet Mail Extensions

NEMA National Electrical Manufacturers Association

OID Object Identifier (ISO 8824)

ROI Region of interest

SOP Service-Object Pair

SR Structured Reporting

UID Unique Identifier

UUID Universal Unique Identifier (ISO/IEC 11578)

URL/URI Uniform Resource Locator / Identifier

VM Value Multiplicity

VR Value Representation

WSDL Web Services Description Language

XSD XML Schema Definition

XML eXtensible Markup Language

XPath XML Path Language

5 Conventions

Terms listed in Section 3 Definitions are capitalized throughout the document.

6 Application Hosting Overview

This section describes the capabilities of the API, gives an example of the sequence of operations, and summarizes the remaining sections of this Part.

The APIs are shared by a Hosting System and one or more Hosted Applications.

The API is agnostic to the hardware platform, the operating system, and the GUI. The API supports requesting space in the GUI, if available. The API supports headless operation (i.e., no GUI).

The APIs are defined using Web Services Definition Language (WSDL) to be programming language, platform, and technology neutral. The APIs are designed to maximize language independence while minimizing the impact on efficiency of utilizing web services technology. The interfaces support both a networked file-based and a shared-memory interaction model. The API supports manual configuration, but not discovery.

The API can provide DICOM Data Sets and other data to the Hosted Application and can accept DICOM Data Sets and other data created by the Hosted Application, incrementally or upon completion. The Hosted Application has granular access to data provided by the Hosting System (e.g., single attributes, a subset of the pixel data, etc.) and only that data. The API utilizes DICOM semantics, but not necessarily DICOM network transfer syntax. The Hosting System provides a mechanism to the Hosted Application for generating UIDs.

The API allows the Hosting System to suspend and/or cancel the operation of the Hosted Application and regain user interface control. The API supports returning status information from the Hosted Application to the Hosting System and tracking the state of the Hosted Application.

The Hosting System has a mechanism to launch or connect to one or more Hosted Applications, verify that the Hosted Application has started successfully, and then pass the initial data objects. All interactions start in the Hosting System. A typical sequence of events is as follows:

1. The Hosting System identifies and locates the Hosted Application appropriate to the task and data using host-specific methods. Often the desired application is selected by the user of the system or is identified in a work list entry.

2. The Hosting System launches the application, essentially issuing a 'run' or 'exec' command, passing parameters that the Hosted Application uses to establish bilateral communications between the two.

3. The Hosting System uses the API to initiate a processing task in the Hosted Application and notifies it of its input data.

4. The Hosted Application uses the API to pull information from the Hosting System about the input data, including the location of the bulk pixel data.

5. The Hosted Application may use file I/O, memory mapping, or any other appropriate method to gain access to the bulk pixel data.

6. The Hosted Application may also use the API to inform the Hosting System of the status of the processing, for example progress, any warnings or errors encountered.

7. The Hosting System might use the API to suspend or cancel processing in the Hosted Application.

8. If the Hosting System suspended processing in the Hosted Application, it may use the API to instruct the Hosted Application to resume processing.

9. The Hosted Application, as it processes the input data, might create output objects, and use the API to inform the Hosting System of their existence.

10. The Hosting System uses the API to pull information about the output objects from the Hosted Application, including the location of the bulk data.

11. The Hosting system might use file I/O, memory mapping, or any other appropriate method to gain access to the output bulk data, if needed.

12. Once the Hosting System has pulled the output data from the Hosted Application, it uses the API to instruct the Hosted Application to wait for the next processing task (i.e., tells the Hosted Application to idle).

13. If the Hosting System has another task for the Hosted Application to perform, it may use the API to start that task, following this sequence of events beginning at Step 3.

14. When the Hosting System no longer needs the Hosted Application, it may use the API to request that the Hosted Application exit.

Section 7 describes in greater detail the Hosted Application Life Cycle.

Section 8 describes the base interfaces between the Hosting System and the Hosted Application.

Section 9 describes the custom data types and data structures used by the interfaces.

Section 10 describes the general form of models used by the model-based interfaces, and the conventions used in defining those models. The models defined by this Standard are described in the Annexes.

7 Hosted Application Life Cycle

7.1 Initialization

The Hosting System initializes a Hosted Application by issuing a run command or its equivalent (e.g exec function in the C language) with command line parameters to specify the end point references (URLs) to be used for the interfaces. One end point reference is used by the Hosted Application to access the Host interface provided by the Hosting System. The second end point reference is where the Hosting System will look for the Application interface provided by the Hosted Application. The Host and Application interfaces are described in Section 8. If issued from a command prompt or shell, the run command may appear as:

app--hostURL *url1*--applicationURL *url2*

Note

1. In this startup methodology, it is the Hosting System, not the Hosted Application that specifies both URLs. The Hosted Application must respond at the URL assigned to it by the Hosting System.

2. A Hosted Application implementation where the Hosted Application runs remotely or on an application server might utilize a startup or proxy application to appropriately map between the URL provided by the Hosting System and the actual URL that the Hosted Application is using.

Figure 7.1-1 shows a sequence diagram of Hosted Application initialization. Once the Hosted Application has initialized and is ready to begin processing data, it changes its state to IDLE and notifies the Hosting System of the state change using a call to the notifyStateChanged() method, thus informing the Hosting System that the Hosted Application is ready to go.

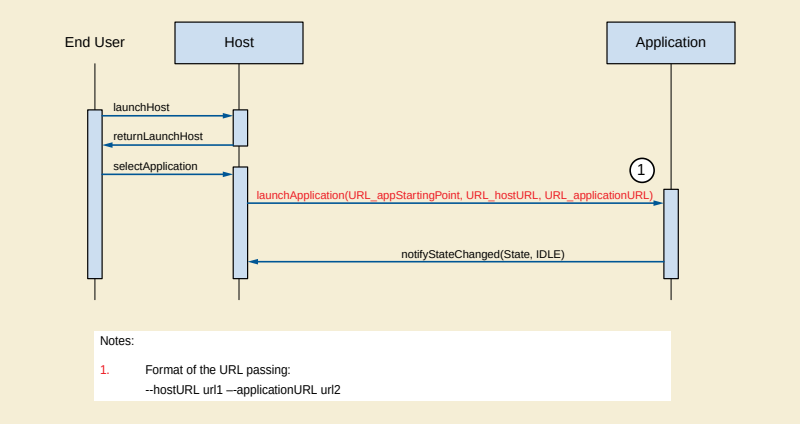


Figure 7.1-1. Hosted Application Initialization Sequence

7.2 States

Figure 7.2-1 shows the state diagram for a Hosted Application. The states are defined in Table 7.2-1.

Table 7.2-1. States

State	Description
IDLE	In IDLE state the Hosted Application is waiting for a new task assignment from the Hosting System. This is the initial state when the Hosted Application starts.
INPROGRESS	The Hosted Application is performing the assigned task.
SUSPENDED	The Hosted Application is stopping processing and is releasing as many resources as it can, while still preserving enough state to be able to resume processing.

State	Description
COMPLETED	The Hosted Application has completed processing, and is waiting for the Hosting System to access and release any output data from Hosted Application.
CANCELED	The Hosted Application is stopping processing, and is releasing all resources with no chance to resume processing.
EXIT	The terminal state of the Hosted Application.

The transitions between states are described in Table 7.2-2.

Table 7.2-2. Transitions Between States

State	Trigger	New State
not started	Hosting System launches the Hosted Application (e.g., run, exec).	IDLE
IDLE	Hosting System calls Application.setState (EXIT).	EXIT
IDLE	Hosting System calls Application.setState (INPROGRESS).	INPROGRESS
INPROGRESS	Hosting System calls Application.setState (SUSPENDED).	SUSPENDED
INPROGRESS	Hosting System calls Application.setState (CANCELED).	CANCELED
INPROGRESS	Hosted Application encounters an error that prevents further processing, but is still healthy enough to perhaps start another task. The Hosted Application shall report this error through a call to notifyStatus() with a statusType of FATALERROR prior to transitioning to the CANCELED state.	CANCELED
INPROGRESS	Hosted Application finishes its processing.	COMPLETED
SUSPENDED	Hosting System calls Application.setState (INPROGRESS).	INPROGRESS
SUSPENDED	Hosted Application encounters an error (e.g., during suspension) that prevents further processing, but is still healthy enough to perhaps start another task. The Hosted Application shall report this error through a call to notifyStatus() with a statusType of FATALERROR prior to transitioning to the CANCELED state.	CANCELED
SUSPENDED	Hosting System calls Application.setState (CANCELED).	CANCELED
COMPLETED	Hosting System calls Application.setState (IDLE), after capturing all pertinent output data from the Hosted Application.	IDLE
CANCELED	Hosted Application releases all resources and is ready for the next task.	IDLE

The Hosted Application notifies the Hosting System of all state transitions by calling the notifyStateChanged() method.

Note

If a Hosted Application does not respond to state change requests made by the Hosting System, the Hosting System may 'hard abort' the Hosted Application in some implementation specific manner, such as by killing the process in which the Hosted Application is executing.

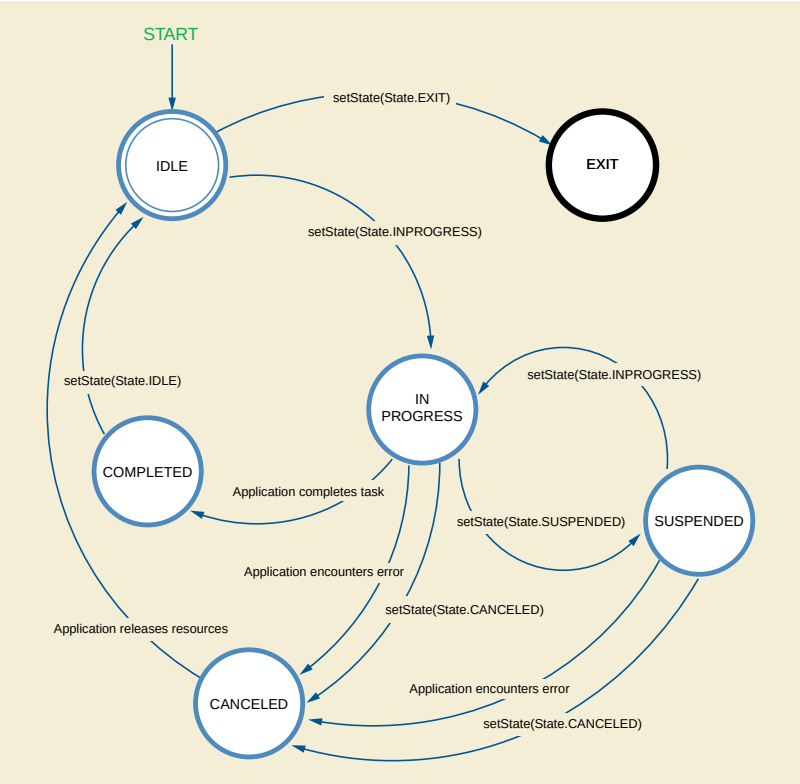


Figure 7.2-1. State Diagram of Hosted Applications.

8 Interfaces

There are three base interfaces defined in this Part, as shown in Figure 8-1. One, named "Application", represents the Hosted Application, and is utilized by the Hosting System to control the Hosted Application. The second, named "Host", represents the Hosting System, and is utilized by the Hosted Application to request services from and to notify the Hosting System of events during the execution of the Hosted Application. The third, named "DataExchange" is an interface used by both the Hosting System and the Hosted Application to communicate information about the data to be exchanged. Thus, the entire Hosted Application ("ApplicationService") implementation consists of the combination of the "Application" and "DataExchange" base interfaces, while the entire Hosting System ("HostService") implementation consists of the combination of the "Host" and "DataExchange" base interfaces.

The interfaces are defined as a set of methods using Web Services Description Language (WSDL). The implementers shall change the end point references (i.e., the "location" XML Attribute within the "address" XML Element within the "port" XML Elements of a "service" XML Element) in the WSDL specification as needed to deploy Hosted Applications and Hosting Systems that utilize these interfaces.

Note

The major (non-backward compatible) versions of the interfaces are reflected in the values of the "name" and "targetNamespace" XML Attributes of the "definitions" XML Element in the WSDL specification of the interfaces. Any changes to the interfaces that are not backward compatible will utilize new values for "name" and "targetNamespace" XML Attributes of the "definitions" XML Element.

Minor (backward compatible) versions of the interfaces may be reflected in the values of the "targetNamespace" XML Attribute of any new "schema" XML Element where new input or output data types are defined in the WSDL specifications, and/or in the values of the "name" XML Attributes of the "portType" and "service" XML Elements where new messages and operations are associated as new services in the WSDL specifications of the interfaces. To maintain backward compatibility, the names of existing elements, messages, and operations in the WSDL specification of the interfaces remain the same.

These methods utilize a set of basic data types and more complex data structures to communicate parameters, which are defined using XML Schemas. Later sections of this document provide more detailed descriptions of the interfaces and data structures, along with sequence diagrams illustrating how the interfaces are used.

The actual WSDL code and XML Schemas that specify this interface are defined in Annex B.

Note

1. WSDL is a platform and programming language independent means of specifying an interface between two cooperating applications. The applications need not be written in the same programming language.

2. The interfaces do not directly address reporting of SOAP communications problems. If a problem occurs in the communications between the Hosting System and a Hosting Application during the execution of a WSDL interface call, this should be reported by the SOAP libraries utilized by an implementation, e.g., thrown as an exception.

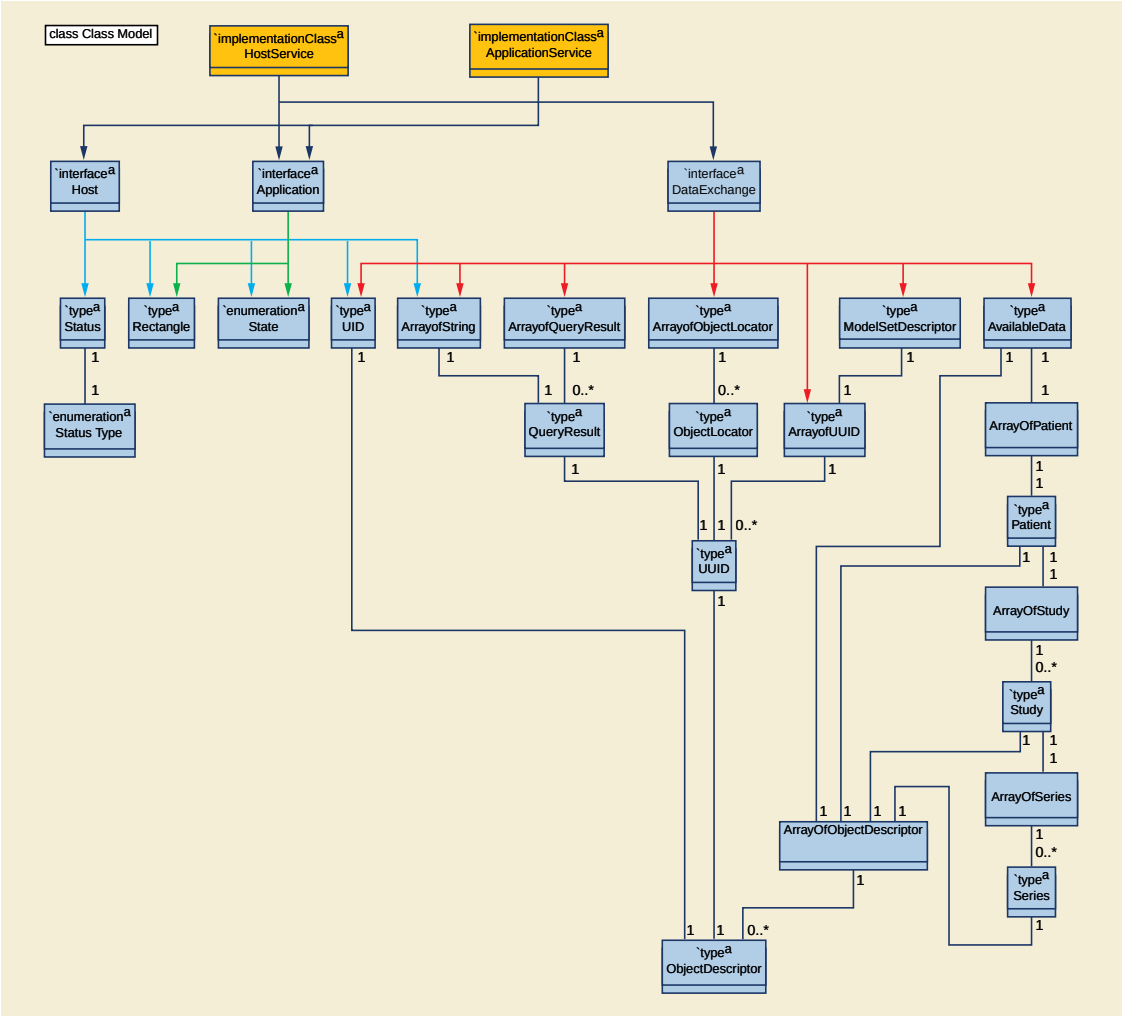


Figure 8-1. Diagram of the Interface Between the Hosting System and the Hosted Application

8.1 Application Interface

The following sections describe the methods of the Application interface.

8.1.1 getState() : State

The Hosted Application returns its current state to the caller.

This method may be called at any time.

Note

1. A Hosting System may use this method as an alternative to tracking Hosted Application state changes reported by the notifyStateChanged() method call.
2. A Hosting System may use this method to determine if a Hosted Application is still in operation (i.e., did not die without calling the notifyStateChanged() method with an EXIT state).

8.1.2 setState(newState : State) : boolean

The Hosting System requests that the Hosted Application switch to the newState.

The Hosted Application returns TRUE from the method if the Hosted Application received the request, and the requested state change is allowed in the state diagram. Otherwise, the method returns FALSE. A return value of TRUE does not indicate that the state of the Hosted Application has changed to the newState; it merely indicates that the requested state change is valid, and will be made at the soonest opportunity. Once the Hosted Application switches to the requested state, it shall inform the Hosting System through the notifyStateChanged() method of the Host interface.

Note

The asynchronous response to a state change is intended to minimize blocking the Hosting System while waiting for a potentially time-consuming state change in the application.

The Hosted Application shall ignore any setState() and return TRUE when the Hosted Application is already in requested state (i.e., this is a repeated call with the same state).

If the Hosted Application receives a second setState() request for a different state prior to completing a previous request, then the Hosted Application shall abort or ignore the previous request, and begin processing the latest request.

This method may be called at any time, however may not have any effect (other than a return of FALSE) if the requested new state is not an allowed transition from the current state.

8.1.3 bringToFront(requestedScreenArea : Rectangle) : boolean

By calling this method, the Hosting System is asking the Hosted Application to take whatever steps are needed to make its GUI visible as the topmost window, and to gain focus.

If possible, the Hosted Application shall resize and reposition itself to the requestedScreenArea. If requestedScreenArea is missing or null, the Hosted Application may retain its current size and location on the screen.

The method returns TRUE if the Hosted Application received the request and will act on it. Otherwise it returns FALSE.

A Hosted Application shall act on this method if the Hosted Application is in the IDLE or INPROGRESS state. A Hosted Application is not required to act on this method if the Hosted Application is not in the IDLE or INPROGRESS state.

A Hosted Application that has no GUI (e.g., a headless analysis application), where becoming visible and gaining focus has no meaning, shall always return TRUE from this method.

8.2 Host Interface

The following sections describe the methods of the Host interface.

8.2.1 generateUID() : UID

Returns a newly created DICOM UID that the Hosted Application might use, e.g., to create new data objects and structures.

This method may be called at any time.

8.2.2 getAvailableScreen(appPreferredScreen : Rectangle) : Rectangle

The Hosted Application supplies its preferred screen size in the appPreferredScreen parameter. The Hosting System may utilize this information as a hint, but may return a window location and size that best suits the Hosting System's GUI.

The method returns the window location and size that the Hosting System would prefer that the Hosted Application utilize. However, there are no requirements that the Hosted Application act on that information.

This method may be called at any time.

8.2.3 getOutputLocation(preferredProtocols: ArrayOfString) : String

The method returns a URI that a Hosted Application may use to store output that it may provide back to the Hosting System (e.g., in response to a getData() call).

The Hosted Application indicates, in order of preference, the protocols it can use to store data. The Hosted Application shall at least support both the http: and the file: protocols. The Hosting System selects the most appropriate protocol, potentially taking into account system or security considerations as well as the order of preference. The Hosting System uses the selected protocol in setting up the resources and generating the URI returned to the Hosted Application.

Note

1. There may be limitations when using the http: protocol when compared to the file: protocol. Some functions that might work with a file: protocol such as seek, rewrite, and delete, may not work with the http: protocol. The Hosted Application should assume that it can only write once in sequential order when the returned output location uses the http: protocol.

2. If any authentication information is needed in order to access the data, this authentication information may be included in the URI.

The Hosting System shall keep the URI active while the Hosted Application is in any state other than IDLE or EXIT, or until such time that the Hosted Application returns the URI to the Hosting System (e.g., in an ObjectLocator returned to the Hosting System in response to a getData() call). The disposition of the data that the Hosted Application sends to this URI is the responsibility of the Hosting System after the Hosted Application transitions to the IDLE state or after the Hosted Application returns the URI to the Hosting System (e.g., in an ObjectLocator returned to the Hosting System in response to a getData() call). After the Hosted Application transitions to IDLE state, the Hosting System need not keep the URI active.

The Hosted Application shall only call this method if the Hosted Application is at the INPROGRESS or COMPLETED states.

8.2.4 notifyStateChanged(state : State) : void

The Hosted Application shall invoke this method each time the Hosted Application successfully transitions to a new state. The new state is passed in the state parameter.

Note

While all Hosting Systems need to accept this interface call method, they may track the current Application State in other ways, such as by polling for the state using the Application getState() method.

8.2.5 notifyStatus(status : Status) : void

The Hosted Application may inform the Hosting System of notable events that occur during execution by invoking this method, passing the information in the status parameter.

Note

The Hosting System typically would log these events to facilitate debugging. It may, at its discretion, display the information to the user.

This method may be called at any time.

8.3 DataExchange Interface

The interface used to exchange information about data being transferred between a source and a recipient is the same for both the Hosting System and the Hosted Application. Implementations of the Application interface shall also include the DataExchange interface. Implementations of the Host interface shall also include the DataExchange interface. In other words, the DataExchange interface is symmetric with respect to the Hosting System and Hosting Application.

The data being exchanged between the Hosting System and the Hosted Application can either be passed as files, or may be described in models that might be queried by the recipient.

Recipients that can parse DICOM objects are able to request the file-based methods. The sequence diagram in Figure 8.3-1 illustrates one potential exchange using the file-based methods.

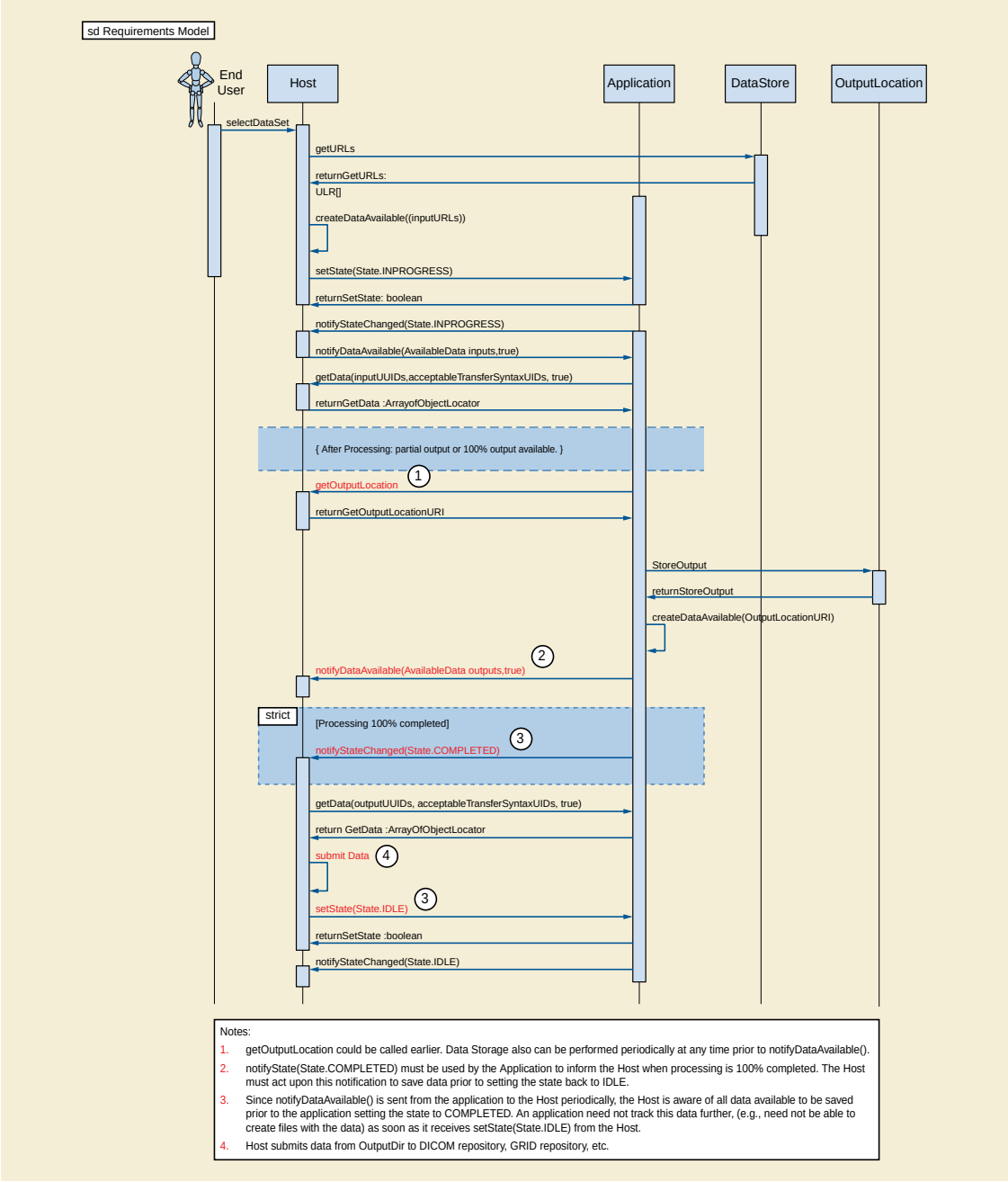


Figure 8.3-1. Example File-based Data Exchange Sequence

The advantage of using the model-based methods is that the recipient need not know how to parse the data formats, but instead can use commonly available tools for manipulating XML Infosets to extract data from the models.

The model-based interfaces can work with a variety of models. Particular models are identified by a UID. The models can either be an abstraction of the data, or can be a model of some native format. Models defined by the DICOM Standard are described in Annex A. Models are described as XML Infosets, even though the original data might never be actually represented in XML form. The source providing the data handles the mapping from the models back to the original data format.

Abstract models allow a recipient to work with data without regard to what its native form is. For example, data from a variety of image formats, such as DICOM, TIFF, JPEG, NIfTI, or Analyze, could be included in an abstract image model. The recipient can then work

with the data even though the recipient has no knowledge of how the data was natively represented. Abstract models may have been derived from data referenced in multiple ObjectDescriptors (e.g., multiple CT slices combined into a single volume).

Abstract models generally do not include the full richness of data that is available in native representations. For example, an abstract image model derived from DICOM data normally would include references to 'cooked' pixel data and its spatial organization, but might not include many of the modality-specific Attributes. To allow recipients to access such details the supplier of an abstract model can provide references to the ObjectDescriptors, in the form of UUIDs, from which that abstract model was derived. The recipient may gain access to any attribute of the original data formats through the source ObjectDescriptors.

The sequence diagram in Figure 8.3-2 illustrates one potential exchange using the model-based methods. It also illustrates the Hosted Application returning partial outputs, one potential way a Hosted Application might use the getOutputLocation() method, and potential uses of the releaseModel() and releaseData() methods.

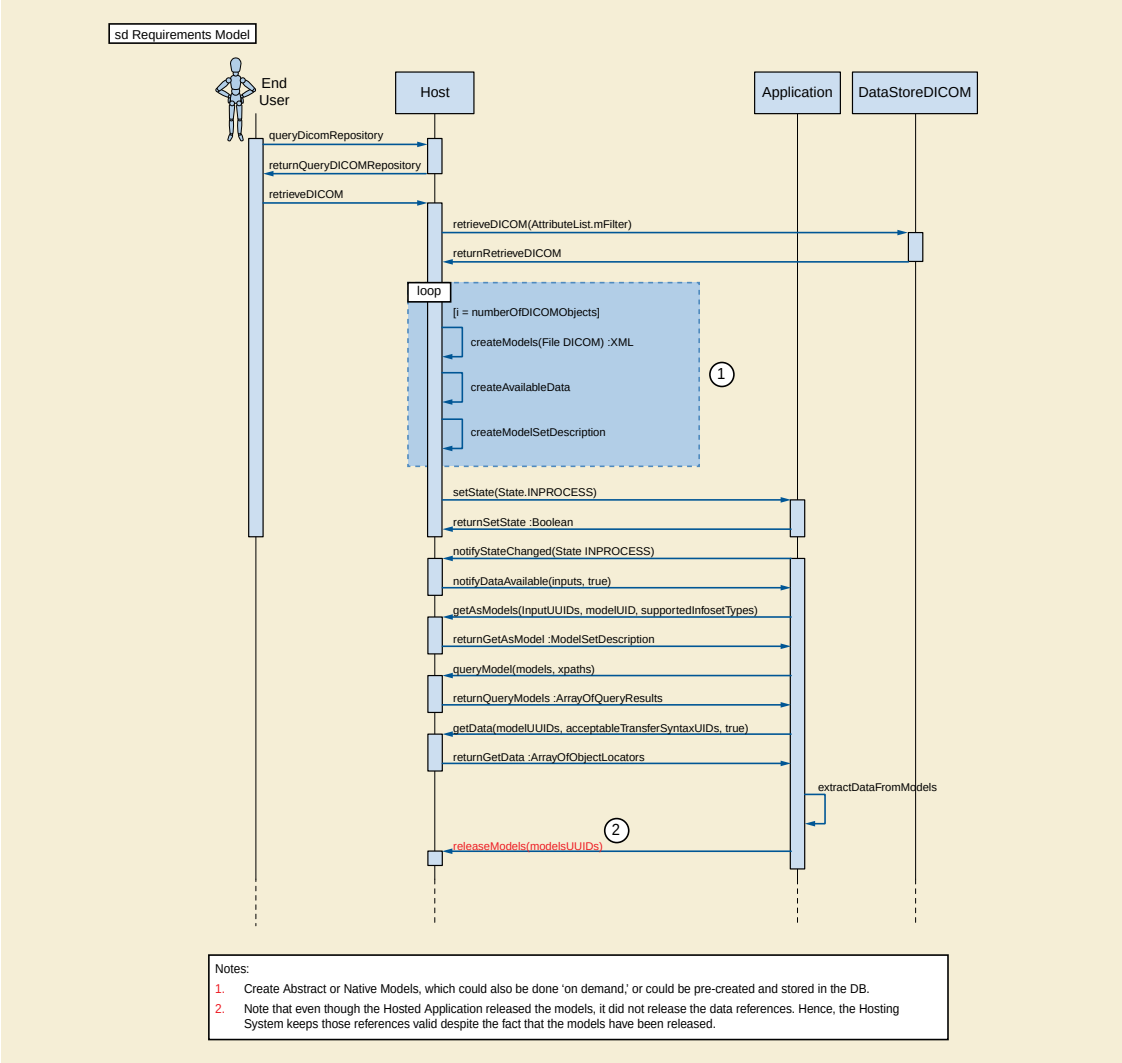


Figure 8.3-2. Example Model-based Data Exchange Sequence (continued on next page)

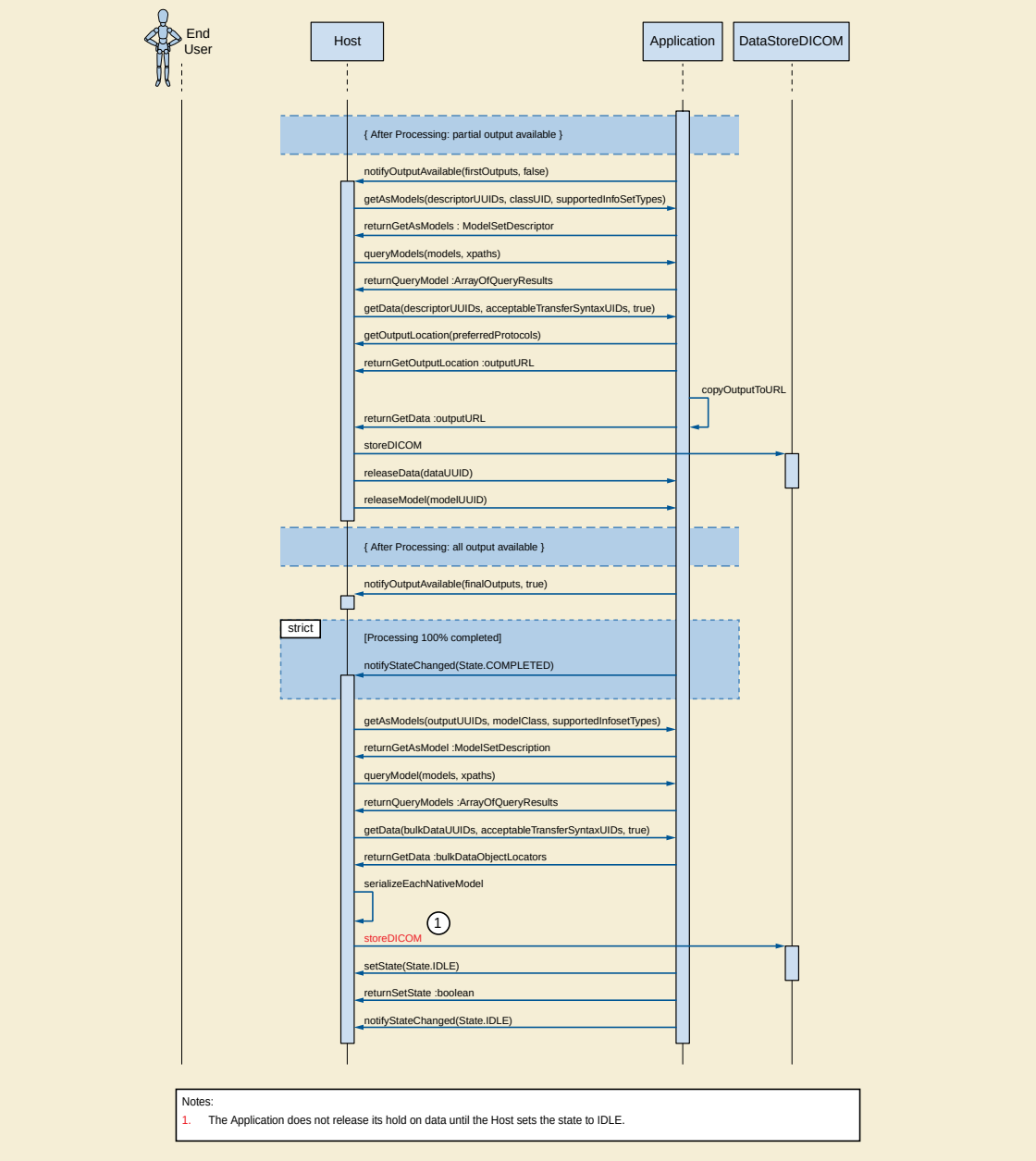


Figure 8.3-2b. Example Model-based Data Exchange Sequence (continued from previous page)

Hosting Systems shall support both the file-based and model-based interfaces, both as a data source as well as a data recipient.

Hosted Applications shall support at least one of the file-based or model-based interfaces, as either a data source or as a data recipient, as needed by the Hosted Application. If a Hosted Application supports the model-based interfaces, it shall support at least one of the models defined in Annex A. Hosted Applications may choose to implement only those portions of those interfaces that the Hosted Application actually uses; however, all interface methods that a Hosting System may call must be available for the Hosting System to call, even if the Hosted Application does not do anything but return appropriately.

The following sections describe the methods of the DataExchange interface.

8.3.1 notifyDataAvailable(data : AvailableData, lastData : boolean) : boolean

The source of the data calls this method with descriptions of the available data that it can provide to the recipient. If the source of the data expects that additional data will become available, it shall pass FALSE in the lastData parameter. Otherwise, it shall pass TRUE in the lastData parameter, and shall not make any further calls to notifyDataAvailable until after it has transitioned through the IDLE state once more.

The source of the data shall be able to provide the data in the form identified in the AvailableData structure.

A Hosting System uses this method to inform a Hosted Application of input data that the Hosted Application should work with. A Hosted Application uses this method to inform the Hosting System of outputs produced by the Hosted Application.

This method returns TRUE if the recipient of the data successfully received the AvailableData list. Otherwise this method returns FALSE.

Note

A Hosted Application that is recipient of this call, but that was unsuccessful in receiving the AvailableData list might report a reason for the failure in a notifyStatus method call.

The source of the data shall not include in AvailableData any references to data that were sent in a previous successful notifyDataAvailable call (i.e., one where the method call returned TRUE).

A Hosted Application shall not transition into the COMPLETED state if it has received or sent a notifyDataAvailable() call with a lastData indicator of FALSE.

The source of the data may call notifyDataAvailable() with an empty data list.

Note

Calling notifyDataAvailable() with an empty list is useful for setting the lastData indicator to TRUE.

This method shall only be called if the Hosted Application is at the INPROGRESS state.

8.3.2 getData(objectUUIDs : ArrayOfUUID, acceptableTransferSyntaxUIDs : ArrayOfUID, includeBulkData : boolean) : ArrayOfObjectLocator

The recipient of data invokes this method to gain access to binary data provided by the source of the data. The source of the data provides a URI that the recipient may open as a byte stream to retrieve the data.

Note

The provider of the data may delay the actual preparation of binary data until the recipient actually requests it.

The objectUUIDs array provides the UUIDs of the binary data that the source wishes to retrieve. Each of the UUIDs in that array are drawn either from the ObjectDescriptors provided in the AvailableData structure received by the recipient in one or more notifyDataAvailable() method calls, or from bulk data pointers in models accessed by the recipient.

If the UUID came from an ObjectDescriptor, the source returns ObjectLocators of the binary objects using the MIME content type and class UID listed in the ObjectDescriptor within the AvailableData structure associated with each UUID. If the UUID came from a Data Exchange Model, then the source returns the binary bulk data described within the model.

The recipient lists the desired Transfer Syntax for the bulk data via the acceptableTransferSyntaxUIDs parameter. The recipient shall list in order of preference in the acceptableTransferSyntaxUIDs parameter the UIDs of the Transfer Syntaxes that it will accept for the data represented by objectUUIDs. The provider of the data shall select and use the first transfer syntax in the list that it supports. For DICOM data, the provider of data shall as a minimum support the Explicit VR Little Endian transfer syntax. The acceptableTransferSyntaxUIDs may be empty for those MIME content types where Transfer Syntax has no meaning.

When retrieving binary data identified by a UUID from an ObjectDescriptor, if the recipient sets the includeBulkData flag to TRUE, then the source shall supply the bulk data within the data stream. Otherwise, the source may, but is not required to, omit bulk data such as pixel data.

Note

The includeBulkData flag is useful, for example, when the recipient wishes to work with the description of the pixel data in binary DICOM form, in order to decide whether or not to retrieve the pixel data itself.

The method returns one ObjectLocator for each UUID passed into the method within the objectUUIDs array. The ObjectLocator describes a file where the recipient can read in the data referred to by that particular object's UUID.

When the recipient is finished with data referred to by an ObjectLocator URI, it may call the releaseData() method to free up the resources being consumed to provide those URIs. Any data references not explicitly released by the recipient of the data are released implicitly when the Hosted Application enters the IDLE state.

The recipient may call getData() multiple times for data referenced by a given ObjectDescriptor or bulk data UUID. Each call to getData() shall be matched by either an explicit or implicit call to releaseData().

This method shall only be called if the Hosted Application is at the INPROGRESS or COMPLETED states. A Hosting System may also call this method when the Hosted Application is in the SUSPENDED state.

8.3.3 getAsModels(objectUUIDs : ArrayOfUUID, classUID : UID, supportedInfosetTypes : ArrayOfMimeType) : ModelSetDescriptor

The recipient of data invokes this method to ask that the source of the data provide the data referenced by objectUUIDs array as models of the type referenced by classUID. The objectUUIDs are drawn from the ObjectDescriptors passed to the recipient of the data in one or more notifyDataAvailable() calls.

The recipient of the data shall list in supportedInfosetTypes in order of preference the MIME types that the recipient can process as Infosets. Recipients of data shall support the "text/xml" MIME type, which shall always be included in the supportedInfosetTypes array. The provider of data shall select the first entry in that array that it supports.

The ModelSetDescriptor returned by this method contains the UUIDs of the models provided by the source, as well as the UUIDs of data objects referred to by the objectUUIDs array that could not be represented in the requested model.

The recipient may call getAsModels() multiple times for data referenced by a given UUID. Each successful call returns a different model UUID.

When the recipient is finished with a set of models, it may call the releaseModels() method to free up the resources being consumed to provide those models. Any models not explicitly released by the recipient of the data are released implicitly when the Hosted Application enters the IDLE state.

This method shall only be called if the Hosted Application is at the INPROGRESS or COMPLETED states. A Hosting System may also call this method when the Hosted Application is in the SUSPENDED state.

8.3.4 queryModel(models : ArrayOfUUID, xpaths : ArrayOfString) : ArrayOfQueryResult

The recipient of data invokes this method to request that the source of the data return the subset of data referred to in each of the XPath query strings passed in the xpath parameter from each of the models identified by the UUIDs passed in the model array. Each of the XPath query strings is applied to each of the models referred to in the model array.

The UUIDs passed in the model array shall be chosen from those returned by one or more getAsModels() method calls.

The results of the query are returned by the method as XML Infosets, encoded in XML returned as a string. Each result from a particular model UUID is returned as a QueryResult element in the returned array for each xpath string. In other words, the number of QueryResults returned is the number of UUIDs in the model array times the number of XPath queries strings in the xpath array.

Note

This method is principally used when the infoset type is "text/xml".

This method shall only be called if the Hosted Application is at the INPROGRESS or COMPLETED states. A Hosting System may also call this method when the Hosted Application is in the SUSPENDED state.

8.3.5 queryInfoset(models : ArrayOfUUID, xpaths : ArrayOfString) : ArrayOfQueryResultInfoset

The recipient of data invokes this method to request that the source of the data return the subset of data referred to in each of the XPath query strings passed in the xpath parameter from each of the models identified by the UUIDs passed in the model array. Each of the XPath query strings is applied to each of the models referred to in the model array.

The UUIDs passed in the model array shall be chosen from those returned by one or more getAsModels() method calls.

The results of the query are returned by the method as XML Infosets, encoded in XML, returned as a byte array encoded in the form negotiated during the getAsModel() call. Each result from a particular model UUID is returned as a QueryResultInfoset element in the returned array for each xpath string. In other words, the number of QueryResultInfoset structures returned is the number of UUIDs in the model array times the number of XPath queries strings in the xpath array.

Note

This method is principally used when the infoset type is not string based, for example a "application/fastinfoset". If called on a model using the "text/xml" infoset type, a conversion from a byte array to a string would be needed.

This method shall only be called if the Hosted Application is at the INPROGRESS or COMPLETED states. A Hosting System may also call this method when the Hosted Application is in the SUSPENDED state.

8.3.6 releaseData(objectUUIDs : ArrayOfUUID) : void

The recipient of data invokes this method to release access to binary data provided by the source of the data through a getData() call. The ArrayOfUUID identifies the data streams that the recipient is releasing. The UUIDs in this array shall be drawn from the locator fields in ObjectLocators returned by calls to getData().

8.3.7 releaseModels(objectUUIDs : ArrayOfUUID) : void

The recipient of data invokes this method to release access to models provided by the source of the data. The ArrayOfUUID identifies the models that the recipient is releasing. The UUIDs in this array shall be drawn from the models fields in ModelSetDescriptors returned by calls to getAsModels().

9 Data Types and Structures

9.1 Arrayof[type]

A wrapper object representing the encapsulation of an array of a specific type. Used in parameters to and return values from API functions to enable cross-platform compatibility. The wrapper contains a single field, which is an array of the type being stored. The field name is the Type name with the first letter in lowercase instead of uppercase.

Note

This construct was needed to support Microsoft® .NET language bindings even though it looks ugly in Java.

9.2 AvailableData

A data structure that communicates what data is available to the recipient. The data is organized in a hierarchical fashion, communicating patients, studies, series, and finally ObjectDescriptors that identify available data objects. The fields in the data structure are:

- ObjectDescriptors : ObjectDescriptor[] - An array of ObjectDescriptor data structures listing data that either applies to multiple patients, or does not fit into the patient / study / series hierarchy.
- Patients : Patient[] - An array of Patient data structures.

9.2.1 ObjectDescriptor

A data structure with the following fields:

- DescriptorUUID : UUID - the UUID that the interface utilizes to track this particular data object.
- MimeType : MimeType - the MIME content type of this particular data object, in its most natural form available from the source. The most natural form is typically the form in which the source maintains the data in its database, for example a DICOM file.
- ClassUID : UID - the UID that represents the class of this data object in the form described by mimeType. For objects whose mimeType refers to a data exchange model such as those defined in Annex A, this is the UID of that model. For objects whose mimeType is application/dicom, this is the SOP Class UID of the DICOM object. This may be empty for those objects whose MIME content types have no additional classes.
- TransferSyntaxUID : UID - the UID that represents the Transfer Syntax of this data object in the form described by mimeType. This may be empty for those objects of a MIME content type where Transfer Syntax has no meaning.
- Modality : String - the modality that best represents where this data originated from. Standard values are drawn from the Defined Terms listed for the Modality (0008,0060) Attribute in the Section C.7.3.1.1.1 “Modality” in PS3.3.

9.2.2 Patient

A data structure that communicates data for a particular patient. The fields in the data structure are:

- Name : String - The name of the patient, formatted as described for the PN VR in PS3.5. For DICOM SOP Instances this is the value of the Patient's Name (0010,0010) Attribute.
- ID : String - A string used as the identifier for a particular patient, formatted as described for the LO VR in PS3.5. For DICOM SOP Instances this is the value of the Patient ID (0010,0020) Attribute.
- AssigningAuthority : String - The organization who assigned the id to the patient, formatted as described for the LO VR in PS3.5. For DICOM SOP Instances this is the value of the Issuer of Patient ID (0010,0021) Attribute.
- Sex : String - The sex of the patient. For DICOM SOP Instances this is the value of the Patient's Sex (0010,0040) Attribute. In all other cases it shall take on the values permissible for the DICOM Sex (0010,0040) Attribute.
- BirthDate: String The birth date of the patient, formatted as described for the DA VR in PS3.5. For DICOM SOP Instances this is the value of the Patient's Birth Date (0010,0030) Attribute.

- ObjectDescriptors : ObjectDescriptor[] - An array of ObjectDescriptor data structures listing data that applies to this patient, but that do not apply to any particular study of this patient.
- Studies : Study[] - An array of Study data structures.

At least one of objectDescriptors or studies shall be present.

9.2.3 Study

A data structure that communicates data for a particular study. The fields in the data structure are:

- StudyUID : UID - The UID of the study. For DICOM SOP Instances this is the value of the Study Instance UID (0020,000D) Attribute.
- ObjectDescriptors : ObjectDescriptor[] - An array of ObjectDescriptor data structures listing data that applies to this study (within the enclosing patient), but that do not apply to any particular series within this study.
- Series : Series[] - An array of Series data structures.

9.2.4 Series

A data structure that communicates data for a particular series. The fields in the data structure are:

- SeriesUID : UID - The UID of the series. For DICOM SOP Instances this is the value of the Series Instance UID (0020,000E) Attribute.
- ObjectDescriptors : ObjectDescriptor - An array of ObjectDescriptor data structures listing data existing in this series (within the enclosing Study, within the enclosing Patient).

  Note

  Most DICOM Composite SOP Instances would be identified by objectDescriptors at the Series level.

9.3 MimeType

A data type whose values are Defined Terms that identify particular MIME content types. The syntax of the string defining a MIME content type is defined in [RFC2045]. Top level MIME content types are defined in [RFC2046]. MIME content types are drawn from the list managed by the Internet Assigned Numbers Authority (IANA) whose web site is at http://www.iana.org/assignments/media-types/, as described in [RFC2048].

9.4 ModelSetDescriptor

A data structure returned from the getAsModels() method with the following fields:

- InfosetType : MimeType - the MIME type of the infoset, selected by the source of the data from the list passed to it by the recipient in a getAsModels() call.
- Models : UUID[] - an array of UUIDs referring to models that have been created from the set of data objects referred to by the array of UUIDs passed into the getAsModels() call.
- FailedSourceObjects : UUID[] - an array of UUIDs designating data objects referred to the array of UUIDs passed into the getAsModels() call that could not be represented in the requested model class.

  Note

  For example, if the array of UUIDs passed into the getAsModels() call includes 100 CT slices from the same frame of reference (i.e., a volume stack), plus 1 GSPS object, and if the caller requested an Abstract Multi-Dimensional Image model, then the ModelSetDescriptor returned by GetAsModels() would include one UUID in the models array, identifying the CT volume image data created from the 100 CT slices, and one UUID in the failedSourceObjects array, specifying the UUID for the GSPS object.

9.5 ObjectLocator

A data structure that represents the location from which the recipient of a data object can retrieve that object. It consists of the following fields:

- Locator : UUID - the UUID that the interface utilizes to track this particular ObjectLocator.
- Source : UUID - the UUID of the source that is supplying data for this ObjectLocator. This UUID matches the UUID in the Object-Descriptor if trying to retrieve the data in its natural form (e.g., as a file or byte stream). This UUID matches the UUID in a bulk data pointer when retrieving bulk data from a model.
- TransferSyntax : UID - the transfer syntax in which this data is encoded, selected by source of the data from the list passed in by the recipient of the data in the acceptableTransferSyntaxUIDs parameter of the getData() call. This may be empty for those objects of a MIME content type where Transfer Syntax has no meaning.
- Length: long - the length of the data object referred to by the UUID.
- Offset: long - the offset within the file or byte stream where the data object begins.
- URI: URI - the URI that identifies the resource from which the recipient might retrieve the data object, typically but not limited to a file on the local file system. The recipient shall be able to access the data within the object using file IO or memory mapping.

9.6 QueryResult

A data structure that holds the results from an XPath query of a model. It consists of the following fields:

- Model : UUID - the UUID of the model from which this result came.
- XPath : String - the XPath query string that led to this result.
- Results : XPathNode[] - an array of XPathNodes holding the query results.

9.7 QueryResultInfoset

A data structure that holds the results from an XPath query of a model. It consists of the following fields:

- Model : UUID - the UUID of the model from which this result came.
- XPath : String - the XPath query string that led to this result.
- Results : XPathNodeInfoset[] - an array of XPathNodeInfoset structures holding the query results.

9.8 Rectangle

A data structure that defines a rectangular region on a display screen. The fields in the data structure are:

- RefPointX : int
- RefPointY : int

that define the location of the top left corner of the region in screen coordinates, and

- Width : int
- Height : int

that specify the extents of the region. Screen coordinates are defined starting from an origin of 0,0 in the upper left corner of the screen, and extend in the positive direction down and to the right.

9.9 State

State is an enumerated data type with the following values:

- IDLE
- INPROGRESS
- COMPLETED
- SUSPENDED
- CANCELED
- EXIT

The interpretation of these enumerated values is defined in section7.2 States.

9.10 Status

A data structure with the following fields:

- StatusType : StatusType - the severity level of this status message.
- CodingSchemeDesignator : String - the Coding Scheme in which the codeValues are defined. The use of codeValue shall be consistent with the use of the DICOM Code Value (0008,0100) Attribute as specified in PS3.3.
- CodeValue : String - the particular code value within the designated Coding Scheme that represents the nature of this status message. The use of message shall be consistent with the use of the DICOM Code Meaning (0008,0104) Attribute as specified in PS3.3.
- CodeMeaning : String - a displayable string for the code value. The use of message shall be consistent with the use of the DICOM Code Meaning (0008,0104) Attribute as specified in PS3.3.
- Any other field from the Coded Terminology macro defined in Section 10.1.

9.10.1 StatusType

An enumerated data type with the following values and definitions:

- INFORMATION - the status is for informational purposes only.
- WARNING - indicates a condition that might impact the speed or quality of the work done by the Hosted Application, but that does not prevent the Hosted Application from completing its task.
- ERROR - indicates a condition that might prevent the Hosted Application from correctly completing its task. The Hosted Application will attempt to continue.
- FATALERROR - indicates a condition that prevents the Hosted Application from completing its task. The Hosted Application will not attempt to continue, and will transition automatically to the CANCELED state.

9.11 UID

A string of period-separated digits representing a Unique Identifier (see PS3.5), formatted as described for the UI VR in PS3.5.

9.12 UUID

A string representing a Universally Unique Identifier as defined in [ITU-T X.667], using the hexadecimal representation form.

9.13 XPathNode

A data structure with the following fields, which represents the output from an XPath query of a model, returned in a string-based representation.

- NodeType : XPathNodeType
- Value : String

9.14 XPathNodeInfoset

A data structure with the following fields, which represents the output from an XPath query of a model returned in a byte array representation.

- NodeType : XPathNodeType
- InfosetValue : byte[]

9.15 XPathNodeType

An enumeration of the types of results that may come back from an XPath query.

Note

This enumeration is compatible with a similar enumeration utilized in the Microsoft .NET framework.

- Root - the result is the top level node of the XML Infoset (i.e., the result is the entire XML Infoset).
- Element - the result is an XML Element within the XML Infoset (i.e., the result is a subset of the XML Infoset).
- Attribute - the result is an XML Attribute of an XML Element within the XML Infoset.
- Text - the result is the textual content of an XML Element within the XML Infoset. Equivalent to the Document Object Model (DOM) Text and CDATA node types. Contains at least one character.
- SignificantWhitespace - the result is the content of an XML Element within the XML Infoset, where the content consists only of significant whitespace (e.g., xml:space was set to preserve). White space code points are SPACE (U0020), TAB (U0009), CARRIAGE RETURN (U000D), or LINE FEED (U000A) of ISO 10646 (Unicode).
- Whitespace - the result is the content of an XML Element within the XML Infoset, where the content consists only of whitespace. White space code points are SPACE (U0020), TAB (U0009), CARRIAGE RETURN (U000D), or LINE FEED (U000A) of ISO 10646 (Unicode).
- Comment - the result is a comment within the XML Infoset.
- Namespace - the result is a namespace directive within the XML Infoset.
- ProcessingInstruction - the result is a processing instruction within the XML Infoset.
- All - the result may contain any of the types defined in XPathNodeType.

10 Data Exchange Model Conventions

Models that can be used by the model-based DataExchange interface methods are defined in Annex A. These models are defined in the form of XML Schemas written in Relax NG Compact form of DSDL as specified by [ISO/IEC 19757-2].

Note

An implementer may translate the Relax NG Compact form to other forms for use within their implementation as long as the exchanged XML Infosets will validate against the schema specified by the Standard. For example, a particular implementation may internally utilize a schema with stronger validation rules (e.g., using Schematron rules as specified in [ISO/IEC 19757-3], or using W3C XML Scheme Definition (XSDL) Language [XSD1] with assertion rules) as long as the XML produced for exchange over the interface can be parsed with the schema specified in the Standard, and that XML from well-formed DICOM objects produced by the schema specified in the Standard can still be utilized by the implementation's internal schema.

Actual instances of the models are XML Infosets. Annex A follows the following conventions in describing models.

Note

1. The models are defined via XML schemas to allow the use of commonly available tools to manipulate and navigate the model. For example, an XPath statement can be used to identify portions of interest within the model such as a particular DICOM Attribute and extract it.

2. Some implementations may parse directly from the incoming object, which may not be in XML form, into an internal representation, such as the domain object model (DOM) without ever having converted the object to XML.

Within each model description is a table of XML Elements and XML Attributes used to describe an XML Infoset of that model. These tables utilize the following conventions:

1. XML Element names (listed in the first column) are in CamelCase, with the first letter capitalized.

2. XML Attribute names (listed in the first column) are in camelCase with the first letter in lower case.

3. XML Element and XML Attribute names with a set of ">" characters in front of them are nested within the first preceding XML Element with one fewer ">" characters in front of its name. A nested XML Attribute is associated with the immediately enclosing XML Element.

4. The entries in the "Optionality" column have the following interpretations:

   - "R" indicates that the XML Element or XML Attribute is required in all models.

   - "C" indicates that the XML Element or XML Attribute is required in all models if the condition stated in the Description is met.

   - "O" indicates that the XML Element or XML Attribute is optional.

   - If the XML Element or XML Attribute is nested inside another XML Element, and that enclosing XML Element is not present (i.e., it is defined with an Optionality of "O" and is not present in the XML Infoset, or it is defined with an Optionality of "RC" and is not included in the XML Infoset because the condition was not met), then the nested XML Element or XML Attribute shall not be present in the XML Infoset irrespective of its optionality.

5. The entries in the "Cardinality" column have the following interpretations:

   - "A" indicates that this is represented as an XML Attribute instead of an XML Element, hence has a cardinality of 1 by definition.

   - "1" indicates that only a single instance of this XML Element is included inside the immediately enclosing XML Element, or at the top level if this XML Element is not nested inside another XML Element.

   - "0-n" indicates that zero to n instances of this XML Element are included inside the immediately enclosing XML Element, or at the top level if this XML Element is not nested inside another XML Element.

   - "1-n" indicates that one to n instances of this XML Element are included inside the immediately enclosing XML Element, or at the top level if this XML Element is not nested inside another XML Element.

6. Sets of XML Elements and XML Attributes that are often repeated within models may be defined as macros. Macros that may have general applicability are defined in this section. Macros that are unique to a particular model may be defined in the Annex specific that model. When a macro is included within a table, it is as if the contents of the Macro's table were inserted within the table referencing the macro. Any set of ">" characters in front of the directive to include a Macro in the table are prepended to the XML Element and XML Attribute names listed in the Macro's table.

10.1 Coded Terminology

Models may make use of coded terminology. The representation of coded terminology in DICOM is described in PS3.3. Specific terminology of interest, organized in Context Groups in PS3.16, can be referenced using the following macro.

Table 10.1-1a. Basic Coded Terminology Macro

Name	Optionality	Cardinality	Description
CodeValue	R	1	The particular code value identifying the referenced term or concept. See Section 8.1 in PS3.3. The same XML Element CodeValue is used regardless of the length or encoding of the code value. I.e., separate XML elements are not used when the value is obtained from Long Code Value (0008,0119) or URN Code Value (0008,0120) rather than Code Value (0008,0100).
CodingSchemeDesignator	R	1	Designates the Coding Scheme in which the CodeValue is defined. See Section 8.2 in PS3.3.
CodingSchemeVersion	C	1	See Section 8.2 in PS3.3. Required if the CodingSchemeDesignator is not sufficient to identify the CodeValue unambiguously. May be present otherwise.
CodeMeaning	O	0-1	A brief human readable description of what the coded value means. See Section 8.3 in PS3.3.

Table 10.1-1b. Enhanced Coded Terminology Macro

Name	Optionality	Cardinality	Description
ContextIdentifier	O	0-1	Identifies a Context Group defined within a Mapping Resource from which the values of CodeValue and CodeMeaning were selected or have been added as a Private Context Group extension. See Section 8.6 in PS3.3 and Section 8.7 in PS3.3.
ContextUID	O	0-1	Uniquely identifies the Context Group. See Section 8.6 in PS3.3.
MappingResource	C	1	See Section 8.4 in PS3.3. Required if ContextIdentifier is present.
MappingResourceUID	O	0-1	Uniquely identifies the Mapping Resource that defines the Context Group.
MappingResourceName	O	0-1	Name of Mapping Resource like name of an Institution or Organization.
ContextGroupVersion	C	1	See Section 8.5 in PS3.3. Required if ContextIdentifier is present.
ContextGroupExtensionFlag	O	0-1	Indicates whether the Coded Term is selected from a private extension of the Context Group identified in the ContextIdentifier. See Section 8.7 in PS3.3. Enumerated Values: "Y" "N"
ContextGroupLocalVersion	C	1	See Section 8.7 in PS3.3. Required if the value of ContextGroupExtensionFlag is "Y".

Name	Optionality	Cardinality	Description
ContextGroupExtensionCreatorUID	C	1	Identifies the person or organization who created an extension to the Context Group. See Section 8.7 in PS3.3. Required if the value of contextGroupExtensionFlag is "Y".

Table 10.1-1. Coded Terminology Macro

Name	Optionality	Cardinality	Description
BASIC CODED ENTRY ATTRIBUTES			
Include Table 10.1-1a			
DicomAttribute that encodes EquivalentCodeSequence	O	0-1 with 1-n Item	Codes that are considered equivalent by the creating system. See Section 8.9 in PS3.3.
>Include Table 10.1-1a			
>Include Table 10.1-1b			
ENHANCED ENCODING MODE			
Include Table 10.1-1b			

10.2 Person Name Components

The Person Name Components follow the definitions given in PS3.5 of the DICOM Standard. The PS3.5 description of the usage of Person Name Components also applies to this macro.

Table 10.2-1. Person Name Components Macro

Name	Optionality	Cardinality	Description
FamilyName	O	0-1	The person's family or last name. See the description of the PN VR in DICOM PS3.5.
GivenName	O	0-1	The person's given or first names. See the description of the PN VR in DICOM PS3.5.
MiddleName	O	0-1	The person's middle names. See the description of the PN VR in DICOM PS3.5.
NamePrefix	O	0-1	The person's name prefix. See the description of the PN VR in DICOM PS3.5.
NameSuffix	O	0-1	The person's name suffix. See the description of the PN VR in DICOM PS3.5.

A Data Exchange Models

A.1 Native DICOM Model

A.1.1 Usage

The Native DICOM Model defines a representation of binary-encoded DICOM SOP Instances as XML Infosets that allows a recipient of data to navigate through a binary DICOM data set using XML-based tools instead of relying on tool kits that understand the binary encoding of DICOM.

Note

1. It is not the intention that this form be utilized as the basis for other uses. This form does not take advantage of the self-validation features that could be possible with a pure XML representation of the data.

2. As per the XML standard, XML tags are case sensitive. The case convention for elements is an upper case initial letter, camel case. The case convention for attributes is a lower initial letter, camel case. Keywords referenced in the XML schema are the DICOM title case from the definitions in PS3.6.

With the exception of padding to an even byte length, a data source that is creating a new instance of a Native DICOM Model (e.g., the result from some analysis application) shall follow the DICOM encoding rules (e.g., the handling of character sets) in creating Values for the DicomAttributes within the instance of the Native DICOM Model. Attribute Values encoded in a Native DICOM Model are not required to be padded to an even byte length.

Note

1. Attribute objects within a DICOM JSON Model object are ordered by their property name in ascending order (see Section F.2.2 “DICOM JSON Model Object Structure” in PS3.18). Elements within an XML Infoset following the Native DICOM Model definition are not required to be ordered.

2. The XML is not required to be in a Canonical representation (http://www.w3.org/TR/xml-c14n/).

3. The DICOM Attribute Specific Character Set (0008,0005), if present, reflects the character encoding of the original DICOM dataset and does not indicate the character encoding of the XML Infoset, which is determined by the Infoset encoding declaration. This DICOM Attribute is preserved as metadata and may be used to reconstruct the source binary DICOM encoding. See Section C.12.1.1.2 in PS3.3 for further details.

4. When translating BulkData with character set dependent VRs (SH, LO, ST, PN, LT, UC or UT) to XML representation, the encoding follows the declared Infoset character encoding. For InlineBinary and BulkData with VR of UN, the source byte encoding is preserved without character set translation.

5. If InlineBinary or BulkData with VR of UN is interpreted as containing character string content, the Specific Character Set (0008,0005) of the source DICOM dataset is used to determine the character encoding.

Group Length (gggg,0000) attributes shall not be included in a Native DICOM Model instance.

A data recipient that converts data from an instance of the Native DICOM Model back into a binary encoded DICOM object shall adjust the padding to an even byte length as necessary to meet the encoding rules specified in DICOM PS3.5.

A.1.2 Identification

The ObjectDescriptors MIME content type for the Native DICOM Model shall be "application/x-dicom.native".

The ObjectDescriptors class UID for the Native DICOM Model shall be "1.2.840.10008.7.1.1".

A.1.3 Support

Support of the Native DICOM Model as both a data source and a data recipient shall be required of all Hosting Systems implementing the interface.

Support of the Native DICOM Model as either a data source or a data recipient shall be optional for all Hosted Applications implementing the interface.

A.1.4 Information Model

A diagram of the Native DICOM Model appears in Figure A.1.4-1.

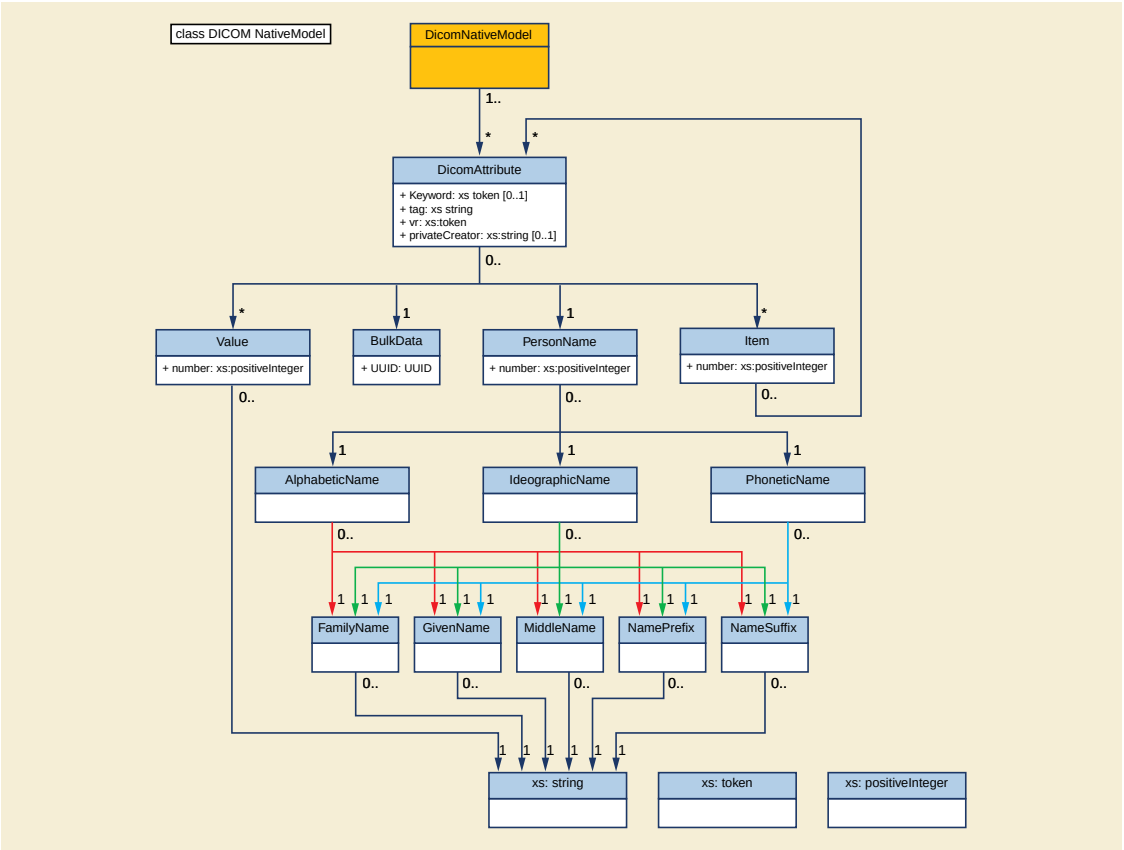


Figure A.1.4-1. Native DICOM Model

A.1.5 Description

Table A.1.5-1. Native DICOM Model

Name	Optionality	Cardinality	Description
NativeDicomModel	R	1	An Infoset (as defined in W3C Recommendation XML Information Set "http://www.w3.org/TR/xml-infoset/") representing the content of a DICOM Data Set (as defined in PS3.5). The directivexml:space="preserve" shall be included. Examples include: • the contents of an entire DICOM Composite Instance (as defined in PS3.3) in response to a native model request, or • the contents of part of a DICOM Composite Instance in response to a query on a native model, or • the contents of a PS3.18 Studies Service Store (STOW-RS) response • the contents of a Sequence Item (as defined in PS3.5), recursively included within an Infoset Value element.
Include Table A.1.5-2 "DICOM Data Set Macro"			

Table A.1.5-2. DICOM Data Set Macro

Name	Optionality	Cardinality	Description
DicomAttribute	O	0-n	An Infoset element corresponding to each DICOM Attribute.
>keyword	C	A	The keyword as defined in PS3.6. Required unless the DICOM Data Element is unknown to the host.
>tag	R	A	The four-digit zero-padded hexadecimal values of the Group and Element Numbers of the Data Element Tag, concatenated as a single string without a delimiter and with lowercase letters disallowed. E.g., Data Element (0010,0020) would have a tag of "00100020". For Private Data Elements, the two most significant hexadecimal characters of the Element Number shall be 00, since the Private Creator is explicitly conveyed and the block used in the DICOM encoding shall not be sent (i.e., a Private Data Element has the form gggg00ee).
>vr	O	A	The Value Representation of this element, represented as a two character uppercase string, as defined in PS3.5 and specified for this Data Element in PS3.6. Note Implementations may utilize the Value Representation to validate data values, if desired.
>privateCreator	C	A	The value of the Private Creator Data Element corresponding to the block in which this Private Data Element is used. Required for Private Data Elements. Shall not be present otherwise (i.e., for Data Elements defined by the DICOM Standard).

Name	Optionality	Cardinality	Description
>Value	C	1-n	A Value from the Value Field of the DICOM Data Element. There is one Infoset Value element for each DICOM Value or Sequence Item. Required if the DICOM Data Element represented is not zero length and an Item, PersonName, InlineBinary or BulkData XML element is not present. Shall not be used if the VR of the enclosing Attribute is either SQ or PN.
>>number	R	A	The order in which the Value occurs within the DICOM Value Field, as a number monotonically increasing starting from 1 by 1. Note The Number XML Attribute is used to preserve the original order.

Name	Optionality	Cardinality	Description

>> *plain character data*	C	1	A single DICOM value encoded as plain character data.
			
			E.g., a DICOM Decimal String Value Field that contained two delimiter-separated values, e.g., "0.5\0.4" would be encoded as two Infoset Value elements:
			
			`<Value number="1">0.5</Value>`
			`<Value number="2">0.4</Value>`
			
			A Code String Value Field that containing three delimiter-separated values, the second of which was zero length, "MPG\\XR3", would be encoded as:
			
			`<Value number="1">MPG</Value>`
			`<Value number="2"></Value>`
			`<Value number="3"><XR3</Value>`
			
			Contrast the latter example with a zero length Value Field, in which case there would be no Infoset Value elements at all.
			
			For DICOM Data Elements whose VR is AT, each value shall be encoded as the four-digit zero-padded hexadecimal values of the Group and Element Numbers of the Data Element Tag, concatenated as a single string without a delimiter and with lowercase letters disallowed.
			
			The character encoding is that declared for the Infoset, regardless of any DICOM Specific Character Set, and any necessary translation from the DICOM Specific Character Set to the Infoset character encoding shall have been performed.
			
			Note
			
			1. This translation might not be completely lossless, particularly with Asian character sets.
			
			2. For those VRs specified in PS3.5 as being affected by Specific Character Set (0008,0005) and permitting multiple delimiter-separated string Values, i.e., SH, LO, PN, and UC, the bit combination 05/12 (0x5C) is used to separate Values into the “Value” array, regardless of whether the Graphic Character represented by 05/12 is the BACKSLASH (such as in the Default Character Set) or the YEN SIGN (such as in [JIS X 0201]). See Section 6.1.2.3 “Encoding of Character Repertoires” in PS3.5. The expected behavior on transcoding from JSON to other representations and vice versa is to preserve the semantics of multiple separate values, whether they are encoded as a multi-valued array in JSON or as a delimiter-separated character stream in PS3.5 representations.
			
			3. Graphic Characters that match the delimiter specified for the Character Set for multi-valued VRs cannot be represented as Values in some Character Sets when encoded in PS3.5 representations. Accordingly, it is inadvisable to use a Graphic Character within a JSON value for the SH, LO, PN, and UC VRs such as a BACKSLASH or YEN SIGN that will be represented as 05/12 in the Default Character Set or [JIS X 0201], because when transcoding to a PS3.5 representation, since there is no escape mechanism defined, they may inadvertently be interpreted as delimiters, splitting a Value. For example, DOS-style paths that use a BACKSLASH as a file component separator cannot be encoded within in a single Value of one of the affected VRs.

>Item	C	1-n	A DICOM sequence item, in other words a nested DICOM Data Set. Required if the DICOM Data Element represented is a Sequence (has a VR of "SQ") and is not zero length. Not allowed otherwise.

Name	Optionality	Cardinality	Description
>>number	R	A	The order in which the Item occurs within a Sequence of Items, as a number monotonically increasing from 1 by 1. Note The Number XML Attribute is used to preserve the original order.
>>*Include Table A.1.5-2 "DICOM Data Set Macro"*	R	1	Recursively includes the Data Set corresponding to a Sequence Item.
>PersonName	C	1-n	A parsed representation in XML of a DICOM Data Element containing a name (i.e., whose VR is PN). Note Parsing Attributes with a VR of PN into an XML representation of the name groups and components simplifies the creation of XPath statements to pull only portions of names out of the DICOM data. Required if the DICOM Data Element represented has a VR of PN and is not zero length. Not allowed otherwise. The rules defined in DICOM PS3.5 on the usage of the Alphabetic, Ideographic, and Phonetic groups of name components within a DICOM Attribute with a Value Representation of PN apply.
>>number	R	A	The order in which the PersonName occurs within the DICOM Value Field, as a number monotonically increasing from 1 by 1. Note The Number XML Attribute is used to preserve the original order.
>>Alphabetic	O	0-1	A group of name components that are represented in alphabetical characters (see the definition for the Value Representation of PN in PS3.5).
>>>*Include Table 10.2-1 "Person Name Components Macro"*			
>>Ideographic	O	0-1	A group of name components that are represented in ideographic characters (see the definition for the Value Representation of PN in PS3.5).
>>>*Include Table 10.2-1 "Person Name Components Macro"*			
>>Phonetic	O	0-1	A group of name components that are represented in phonetic characters (see the definition for the Value Representation of PN in PS3.5).
>>>*Include Table 10.2-1 "Person Name Components Macro"*			

Name	Optionality	Cardinality	Description
>BulkData	C	1	A reference to a blob of data that the recipient may retrieve through use of the GetData() method, a PS3.18 Studies Service Retrieve (WADO-RS) transaction or a PS3.18 Studies Service Store (STOW-RS) transaction. Required if the DICOM Data Element represented is not zero length and an XML Infoset Value, Item, InlineBinary or PersonName element is not present. Shall not be used for a Data Element with a Value Representation of SQ. The provider of the data may use a BulkData reference at its discretion to avoid encoding a large DICOM Value Field as text by value in the Infoset. For example, pixel data or look up tables. There is a single BulkData Infoset element representing the entire Value Field, and not one per Value in the case where the Value Multiplicity is greater than one. Note E.g., a LUT with 4096 16 bit entries that may be encoded in DICOM with a Value Representation of OW, with a VL of 8192 and a VM of 1, or a US VR with a VL of 8192 and a VM of 4096 would both be represented as a single BulkData element. All rules (e.g., byte ordering and swapping) in PS3.5 apply. Note 1. Implementers should pay particular attention to the PS3.5 rules regarding the Value Representations of OD, OF, OL, OV and OW. 2. If the BulkData has a string or text Value Representation, the value(s) of the DICOM Specific Character Set Data Element, if present, might be used by the recipient to determine the encoding of the retrieved bulk data from the source DICOM dataset.
>>uuid	C	A	An identifier of this bulk data reference formatted as a UUID using the hexadecimal representation defined in [ITU-T X.667]. Required if BulkData URI is not present. Shall not be present otherwise.
>>uri	C	A	The HTTP(S) URI for this bulk data reference. Required if the NativeDicomModel was: • returned in response to a PS3.18 Studies Service Retrieve (WADO-RS) Retrieve Metadata request Shall not be present otherwise.

Name	Optionality	Cardinality	Description
>InlineBinary	C	1	The Value Field of the enclosing Attribute encoded as base64. Required if the DICOM Data Element represented is: • not zero length • the VR if the enclosing Attribute is OB, OD, OF, OL, OV, OW, or UN • an XML Infoset Value or BulkData XML element is not present Shall not be present otherwise. There is a single InlineBinary Infoset element representing the entire Value Field, and not one per Value in the case where the Value Multiplicity is greater than one. Note E.g., a LUT with 4096 16 bit entries that may be encoded in DICOM with a Value Representation of OW with a VL of 8192 and a VM of 1 would be represented as a single InlineBinary element. All rules (e.g., byte ordering and swapping) in PS3.5 apply. Note Implementers should pay particular attention to the PS3.5 rules regarding the value representations of OD, OF, OL, OV and OW.

A.1.6 Schema

The Normative version of the Relax NG Compact Schema for the Native DICOM Model follows:

```
default namespace="http://dicom.nema.org/PS3.19/models/NativeDICOM"

# This schema was created as an intermediary, a means of describing
# native binary encoded DICOM objects as XML Infosets, thus allowing
# one to manipulate binary DICOM objects using familiar XML tools.
# As such, the schema is designed to facilitate a simple, mechanical,
# bi-directional translation between binary encoded DICOM and XML-like
# constructs without constraints, and to simplify identifying portions
# of a DICOM object using XPath statements.
#
# Since this schema has minimal type checking, it is neither intended
# to be used for any operation that involves hand coding, nor to
# describe a definitive, fully validating encoding of DICOM concepts
# into XML, as what one might use, for example, in a robust XML
# database system or in XML-based forms, though it may be used
# as a means for translating binary DICOM Objects into such a form
# (e.g., through an XSLT script).

start = element NativeDicomModel { DicomDataSet }

# A DICOM Data Set is as defined in PS3.5.  It does not appear
# as an XML Element, since it does not appear in the binary encoded
# DICOM objects.  It exists here merely as a documentation aid.
DicomDataSet = DicomAttribute*

DicomAttribute = element DicomAttribute {
  Tag, VR, Keyword?, PrivateCreator?,
  (BulkData | Value+ | Item+ | PersonName+ | InlineBinary)?
```

```
}
BulkData = element BulkData{ UUID | URI }
Value = element Value { Number, xsd:string }
InlineBinary = element InlineBinary { xsd:base64Binary }
Item = element Item { Number, DicomDataSet }
PersonName = element PersonName {
  Number,
  element Alphabetic  { NameComponents }?,
  element Ideographic { NameComponents }?,
  element Phonetic    { NameComponents }?
}

NameComponents =
  element FamilyName {xsd:string}?,
  element GivenName  {xsd:string}?,
  element MiddleName {xsd:string}?,
  element NamePrefix {xsd:string}?,
  element NameSuffix {xsd:string}?

# keyword is the attribute tag from PS3.6
# (derived from the DICOM Attribute's name)
Keyword = attribute keyword { xsd:token }
# canonical XML definition of Hex, with lowercase letters disallowed
Tag = attribute tag { xsd:string{ minLength="8" maxLength="8" pattern="[0-9A-F]{8}" } }
VR = attribute vr { "AE" | "AS" | "AT"| "CS" | "DA" | "DS" | "DT" | "FL" | "FD"
                  | "IS" | "LO" | "LT" | "OB" | "OD" | "OF" | "OL" | "OV" | "OW" | "PN" | "SH" | "SL"
                  | "SQ" | "SS" | "ST" | "SV" | "TM" | "UC" | "UI" | "UL" | "UN" | "UR" | "US" | "UT" | "UV" }
PrivateCreator = attribute privateCreator{ xsd:string }
UUID = attribute uuid { xsd:string }
URI = attribute uri { xsd:anyURI }
Number = attribute number { xsd:positiveInteger }
```

A.1.7 Examples

Here is an example XPath query to extract the code meaning of the first item in the View Code Sequence:

/NativeDicomModel/DicomAttribute[@keyword="ViewCodeSequence"]/Item[@number=1]/DicomAttribute[@keyword="CodeMeaning"]/Value[@number=1]

A.2 Abstract Multi-Dimensional Image Model

A.2.1 Usage

The Abstract Multi-Dimensional Image Model can be used to refer to a discretely-sampled, multi-dimensional image data. The sample values may either be single-valued (a scalar) or a vector of values (a vector). An example would be a time varying series of three dimensional images set at multiple energy levels. The Abstract Multi-Dimensional Image Model is patterned after the Enhanced Multi-frame family of DICOM objects. In mathematical terms, this is any data set that is defined by a function *I (x,y,z,t,…)*, where *(x,y,z,t,…)* are the dimensions, and the sample value of *I* is either a vector of components or a scalar (i.e., a single component). The primary purpose of this model is to allow applications to process image data without concern as to the underlying format of the data.

When converting DICOM SOP Instances into Abstract Multi-Dimensional Image Models, a provider of data shall follow these rules as closely as it practically can:

> Note
>
> Deterministic behavior is not expected nor guaranteed when making conversions between DICOM SOP Instances and Abstract Multi-Dimensional Image Models. For example, given the same DICOM SOP Instances, different Hosting Systems may create Abstract Multi-Dimensional Image Models that differ in some details, such as the Units of the Component values or in the Dimensions.

1. Multiple DICOM SOP Instances from the same series that have the same Frame of Reference UID shall be combined into a single instance of the Abstract Multi-Dimensional Image Model. DICOM SOP Instances from multiple series that have the same Frame of Reference UUID may be combined into a single instance of the Abstract Multi-Dimensional Image Model, if appropriate.

2. A single DICOM SOP Instance shall not be divided into multiple instances of the Abstract Multi-Dimensional Image Model.

3. The coordinate system utilized within the Abstract Multi-Dimensional Image Model shall use the coordinate system defined by the DICOM objects utilized in the creation of the Abstract Multi-Dimensional Image Model instance if applicable. Where practical, the coordinate system and Dimension definitions utilized within the Abstract Multi-Dimensional Image Model shall be chosen such that interpolation is not required to convert the source data into the Abstract Multi-Dimensional Image Model.

   Note

   Interpolation may be necessary if the source data is not laid out on a frame-based Cartesian coordinate grid.

4. Spatial coordinates, such as Image Position (Patient) (0020,0032), shall be transformed into the coordinate system utilized within the Abstract Multi-Dimensional Image Model instance.

5. The Pixel Data shall be spatially transformed as needed to match the Semantics and Units of the Dimensions of the Abstract Multi-Dimensional Image Model into which the pixels values are being placed.

6. Any embedded overlays within the Pixel Data (7FE0,0010) Attribute shall be stripped out of the pixel values and the pixel values sign extended or converted as needed to match the datatype of the Component of the Abstract Multi-Dimensional Image Model into which the pixels values are being placed.

7. The pixel values of the Pixel Data shall be transformed as needed to match the Semantics and Units of the Component of the Abstract Multi-Dimensional Image Model into which the pixels values are being placed.

   Note

   Typically presentation settings such as VOI and Presentation LUTs are not used in creating Abstract Multi-Dimensional Image Models from DICOM SOP Instances. The exception is when the application of such LUTs is needed to match the Semantics and Units of the Component. Modality LUTs or Rescale Slope and Intercept often must be applied to match the Semantics and Units of the Abstract Multi-Dimensional Image Model.

8. Any pixel values that correspond to the pixel padding values shall be stripped out (i.e., set to zero or other suitable replacement value) and the spatially corresponding values in the PixelMapOfValidData shall be set to the outValue or something other than the inValue, as appropriate.

When converting data within an instance of the Abstract Multi-Dimensional Image Models into DICOM SOP Instances, the recipient of an abstract model instance shall convert the pixel data back into values compatible with the native form of the DICOM SOP Instances being created. This conversion may include recreating Modality LUT information, inserting pixel padding values, converting pixel spacing and origins, etc. as dictated by the SOP Class the data is being converted to. When converting a single Abstract Multi-Dimensional Image Model into multiple DICOM SOP Instances, the DICOM SOP Instances shall all have the same Frame of Reference UID (0020,0052), if permitted by the SOP Class.

A.2.2 Identification

The ObjectDescriptors MIME content type for the Abstract Multi-Dimensional Image Model is "application/x-dicom.abstract".

Note

This is an experimental MIME type. A formal MIME type will be applied for. Implementations will be expected to support both the experimental and formal MIME type going forward without a version change to the interface.

The ObjectDescriptors class UID for the Abstract Multi-Dimensional Image Model is "1.2.840.10008.7.1.2".

A.2.3 Support

Support of the Abstract Multi-Dimensional Image Model as both a data source and a data recipient is required of all Hosting Systems implementing the interface.

Support of the Abstract Multi-Dimensional Image Model as either a data source or a data recipient is optional for all Hosted Applications implementing the interface.

A.2.4 Information Model

A diagram of the Abstract Multi-Dimensional Image Model appears in Figure A.2.4-1.

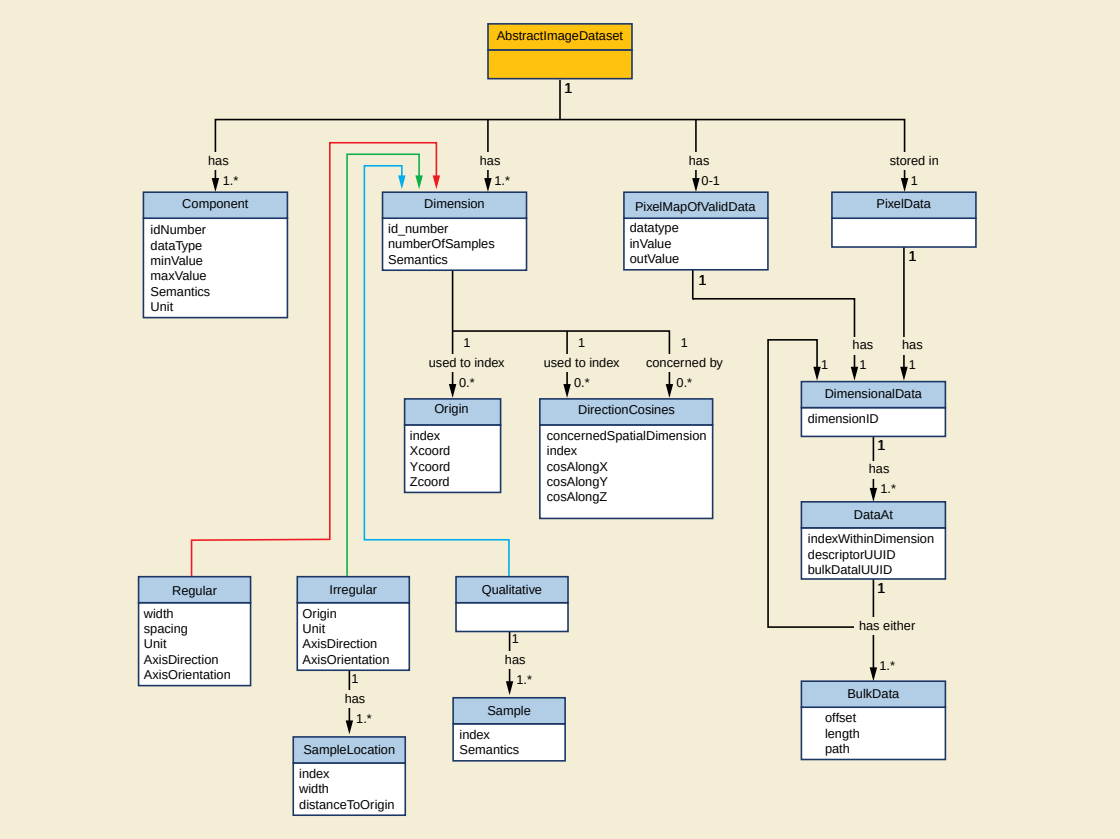


Figure A.2.4-1. Abstract Multi-Dimensional Image Model

A.2.5 Description

Table A.2.5-1. Abstract Image Model

Name	Optionality	Cardinality	Description
AbstractImageDataSet	R	1	The top level element required of all abstract image models, holding the entire abstract image Data Set.
>Component	R	1-n	Describes a component of the function output. If the output is a scalar, there is only one Component. Vector outputs require a Component for each position in the vector. When there are multiple components, the components appear in each value in the order defined by their respective idNumbers.
>>idNumber	R	A	Identifies this particular component, with numbering monotonically increasing from 1.

Name	Optionality	Cardinality	Description
>>datatype	R	A	Describes how this component value is represented. Enumerated values are: SIGNED_INT8 SIGNED_INT16 SIGNED_INT32 UNSIGNED_INT8 UNSIGNED_INT16 UNSIGNED_INT32 FLOAT32 FLOAT64
>>minValue	O	A	The minimum value that this component takes on. If this XML Attribute is missing, this is the minimum value that can be represented by the Datatype.
>>maxValue	O	A	The maximum value that this component takes on. If this XML Attribute is missing, this is the maximum value that can be represented by the Datatype.
>>Semantics	R	1	A coded value describing what this component represents.
>>>*Include Table 10.1-1 "Coded Terminology Macro"*			Defined CID 7180 "Abstract Multi-dimensional Image Model Component Semantic".
>>Unit	R	1	A coded value describing what units this dimension is in.
>>>*Include Table 10.1-1 "Coded Terminology Macro"*			Defined CID 7181 "Abstract Multi-dimensional Image Model Component Unit".
>Dimension	R	1-n	Describes a dimension.
>>idNumber	R	A	Identifies this particular dimension, with numbering starting from 1. Dimensions with a lower idNumber vary faster than those with a higher idNumber.
>>numberOfSamples	R	A	The number of samples in this dimension, for example:the number of columns along the X-axis,the number of rows along the Y-axis,the number of slices along the Z-axis,the number of qualitative descriptions.
>>Semantics	R	1	A coded value describing what this dimension represents.
>>>*Include Table 10.1-1 "Coded Terminology Macro"*			Defined CID 7182 "Abstract Multi-dimensional Image Model Dimension Semantic"
>> Regular	C	1	Used to describe regularly spaced samples in this dimension. Required if neither Irregular nor Qualitative are present. Shall not be present otherwise.
>>>width	R	A	The sample width.
>>>spacing	R	A	The sample spacing.
>>>Unit	R	1	A coded value describing what units the sample width and spacing are in.
>>>>*Include Table 10.1-1 "Coded Terminology Macro"*			Defined CID 7183 "Abstract Multi-dimensional Image Model Dimension Unit".

Name	Optionality	Cardinality	Description
>>>AxisDirection	O	1	The direction of the axis of this dimension. Note This XML Element might only be applicable to spatial dimensions, such as those dealing with linear displacement. Typically this is in relationship to the patient.
>>>>*Include Table 10.1-1 "Coded Terminology Macro"*			Defined CID 7184 "Abstract Multi-dimensional Image Model Axis Direction"
>>>AxisOrientation	O	1	The orientation of the axis of this dimension along which values are increasing. Note This XML Element might only be applicable to spatial dimensions, such as those dealing with linear displacement. Typically this is in relationship to the patient.
>>>>*Include Table 10.1-1 "Coded Terminology Macro"*			Defined CID 7185 "Abstract Multi-dimensional Image Model Axis Orientation"
>>Irregular	C	1	Used to describe irregularly spaced samples in this dimension. Required if neither Regular nor Qualitative are present. Shall not be present otherwise.
>>>origin	R	A	The reference location from which each of the sample locations are measured.
>>>SampleLocation	R	1-n	Describes the locations of each sample as an offset from the origin. There shall be numberOfSamples SampleLocation XML Elements in this sequence.
>>>>index	R	A	The index value of this sample location, with numbering starting from 1 and incrementing to numberOfSamples.
>>>>width	R	A	The sample width.
>>>>distanceToOrigin	R	A	The distance of this sample location from the Origin location.
>>>Unit	R	1	A coded value describing what units the sample widths and locations are in.
>>>>*Include Table 10.1-1 "Coded Terminology Macro"*			Defined CID 7183 "Abstract Multi-dimensional Image Model Dimension Unit".
>>>AxisDirection	O	1	The direction of the axis of this dimension. Note This XML Element might only be applicable to spatial dimensions, such as those dealing with linear displacement. Typically this is in relationship to the patient.
>>>>*Include Table 10.1-1 "Coded Terminology Macro"*			Defined CID 7184 "Abstract Multi-dimensional Image Model Axis Direction"
>>>AxisOrientation	O	1	The orientation of the axis of this dimension along which values are increasing. Note This XML Element might only be applicable to spatial dimensions, such as those dealing with linear displacement. Typically this is in relationship to the patient.

Name	Optionality	Cardinality	Description
>>>>*Include Table 10.1-1 "Coded Terminology Macro"*			Defined CID 7185 "Abstract Multi-dimensional Image Model Axis Orientation"
>>Qualitative	C	1	Used to describe a qualitative dimension. Required if neither Regular nor Irregular are present. Shall not be present otherwise.
>>>Sample	R	1-n	Description of what each sample along this dimension represents. There shall be numberOfSamples Sample XML Elements in this sequence.
>>>>index	R	A	The index value of this sample, with numbering starting from 1 and increasing to numberOfSamples.
>>>>Semantics	R	1	A coded value describing what this sample represents.
>>>>>*Include Table 10.1-1 "Coded Terminology Macro"*			Defined CID 7186 "Abstract Multi-dimensional Image Model Qualitative Dimension Sample Semantic"
>>Origin	O	0-n	Specifies the spatial position in the coordinate system of the Abstract Multi-Dimensional Image Model of the spatial frames or volumes of image data values. Different frames or volumes may either share an origin, or have a different origin for each frame or volume. If there is only a single Origin XML element within this Dimension, then this Origin applies to all samples along this Dimension. Otherwise, there shall be numberOfSamples Origin XML elements, one for each sample along this Dimension. Sample index values for Dimensions whose idNumbers are less than this Dimension's idNumber, are all equal to 1.
>>>index	R	A	Index of the sample to which this Origin applies. If this is a single Origin that applies to all samples along this Dimension, then index shall either be left out or given a value of "0" (zero). Otherwise, the value shall be the appropriate number between 1 and numberOfSamples.
>>>xCoord	R	A	The X position of this Origin in the coordinate system of the Abstract Multi-Dimensional Image Model.
>>>yCoord	R	A	The Y position of this Origin in the coordinate system of the Abstract Multi-Dimensional Image Model.
>>>zCoord	R	A	The Z position of this Origin in the coordinate system of the Abstract Multi-Dimensional Image Model.
>>DirectionCosines	O	0-n	Specifies the direction in the coordinate system of the Abstract Multi-Dimensional Image Model of the Dimension whose idNumber is given in concernedSpatialDimension. The idNumber of the concernedSpatialDimension shall be less than the idNumber of this Dimension. If there is only a single DirectionCosines XML element within this Dimension XML element with a particular concernedSpatialDimension, then this Direction Cosine applies to all samples along this Dimension. Otherwise, there shall be numberOfSamples DirectionCosines XML elements with this particular concernedSpatialDimension, one for each sample along this Dimension.
>>>concernedSpatialDimension	R	A	The idNumber of the particular Dimension for which this DirectionCosines XML element applies. The value of concernedSpatialDimension shall be less than the idNumber of this Dimension.
>>>index	C	A	Index of this direction specification, with numbering starting from 1. If this is a single-valued DirectionCosines that applies to all samples along this Dimension then index shall either be left out or given a value of "0" (zero). Otherwise, the value of index refers to the DirectionCosines of a particular sample value along this Dimension.

Name	Optionality	Cardinality	Description
>>>cosAlongX	R	A	The direction cosine along the X axis of the coordinate system of the Abstract Multi-Dimensional Image Model for this concernedSpatialDimension.
>>>cosAlongY	R	A	The direction cosine along the Y axis of the coordinate system of the Abstract Multi-Dimensional Image Model for this concernedSpatialDimension.
>>>cosAlongZ	R	A	The direction cosine along the Z axis of the coordinate system of the Abstract Multi-Dimensional Image Model for this concernedSpatialDimension.
>PixelData	R	1	Structure that defines where the pixel data is located, organized along dimensional lines.
>>*Include Table A.2.5-2 "Dimensional Data Macro"*			
>PixelMapOfValidData	O	0-1	A pixel map that identifies which pixels either belong in or out of the Data Set. The dimensions of the pixel map match the dimensions of the image data, i.e., there is a one-to-one correspondence between samples in the image data and samples in the pixel map. The pointers to the pixel map data are included in one of the Dimension XML elements.
>>datatype	R	A	Describes how samples in the pixel map are encoded. Enumerated values are: BIT1 UNSIGNED_INT8 For BIT1, the bit ordering starts from the least significant bit going to the most significant bit within an UNSIGNED_INT8 (i.e., 8 bit) byte. The bits are zero-padded to make a full 8-bit byte at the end of the most rapidly changing dimension (i.e., the Dimension whose idNumber is 1).
>>inValue	C	A	The value within the pixel map that indicates that this sample shall be considered as part of the Data Set. All samples whose pixel map values do not match inValue shall not be considered as part of the Data Set. Required if outValue is not present. Shall not be present if outValue is present.
>>outValue	C	A	The value within the pixel map that indicates that this sample shall not be considered as part of the Data Set. All samples whose pixel map values do not match outValue shall be considered as part of the Data Set. Required if inValue is not present. Shall not be present if inValue is present.
>>*Include Table A.2.5-2 "Dimensional Data Macro"*			

Table A.2.5-2. Dimensional Data Macro

>dimensionID	R	A	The idNumber of the Dimension in this AbstractImageDataSet to which this DimensionalData refers.

>DataAt	O	1-n	References to where the image data is located. Only one Dimension XML Element within this AbstractImageDataSet shall have UUIDs for bulk pixel data (i.e., all bulk data references are at the same dimensional level). Note If the source of the data, as part of the model preparation, creates a single file for pixel data from multiple smaller native objects, then in order to provide the descriptorUUID XML Attributes the source may need to create multiple bulkDataUUIDs referring to different offsets within that single pixel data file.
>>indexWithinDimension	R	A	The ordinal position (e.g., index number) of this sample point in the array of data at this level. Numbering starts from 1.
>>descriptorUUID	C	A	A UUID that refers to the ObjectDescriptor from which this data is drawn, formatted in the hexadecimal representation defined by [ITU-T X.667]. Required at the level of the nested tree structure where the source added the data from the descriptorUUID into the Abstract Multi-Dimensional Image Model.
>>bulkDataUUID	C	A	The identifier that the recipient of the data may use in a getData() call to gain access to the bulk pixel data formatted as a UUID using the hexadecimal representation defined in [ITU-T X.667]. Required if the Dimensional Data Macro is not present at this level of the nested tree structure. Shall not be present otherwise.
>>*Conditionally include Table A.2.5-2 "Dimensional Data Macro"*			Only one of bulkDataUUID or Dimensional Data shall be included at each level. If Dimensional Data is included, it shall be the next lower level of the nested tree structure, that is the Dimension with an idNumber one less than the Dimension referred to by the enclosing DimensionalData.

A.2.6 Schema

The Relax NG Compact schema for the Abstract Multi-Dimensional Image Model follows:

```
default namespace = "http://dicom.nema.org/PS3.19/models/AbstractImage"

start = AbstractImageDataSet
AbstractImageDataSet =

 element AbstractImageDataSet {
    element Component{
      attribute idNumber { xsd:positiveInteger },
      attribute datatype { ComponentDatatype },
      attribute minValue { xsd:double }?,
      attribute maxValue { xsd:double }?,
      element Semantics { CodedTerm },
      element Unit { CodedTerm }
    }+,
    element Dimension {
      attribute idNumber { xsd:positiveInteger },
      attribute numberOfSamples { xsd:positiveInteger },
      element Semantics { CodedTerm },
      (element Regular {
         attribute width { xsd:double },
         attribute spacing { xsd:double },
         element Unit { CodedTerm },
         element AxisDirection { CodedTerm }?,
         element AxisOrientation { CodedTerm }?
       }
```

```
       | element Irregular {
           attribute origin { xsd:double },
           element SampleLocation {
             attribute index { xsd:positiveInteger },
             attribute width { xsd:double },
             attribute distanceToOrigin { xsd:double }
           }+,
         element Unit { CodedTerm },
         element AxisDirection { CodedTerm }?,
         element AxisOrientation { CodedTerm }?
         }
       | element Qualitative {
           element Sample {
             attribute index { xsd:positiveInteger },
             element Semantics { CodedTerm }
           }+
         }),
       element Origin {
         attribute index { xsd:nonNegativeInteger }?,
         attribute xCoord { xsd:double },
         attribute yCoord { xsd:double },
         attribute zCoord { xsd:double }
       }*,
       element DirectionCosines {
         attribute concernedSpatialDimension { xsd:positiveInteger },
         attribute index { xsd:nonNegativeInteger }?,
         attribute cosAlongX { xsd:double },
         attribute cosAlongY { xsd:double },
         attribute cosAlongZ { xsd:double }
       }*
     }+,
     element PixelData { DimensionalData },
     element PixelMapOfValidData {
       attribute datatype { PixelMapDatatype },
       (
         attribute inValue { xsd:positiveInteger }
         | attribute outValue { xsd:positiveInteger }
       ),
       DimensionalData
     }?
   }

ComponentDatatype =
     "SIGNED_INT8"
     | "SIGNED_INT16"
     | "SIGNED_INT32"
     | "UNSIGNED_INT8"
     | "UNSIGNED_INT16"
     | "UNSIGNED_INT32"
     | "FLOAT32"
     | "FLOAT64"

PixelMapDatatype =
     "BIT1"
     | "UNSIGNED_INT8"

DimensionalData =
   element DimensionalData {
     attribute dimensionID { xsd:positiveInteger },
     element DataAt
```

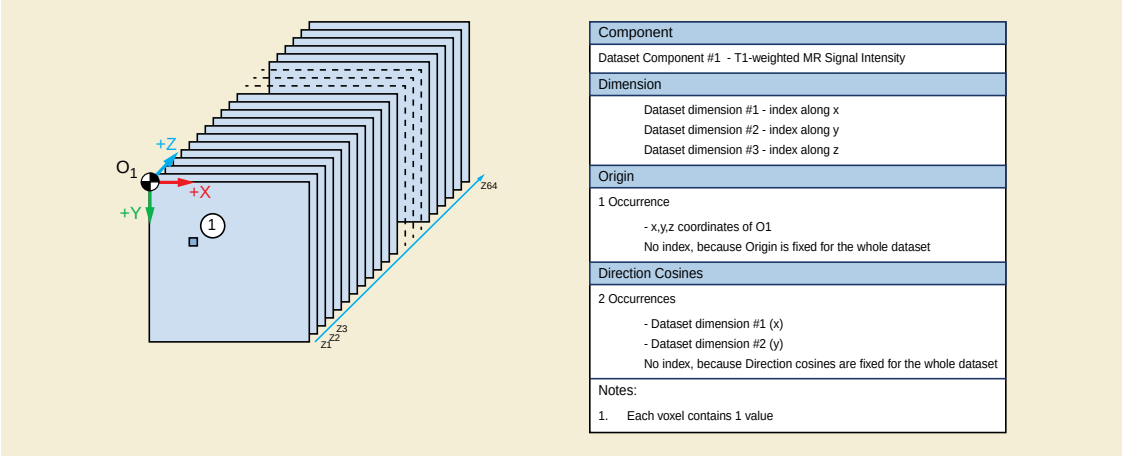
```
    {
      attribute indexWithinDimension { xsd:positiveInteger },
      attribute descriptorUUID { xsd:string }?,
      (DimensionalData | BulkDataPointer)
    }+
  }

BulkDataPointer =
    attribute bulkDataUUID { xsd:string }

CodedTerm =
    element CodeValue { xsd:string },
    element CodingSchemeDesignator { xsd:string },
    element CodingSchemeVersion { xsd:string }?,
    element CodeMeaning { xsd:string }?,
    (
      element ContextIdentifier { xsd:string },
      element ContextUID { xsd:string }?,
      element MappingResource { xsd:string },
      element MappingResourceUID { xsd:string }?,
      element ContextGroupVersion { xsd:string }
    )?,
    (
      element ContextGroupExtensionFlag { xsd:string },
      element ContextGroupLocalVersion { xsd:string }?,
      element ContextGroupExtensionCreatorUID { xsd:string }?
    )?
```

A.2.7 Examples

A.2.7.1 Simple 3D Volume



Component
Dataset Component #1 - T1-weighted MR Signal Intensity
Dimension
Dataset dimension #1 - index along x Dataset dimension #2 - index along y Dataset dimension #3 - index along z
Origin
1 Occurrence - x,y,z coordinates of O1 No index, because Origin is fixed for the whole dataset
Direction Cosines
2 Occurrences - Dataset dimension #1 (x) - Dataset dimension #2 (y) No index, because Direction cosines are fixed for the whole dataset
Notes: 1. Each voxel contains 1 value

Figure A.2.7.1-1. Simple 3D Volume Example

A.2.7.2 Simple 4D Volume

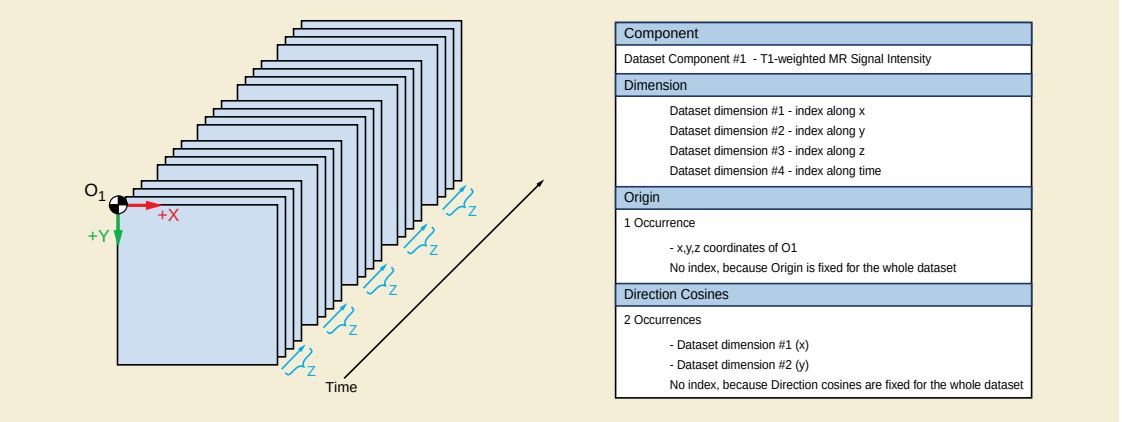


Figure A.2.7.2-1. Simple 4D Volume Example

A.2.7.3 2D Ultrasound

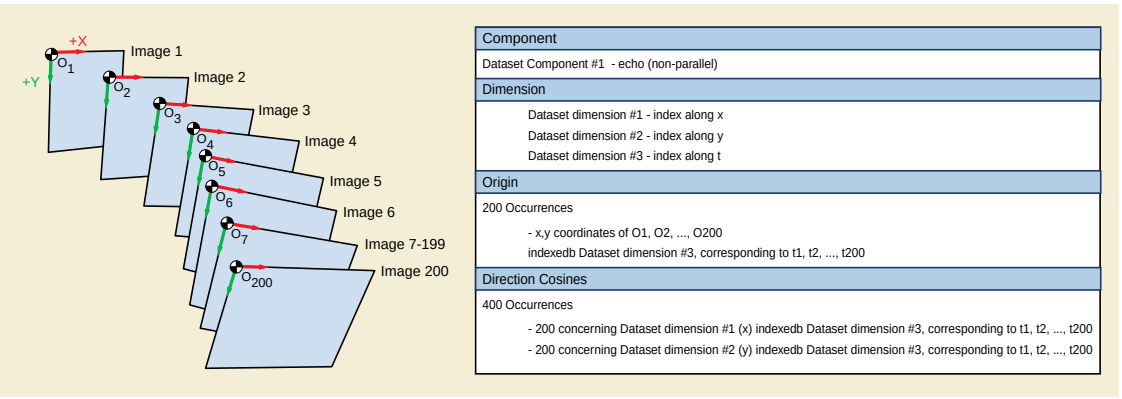
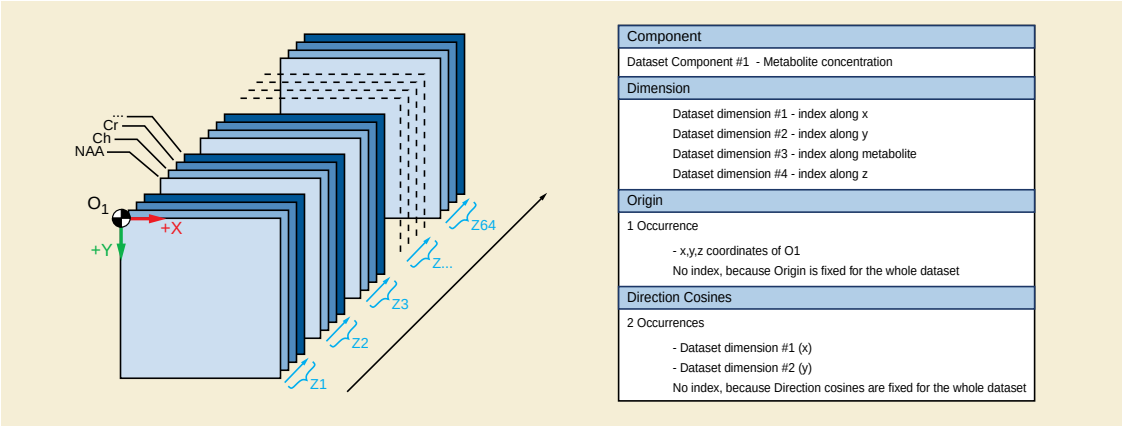


Figure A.2.7.3-1. 2D Ultrasound Example

- In this particular case, we have three dimensions, numbered #1 for displacements along X, #2 for displacements along Y, and #3 to index the time series. If we have 200 images along time (i.e., the *numberOfSamples* XML Attribute is set to 200), we will then have 400 occurrences of the *DirectionCosines* XML Element within the *Dimension* XML Element whose *idNumber* XML Attribute is set to #3 (the dimension referring to time). The 200 first occurrences will have the XML Attribute *concernedSpatialDimension* with value #1 (to specify direction cosines along the X axis) and will be indexed by the XML Attribute *index* varying from 1 to 200 corresponding to the 200 images along time. The 200 following occurrences will have the XML Attribute *concernedSpatialDimension* with value #2 (to specify direction cosines along the Y axis), and will also be indexed by the XML Attribute *index* varying from 1 to 200.

- Similarly, in this example we will have 200 occurrences of the *Origin* XML Element within the Dimension XML Element that has the *idNumber* XML Attribute set to the value 3, and of course by the XML Attribute *index* varying from 1 to 200.

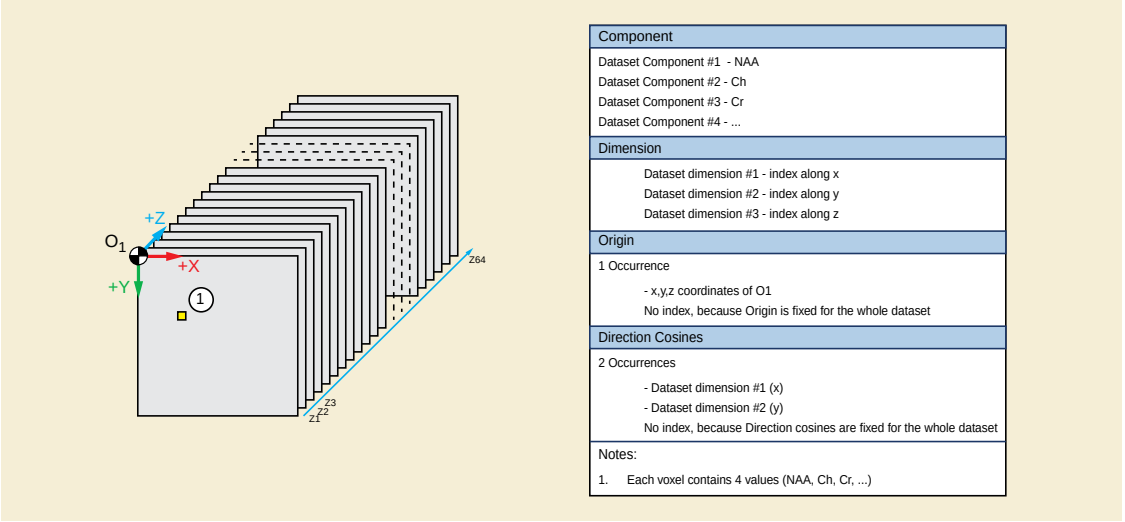
A.2.7.4 3D MR Metabolite Map - Single Component



Component
Dataset Component #1 - Metabolite concentration
Dimension
Dataset dimension #1 - index along x Dataset dimension #2 - index along y Dataset dimension #3 - index along metabolite Dataset dimension #4 - index along z
Origin
1 Occurrence - x,y,z coordinates of O1 No index, because Origin is fixed for the whole dataset
Direction Cosines
2 Occurrences - Dataset dimension #1 (x) - Dataset dimension #2 (y) No index, because Direction cosines are fixed for the whole dataset

Figure A.2.7.4-1. Single Component 3D MR Metabolite Example

A.2.7.5 3D MR Metabolite Map - Multiple Component



Component
Dataset Component #1 - NAA Dataset Component #2 - Ch Dataset Component #3 - Cr Dataset Component #4 - ...
Dimension
Dataset dimension #1 - index along x Dataset dimension #2 - index along y Dataset dimension #3 - index along z
Origin
1 Occurrence - x,y,z coordinates of O1 No index, because Origin is fixed for the whole dataset
Direction Cosines
2 Occurrences - Dataset dimension #1 (x) - Dataset dimension #2 (y) No index, because Direction cosines are fixed for the whole dataset
Notes: 1. Each voxel contains 4 values (NAA, Ch, Cr, ...)

Figure A.2.7.5-1. Multiple Component 3D MR Metabolite Map Example

B Interface Definitions

B.1 Application Interface - Version 20100825

B.1.1 WSDL Definition of the Interface

The following is the content of ApplicationService-20100825.wsdl:

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions name="ApplicationService-20100825"
targetNamespace="http://dicom.nema.org/PS3.19/ApplicationService-20100825"
xmlns:tns="http://dicom.nema.org/PS3.19/ApplicationService-20100825"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:wsa10="http://www.w3.org/2005/08/addressing"
xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <xsd:schema targetNamespace="http://dicom.nema.org/PS3.19/Imports/ApplicationService-20100825">

      <xsd:import namespace="http://dicom.nema.org/PS3.19/ApplicationService-20100825"
      schemaLocation="./ApplicationService-20100825.xsd" />
      <xsd:import namespace="http://schemas.microsoft.com/2003/10/Serialization/"
      schemaLocation="./Types.xsd" />
      <xsd:import namespace="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
      schemaLocation="./ArrayOfString.xsd" />
      <xsd:import namespace="http://schemas.datacontract.org/2004/07/System.Xml.XPath"
      schemaLocation="./XPathNodeType.xsd" />
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="IApplicationService_GetState_InputMessage">
    <wsdl:part name="parameters" element="tns:GetState" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_GetState_OutputMessage">
    <wsdl:part name="parameters" element="tns:GetStateResponse" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_SetState_InputMessage">
    <wsdl:part name="parameters" element="tns:SetState" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_SetState_OutputMessage">
    <wsdl:part name="parameters" element="tns:SetStateResponse" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_BringToFront_InputMessage">
    <wsdl:part name="parameters" element="tns:BringToFront" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_BringToFront_OutputMessage">
    <wsdl:part name="parameters" element="tns:BringToFrontResponse" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_NotifyDataAvailable_InputMessage">
```

```
    <wsdl:part name="parameters" element="tns:NotifyDataAvailable" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_NotifyDataAvailable_OutputMessage">
    <wsdl:part name="parameters" element="tns:NotifyDataAvailableResponse" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_GetData_InputMessage">
    <wsdl:part name="parameters" element="tns:GetData" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_GetData_OutputMessage">
    <wsdl:part name="parameters" element="tns:GetDataResponse" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_ReleaseData_InputMessage">
    <wsdl:part name="parameters" element="tns:ReleaseData" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_ReleaseData_OutputMessage">
    <wsdl:part name="parameters" element="tns:ReleaseDataResponse" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_GetAsModels_InputMessage">
    <wsdl:part name="parameters" element="tns:GetAsModels" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_GetAsModels_OutputMessage">
    <wsdl:part name="parameters" element="tns:GetAsModelsResponse" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_ReleaseModels_InputMessage">
    <wsdl:part name="parameters" element="tns:ReleaseModels" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_ReleaseModels_OutputMessage">
    <wsdl:part name="parameters" element="tns:ReleaseModelsResponse" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_QueryModel_InputMessage">
    <wsdl:part name="parameters" element="tns:QueryModel" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_QueryModel_OutputMessage">
    <wsdl:part name="parameters" element="tns:QueryModelResponse" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_QueryInfoSet_InputMessage">
    <wsdl:part name="parameters" element="tns:QueryInfoSet" />
  </wsdl:message>
  <wsdl:message name="IApplicationService_QueryInfoSet_OutputMessage">
    <wsdl:part name="parameters" element="tns:QueryInfoSetResponse" />
  </wsdl:message>
  <wsdl:portType name="IApplicationService-20100825">
    <wsdl:operation name="GetState">
      <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/GetState"
      message="tns:IApplicationService_GetState_InputMessage" />
      <wsdl:output
        wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/GetStateResponse"
      message="tns:IApplicationService_GetState_OutputMessage" />
    </wsdl:operation>
    <wsdl:operation name="SetState">
      <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/SetState"
      message="tns:IApplicationService_SetState_InputMessage" />
      <wsdl:output
        wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/SetStateResponse"
      message="tns:IApplicationService_SetState_OutputMessage" />
    </wsdl:operation>
    <wsdl:operation name="BringToFront">
      <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/BringToFront"
      message="tns:IApplicationService_BringToFront_InputMessage" />
      <wsdl:output
```

```
      wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/BringToFrontResponse"
    message="tns:IApplicationService_BringToFront_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="NotifyDataAvailable">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/NotifyDataAvailable"
    message="tns:IApplicationService_NotifyDataAvailable_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/NotifyDataAvailableResponse"
    message="tns:IApplicationService_NotifyDataAvailable_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="GetData">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/GetData"
    message="tns:IApplicationService_GetData_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/GetDataResponse"
    message="tns:IApplicationService_GetData_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="ReleaseData">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/ReleaseData"
    message="tns:IApplicationService_ReleaseData_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/ReleaseDataResponse"
    message="tns:IApplicationService_ReleaseData_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="GetAsModels">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/GetAsModels"
    message="tns:IApplicationService_GetAsModels_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/GetAsModelsResponse"
    message="tns:IApplicationService_GetAsModels_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="ReleaseModels">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/ReleaseModels"
    message="tns:IApplicationService_ReleaseModels_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/ReleaseModelsResponse"
    message="tns:IApplicationService_ReleaseModels_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="QueryModel">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/QueryModel"
    message="tns:IApplicationService_QueryModel_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/QueryModelResponse"
    message="tns:IApplicationService_QueryModel_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="QueryInfoSet">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/QueryInfoSet"
    message="tns:IApplicationService_QueryInfoSet_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IApplicationService/QueryInfoSetResponse"
    message="tns:IApplicationService_QueryInfoSet_OutputMessage" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ApplicationService-20100825Binding"
type="tns:IApplicationService-20100825">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="GetState">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IApplicationService/GetState"
    style="document" />
```

```
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="SetState">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IApplicationService/SetState"
    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="BringToFront">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IApplicationService/BringToFront"
    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="NotifyDataAvailable">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IApplicationService/NotifyDataAvailable"
    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetData">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IApplicationService/GetData"
    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ReleaseData">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IApplicationService/ReleaseData"
    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
```

```
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetAsModels">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IApplicationService/GetAsModels"
      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ReleaseModels">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IApplicationService/ReleaseModels"
      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="QueryModel">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IApplicationService/QueryModel"
      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="QueryInfoSet">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IApplicationService/QueryInfoSet"
      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="ApplicationService-20100825">
    <wsdl:port name="ApplicationServiceBinding"
    binding="tns:ApplicationService-20100825Binding">
      <soap:address location="http://localhost/Service" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

B.1.2 Definition of Data Structures Used

B.1.2.1 Primary Definitions

The following is the content of ApplicationService-20100825.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://dicom.nema.org/PS3.19/ApplicationService-20100825"
elementFormDefault="qualified"
targetNamespace="http://dicom.nema.org/PS3.19/ApplicationService-20100825"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import namespace="http://schemas.microsoft.com/2003/10/Serialization/Arrays" />
  <xs:import namespace="http://schemas.datacontract.org/2004/07/System.Xml.XPath" />
  <xs:element name="GetState">
    <xs:complexType>
      <xs:sequence />
    </xs:complexType>
  </xs:element>
  <xs:element name="GetStateResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="GetStateResult" type="tns:State" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="State">
    <xs:restriction base="xs:string">
      <xs:enumeration value="IDLE" />
      <xs:enumeration value="INPROGRESS" />
      <xs:enumeration value="SUSPENDED" />
      <xs:enumeration value="COMPLETED" />
      <xs:enumeration value="CANCELED" />
      <xs:enumeration value="EXIT" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="State" nillable="true" type="tns:State" />
  <xs:element name="SetState">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="state" type="tns:State" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SetStateResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="SetStateResult" type="xs:boolean" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="BringToFront">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="location" nillable="true"
        type="tns:Rectangle" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="Rectangle">
```

```
  <xs:sequence>
    <xs:element minOccurs="0" name="Height" type="xs:int" />
    <xs:element minOccurs="0" name="Width" type="xs:int" />
    <xs:element minOccurs="0" name="RefPointX" type="xs:int" />
    <xs:element minOccurs="0" name="RefPointY" type="xs:int" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Rectangle" nillable="true" type="tns:Rectangle" />
<xs:element name="BringToFrontResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="BringToFrontResult"
      type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="NotifyDataAvailable">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="data" nillable="true"
      type="tns:AvailableData" />
      <xs:element minOccurs="0" name="lastData" type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="AvailableData">
  <xs:sequence>
    <xs:element minOccurs="0" name="ObjectDescriptors" nillable="true"
    type="tns:ArrayOfObjectDescriptor" />
    <xs:element minOccurs="0" name="Patients" nillable="true"
    type="tns:ArrayOfPatient" />
  </xs:sequence>
</xs:complexType>
<xs:element name="AvailableData" nillable="true" type="tns:AvailableData" />
<xs:complexType name="ArrayOfObjectDescriptor">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="ObjectDescriptor"
    nillable="true" type="tns:ObjectDescriptor" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfObjectDescriptor" nillable="true"
type="tns:ArrayOfObjectDescriptor" />
<xs:complexType name="ObjectDescriptor">
  <xs:sequence>
    <xs:element minOccurs="0" name="ClassUID" nillable="true"
    type="tns:UID" />
    <xs:element minOccurs="0" name="MimeType" nillable="true"
    type="tns:MimeType" />
    <xs:element minOccurs="0" name="Modality" nillable="true"
    type="tns:Modality" />
    <xs:element minOccurs="0" name="TransferSyntaxUID" nillable="true"
    type="tns:UID" />
    <xs:element minOccurs="0" name="DescriptorUuid" nillable="true"
    type="tns:UUID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ObjectDescriptor" nillable="true"
type="tns:ObjectDescriptor" />
<xs:complexType name="UID">
  <xs:sequence>
```

```
    <xs:element minOccurs="0" name="Uid" nillable="true" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="UID" nillable="true" type="tns:UID" />
<xs:complexType name="MimeType">
  <xs:sequence>
    <xs:element minOccurs="0" name="Type" nillable="true" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="MimeType" nillable="true" type="tns:MimeType" />
<xs:complexType name="Modality">
  <xs:sequence>
    <xs:element minOccurs="0" name="Modality" nillable="true"
    type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Modality" nillable="true" type="tns:Modality" />
<xs:complexType name="UUID">
  <xs:sequence>
    <xs:element minOccurs="0" name="Uuid" nillable="true" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="UUID" nillable="true" type="tns:UUID" />
<xs:complexType name="ArrayOfPatient">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Patient"
    nillable="true" type="tns:Patient" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfPatient" nillable="true"
type="tns:ArrayOfPatient" />
<xs:complexType name="Patient">
  <xs:sequence>
    <xs:element minOccurs="0" name="AssigningAuthority" nillable="true"
    type="xs:string" />
    <xs:element minOccurs="0" name="DateOfBirth" type="xs:dateTime" />
    <xs:element minOccurs="0" name="ID" nillable="true" type="xs:string" />
    <xs:element minOccurs="0" name="Name" nillable="true" type="xs:string" />
    <xs:element minOccurs="0" name="ObjectDescriptors" nillable="true"
    type="tns:ArrayOfObjectDescriptor" />
    <xs:element minOccurs="0" name="Sex" nillable="true" type="xs:string" />
    <xs:element minOccurs="0" name="Studies" nillable="true"
    type="tns:ArrayOfStudy" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Patient" nillable="true" type="tns:Patient" />
<xs:complexType name="ArrayOfStudy">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Study"
    nillable="true" type="tns:Study" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfStudy" nillable="true" type="tns:ArrayOfStudy" />
<xs:complexType name="Study">
  <xs:sequence>
    <xs:element minOccurs="0" name="ObjectDescriptors" nillable="true"
    type="tns:ArrayOfObjectDescriptor" />
    <xs:element minOccurs="0" name="Series" nillable="true"
    type="tns:ArrayOfSeries" />
    <xs:element minOccurs="0" name="StudyUID" nillable="true"
```

```
    type="tns:UID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Study" nillable="true" type="tns:Study" />
<xs:complexType name="ArrayOfSeries">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Series"
    nillable="true" type="tns:Series" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfSeries" nillable="true" type="tns:ArrayOfSeries" />
<xs:complexType name="Series">
  <xs:sequence>
    <xs:element minOccurs="0" name="ObjectDescriptors" nillable="true"
    type="tns:ArrayOfObjectDescriptor" />
    <xs:element minOccurs="0" name="SeriesUID" nillable="true"
    type="tns:UID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Series" nillable="true" type="tns:Series" />
<xs:element name="NotifyDataAvailableResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="NotifyDataAvailableResult"
      type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="GetData">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="objects" nillable="true"
      type="tns:ArrayOfUUID" />
      <xs:element minOccurs="0" name="acceptableTransferSyntaxes"
      nillable="true" type="tns:ArrayOfUID" />
      <xs:element minOccurs="0" name="includeBulkData" type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="ArrayOfUUID">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="UUID"
    nillable="true" type="tns:UUID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfUUID" nillable="true" type="tns:ArrayOfUUID" />
<xs:complexType name="ArrayOfUID">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="UID"
    nillable="true" type="tns:UID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfUID" nillable="true" type="tns:ArrayOfUID" />
<xs:element name="GetDataResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="GetDataResult" nillable="true"
      type="tns:ArrayOfObjectLocator" />
    </xs:sequence>
  </xs:complexType>
```

```
</xs:element>
<xs:complexType name="ArrayOfObjectLocator">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="ObjectLocator"
    nillable="true" type="tns:ObjectLocator" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfObjectLocator" nillable="true"
type="tns:ArrayOfObjectLocator" />
<xs:complexType name="ObjectLocator">
  <xs:sequence>
    <xs:element minOccurs="0" name="Length" type="xs:long" />
    <xs:element minOccurs="0" name="Offset" type="xs:long" />
    <xs:element minOccurs="0" name="TransferSyntax" nillable="true"
    type="tns:UID" />
    <xs:element minOccurs="0" name="URI" nillable="true" type="xs:anyURI" />
    <xs:element minOccurs="0" name="Locator" nillable="true"
    type="tns:UUID" />
    <xs:element minOccurs="0" name="Source" nillable="true"
    type="tns:UUID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ObjectLocator" nillable="true" type="tns:ObjectLocator" />
<xs:element name="ReleaseData">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="objects" nillable="true"
      type="tns:ArrayOfUUID" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ReleaseDataResponse">
  <xs:complexType>
    <xs:sequence />
  </xs:complexType>
</xs:element>
<xs:element name="GetAsModels">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="objects" nillable="true"
      type="tns:ArrayOfUUID" />
      <xs:element minOccurs="0" name="classUID" nillable="true"
      type="tns:UID" />
      <xs:element minOccurs="0" name="supportedInfoSetTypes" nillable="true"
      type="tns:ArrayOfMimeType" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="ArrayOfMimeType">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="MimeType"
    nillable="true" type="tns:MimeType" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfMimeType" nillable="true"
type="tns:ArrayOfMimeType" />
<xs:element name="GetAsModelsResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="GetAsModelsResult" nillable="true"
```

```
      type="tns:ModelSetDescriptor" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="ModelSetDescriptor">
  <xs:sequence>
    <xs:element minOccurs="0" name="FailedSourceObjects" nillable="true"
    type="tns:ArrayOfUUID" />
    <xs:element minOccurs="0" name="InfosetType" nillable="true"
    type="tns:MimeType" />
    <xs:element minOccurs="0" name="Models" nillable="true"
    type="tns:ArrayOfUUID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ModelSetDescriptor" nillable="true"
type="tns:ModelSetDescriptor" />
<xs:element name="ReleaseModels">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="models" nillable="true"
      type="tns:ArrayOfUUID" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ReleaseModelsResponse">
  <xs:complexType>
    <xs:sequence />
  </xs:complexType>
</xs:element>
<xs:element name="QueryModel">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="models" nillable="true"
      type="tns:ArrayOfUUID" />
      <xs:element minOccurs="0" name="xPaths" nillable="true"
      xmlns:q1="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
      type="q1:ArrayOfstring" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="QueryModelResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="QueryModelResult" nillable="true"
      type="tns:ArrayOfQueryResult" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="ArrayOfQueryResult">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="QueryResult"
    nillable="true" type="tns:QueryResult" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfQueryResult" nillable="true"
type="tns:ArrayOfQueryResult" />
<xs:complexType name="QueryResult">
  <xs:sequence>
    <xs:element minOccurs="0" name="Model" nillable="true" type="tns:UUID" />
    <xs:element minOccurs="0" name="Result" nillable="true"
```

```
    type="tns:ArrayOfXPathNode" />
    <xs:element minOccurs="0" name="XPath" nillable="true"
    type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="QueryResult" nillable="true" type="tns:QueryResult" />
<xs:complexType name="ArrayOfXPathNode">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="XPathNode"
    nillable="true" type="tns:XPathNode" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfXPathNode" nillable="true"
type="tns:ArrayOfXPathNode" />
<xs:complexType name="XPathNode">
  <xs:sequence>
    <xs:element minOccurs="0" name="NodeType"
    xmlns:q2="http://schemas.datacontract.org/2004/07/System.Xml.XPath"
    type="q2:XPathNodeType" />
    <xs:element minOccurs="0" name="Value" nillable="true"
    type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="XPathNode" nillable="true" type="tns:XPathNode" />
<xs:element name="QueryInfoSet">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="models" nillable="true"
      type="tns:ArrayOfUUID" />
      <xs:element minOccurs="0" name="xPaths" nillable="true"
      xmlns:q3="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
      type="q3:ArrayOfstring" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="QueryInfoSetResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="QueryInfoSetResult" nillable="true"
      type="tns:ArrayOfQueryResultInfoSet" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="ArrayOfQueryResultInfoSet">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="QueryResultInfoSet"
    nillable="true" type="tns:QueryResultInfoSet" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfQueryResultInfoSet" nillable="true"
type="tns:ArrayOfQueryResultInfoSet" />
<xs:complexType name="QueryResultInfoSet">
  <xs:sequence>
    <xs:element minOccurs="0" name="Model" nillable="true" type="tns:UUID" />
    <xs:element minOccurs="0" name="Result" nillable="true"
    type="tns:ArrayOfXPathNodeInfoSet" />
    <xs:element minOccurs="0" name="XPath" nillable="true"
    type="xs:string" />
  </xs:sequence>
</xs:complexType>
```

```
  <xs:element name="QueryResultInfoSet" nillable="true"
  type="tns:QueryResultInfoSet" />
  <xs:complexType name="ArrayOfXPathNodeInfoSet">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="XPathNodeInfoSet"
      nillable="true" type="tns:XPathNodeInfoSet" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfXPathNodeInfoSet" nillable="true"
  type="tns:ArrayOfXPathNodeInfoSet" />
  <xs:complexType name="XPathNodeInfoSet">
    <xs:sequence>
      <xs:element minOccurs="0" name="InfoSetValue" nillable="true"
      type="xs:base64Binary" />
      <xs:element minOccurs="0" name="NodeType"
      xmlns:q4="http://schemas.datacontract.org/2004/07/System.Xml.XPath"
      type="q4:XPathNodeType" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="XPathNodeInfoSet" nillable="true"
  type="tns:XPathNodeInfoSet" />
</xs:schema>
```

B.1.2.2 Referenced Definitions

The following is the content of XPathNodeType.xsd:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.datacontract.org/2004/07/System.Xml.XPath"
elementFormDefault="qualified"
targetNamespace="http://schemas.datacontract.org/2004/07/System.Xml.XPath"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="XPathNodeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Root" />
      <xs:enumeration value="Element" />
      <xs:enumeration value="Attribute" />
      <xs:enumeration value="Namespace" />
      <xs:enumeration value="Text" />
      <xs:enumeration value="SignificantWhitespace" />
      <xs:enumeration value="Whitespace" />
      <xs:enumeration value="ProcessingInstruction" />
      <xs:enumeration value="Comment" />
      <xs:enumeration value="All" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="XPathNodeType" nillable="true" type="tns:XPathNodeType" />
</xs:schema>
```

The following is the content of Types.xsd:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/2003/10/Serialization/"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="anyType" nillable="true" type="xs:anyType" />
  <xs:element name="anyURI" nillable="true" type="xs:anyURI" />
  <xs:element name="base64Binary" nillable="true" type="xs:base64Binary" />
  <xs:element name="boolean" nillable="true" type="xs:boolean" />
```

```
  <xs:element name="byte" nillable="true" type="xs:byte" />
  <xs:element name="dateTime" nillable="true" type="xs:dateTime" />
  <xs:element name="decimal" nillable="true" type="xs:decimal" />
  <xs:element name="double" nillable="true" type="xs:double" />
  <xs:element name="float" nillable="true" type="xs:float" />
  <xs:element name="int" nillable="true" type="xs:int" />
  <xs:element name="long" nillable="true" type="xs:long" />
  <xs:element name="QName" nillable="true" type="xs:QName" />
  <xs:element name="short" nillable="true" type="xs:short" />
  <xs:element name="string" nillable="true" type="xs:string" />
  <xs:element name="unsignedByte" nillable="true" type="xs:unsignedByte" />
  <xs:element name="unsignedInt" nillable="true" type="xs:unsignedInt" />
  <xs:element name="unsignedLong" nillable="true" type="xs:unsignedLong" />
  <xs:element name="unsignedShort" nillable="true" type="xs:unsignedShort" />
  <xs:element name="char" nillable="true" type="tns:char" />
  <xs:simpleType name="char">
    <xs:restriction base="xs:int" />
  </xs:simpleType>
  <xs:element name="duration" nillable="true" type="tns:duration" />
  <xs:simpleType name="duration">
    <xs:restriction base="xs:duration">
      <xs:pattern value="\-?P(\d*D)?(T(\d*H)?(\d*M)?(\d*(\.\d*)?S)?)?" />
      <xs:minInclusive value="-P10675199DT2H48M5.4775808S" />
      <xs:maxInclusive value="P10675199DT2H48M5.4775807S" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="guid" nillable="true" type="tns:guid" />
  <xs:simpleType name="guid">
    <xs:restriction base="xs:string">
      <xs:pattern value="[\da-fA-F]{8}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F]{12}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:attribute name="FactoryType" type="xs:QName" />
  <xs:attribute name="Id" type="xs:ID" />
  <xs:attribute name="Ref" type="xs:IDREF" />
</xs:schema>
```

The following is the content of ArrayOfString.xsd:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
elementFormDefault="qualified"
targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="ArrayOfstring">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="string"
      nillable="true" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfstring" nillable="true" type="tns:ArrayOfstring" />
</xs:schema>
```

B.2 Host Interface - Version 20100825

B.2.1 WSDL Definition of the Interface

The following is the content of HostService-20100825.wsdl:

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions name="HostService-20100825"
targetNamespace="http://dicom.nema.org/PS3.19/HostService-20100825"
xmlns:tns="http://dicom.nema.org/PS3.19/HostService-20100825"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:wsa10="http://www.w3.org/2005/08/addressing"
xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <xsd:schema targetNamespace="http://dicom.nema.org/PS3.19/Imports/HostService-20100825">

      <xsd:import namespace="http://dicom.nema.org/PS3.19/HostService-20100825"
      schemaLocation="./HostService-20100825.xsd" />
      <xsd:import namespace="http://schemas.microsoft.com/2003/10/Serialization/"
      schemaLocation="./Types.xsd" />
      <xsd:import namespace="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
      schemaLocation="./ArrayOfString.xsd" />
      <xsd:import namespace="http://schemas.datacontract.org/2004/07/System.Xml.XPath"
      schemaLocation="./XPathNodeType.xsd" />
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="IHostService_GenerateUID_InputMessage">
    <wsdl:part name="parameters" element="tns:GenerateUID" />
  </wsdl:message>
  <wsdl:message name="IHostService_GenerateUID_OutputMessage">
    <wsdl:part name="parameters" element="tns:GenerateUIDResponse" />
  </wsdl:message>
  <wsdl:message name="IHostService_GetAvailableScreen_InputMessage">
    <wsdl:part name="parameters" element="tns:GetAvailableScreen" />
  </wsdl:message>
  <wsdl:message name="IHostService_GetAvailableScreen_OutputMessage">
    <wsdl:part name="parameters" element="tns:GetAvailableScreenResponse" />
  </wsdl:message>
  <wsdl:message name="IHostService_GetOutputLocation_InputMessage">
    <wsdl:part name="parameters" element="tns:GetOutputLocation" />
  </wsdl:message>
  <wsdl:message name="IHostService_GetOutputLocation_OutputMessage">
    <wsdl:part name="parameters" element="tns:GetOutputLocationResponse" />
  </wsdl:message>
  <wsdl:message name="IHostService_NotifyStateChanged_InputMessage">
    <wsdl:part name="parameters" element="tns:NotifyStateChanged" />
  </wsdl:message>
  <wsdl:message name="IHostService_NotifyStateChanged_OutputMessage">
    <wsdl:part name="parameters" element="tns:NotifyStateChangedResponse" />
  </wsdl:message>
  <wsdl:message name="IHostService_NotifyStatus_InputMessage">
    <wsdl:part name="parameters" element="tns:NotifyStatus" />
  </wsdl:message>
  <wsdl:message name="IHostService_NotifyStatus_OutputMessage">
    <wsdl:part name="parameters" element="tns:NotifyStatusResponse" />
```

```
</wsdl:message>
<wsdl:message name="IHostService_NotifyDataAvailable_InputMessage">
  <wsdl:part name="parameters" element="tns:NotifyDataAvailable" />
</wsdl:message>
<wsdl:message name="IHostService_NotifyDataAvailable_OutputMessage">
  <wsdl:part name="parameters" element="tns:NotifyDataAvailableResponse" />
</wsdl:message>
<wsdl:message name="IHostService_GetData_InputMessage">
  <wsdl:part name="parameters" element="tns:GetData" />
</wsdl:message>
<wsdl:message name="IHostService_GetData_OutputMessage">
  <wsdl:part name="parameters" element="tns:GetDataResponse" />
</wsdl:message>
<wsdl:message name="IHostService_ReleaseData_InputMessage">
  <wsdl:part name="parameters" element="tns:ReleaseData" />
</wsdl:message>
<wsdl:message name="IHostService_ReleaseData_OutputMessage">
  <wsdl:part name="parameters" element="tns:ReleaseDataResponse" />
</wsdl:message>
<wsdl:message name="IHostService_GetAsModels_InputMessage">
  <wsdl:part name="parameters" element="tns:GetAsModels" />
</wsdl:message>
<wsdl:message name="IHostService_GetAsModels_OutputMessage">
  <wsdl:part name="parameters" element="tns:GetAsModelsResponse" />
</wsdl:message>
<wsdl:message name="IHostService_ReleaseModels_InputMessage">
  <wsdl:part name="parameters" element="tns:ReleaseModels" />
</wsdl:message>
<wsdl:message name="IHostService_ReleaseModels_OutputMessage">
  <wsdl:part name="parameters" element="tns:ReleaseModelsResponse" />
</wsdl:message>
<wsdl:message name="IHostService_QueryModel_InputMessage">
  <wsdl:part name="parameters" element="tns:QueryModel" />
</wsdl:message>
<wsdl:message name="IHostService_QueryModel_OutputMessage">
  <wsdl:part name="parameters" element="tns:QueryModelResponse" />
</wsdl:message>
<wsdl:message name="IHostService_QueryInfoSet_InputMessage">
  <wsdl:part name="parameters" element="tns:QueryInfoSet" />
</wsdl:message>
<wsdl:message name="IHostService_QueryInfoSet_OutputMessage">
  <wsdl:part name="parameters" element="tns:QueryInfoSetResponse" />
</wsdl:message>
<wsdl:portType name="IHostService-20100825">
  <wsdl:operation name="GenerateUID">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/GenerateUID"
    message="tns:IHostService_GenerateUID_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/GenerateUIDResponse"
    message="tns:IHostService_GenerateUID_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="GetAvailableScreen">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/GetAvailableScreen"
    message="tns:IHostService_GetAvailableScreen_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/GetAvailableScreenResponse"
    message="tns:IHostService_GetAvailableScreen_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="GetOutputLocation">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/GetOutputLocation"
```

```
    message="tns:IHostService_GetOutputLocation_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/GetOutputLocationResponse"
    message="tns:IHostService_GetOutputLocation_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="NotifyStateChanged">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/NotifyStateChanged"
    message="tns:IHostService_NotifyStateChanged_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/NotifyStateChangedResponse"
    message="tns:IHostService_NotifyStateChanged_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="NotifyStatus">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/NotifyStatus"
    message="tns:IHostService_NotifyStatus_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/NotifyStatusResponse"
    message="tns:IHostService_NotifyStatus_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="NotifyDataAvailable">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/NotifyDataAvailable"
    message="tns:IHostService_NotifyDataAvailable_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/NotifyDataAvailableResponse"
    message="tns:IHostService_NotifyDataAvailable_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="GetData">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/GetData"
    message="tns:IHostService_GetData_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/GetDataResponse"
    message="tns:IHostService_GetData_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="ReleaseData">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/ReleaseData"
    message="tns:IHostService_ReleaseData_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/ReleaseDataResponse"
    message="tns:IHostService_ReleaseData_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="GetAsModels">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/GetAsModels"
    message="tns:IHostService_GetAsModels_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/GetAsModelsResponse"
    message="tns:IHostService_GetAsModels_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="ReleaseModels">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/ReleaseModels"
    message="tns:IHostService_ReleaseModels_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/ReleaseModelsResponse"
    message="tns:IHostService_ReleaseModels_OutputMessage" />
  </wsdl:operation>
  <wsdl:operation name="QueryModel">
    <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/QueryModel"
    message="tns:IHostService_QueryModel_InputMessage" />
    <wsdl:output
      wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/QueryModelResponse"
    message="tns:IHostService_QueryModel_OutputMessage" />
```

```
    </wsdl:operation>
    <wsdl:operation name="QueryInfoSet">
      <wsdl:input wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/QueryInfoSet"
      message="tns:IHostService_QueryInfoSet_InputMessage" />
      <wsdl:output
        wsaw:Action="http://dicom.nema.org/PS3.19/IHostService/QueryInfoSetResponse"
      message="tns:IHostService_QueryInfoSet_OutputMessage" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="HostService-YYYYNNDDBinding"
  type="tns:IHostService-20100825">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GenerateUID">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IHostService/GenerateUID"
      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetAvailableScreen">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IHostService/GetAvailableScreen"
      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetOutputLocation">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IHostService/GetOutputLocation"
      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="NotifyStateChanged">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IHostService/NotifyStateChanged"
      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="NotifyStatus">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IHostService/NotifyStatus"
      style="document" />
```

```
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="NotifyDataAvailable">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IHostService/NotifyDataAvailable"
    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetData">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IHostService/GetData"
    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ReleaseData">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IHostService/ReleaseData"
    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetAsModels">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IHostService/GetAsModels"
    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="ReleaseModels">
    <soap:operation
       soapAction="http://dicom.nema.org/PS3.19/IHostService/ReleaseModels"
    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
```

```
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="QueryModel">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IHostService/QueryModel"
      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="QueryInfoSet">
      <soap:operation
         soapAction="http://dicom.nema.org/PS3.19/IHostService/QueryInfoSet"
      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="HostService-20100825">
    <wsdl:port name="HostServiceBinding"
    binding="tns:HostService-YYYYNNDDBinding">
      <soap:address location="http://localhost/Service" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

B.2.2 Definition of Data Structures Used

B.2.2.1 Primary Definitions

The following is the the contents of HostService-20100825.xsd:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://dicom.nema.org/PS3.19/HostService-20100825"
elementFormDefault="qualified"
targetNamespace="http://dicom.nema.org/PS3.19/HostService-20100825"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import namespace="http://schemas.microsoft.com/2003/10/Serialization/Arrays" />
  <xs:import namespace="http://schemas.datacontract.org/2004/07/System.Xml.XPath" />
  <xs:element name="GenerateUID">
    <xs:complexType>
      <xs:sequence />
    </xs:complexType>
  </xs:element>
  <xs:element name="GenerateUIDResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="GenerateUIDResult" nillable="true"
        type="tns:UID" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
<xs:complexType name="UID">
  <xs:sequence>
    <xs:element minOccurs="0" name="Uid" nillable="true" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="UID" nillable="true" type="tns:UID" />
<xs:element name="GetAvailableScreen">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="preferredScreen" nillable="true"
      type="tns:Rectangle" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="Rectangle">
  <xs:sequence>
    <xs:element minOccurs="0" name="Height" type="xs:int" />
    <xs:element minOccurs="0" name="Width" type="xs:int" />
    <xs:element minOccurs="0" name="RefPointX" type="xs:int" />
    <xs:element minOccurs="0" name="RefPointY" type="xs:int" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Rectangle" nillable="true" type="tns:Rectangle" />
<xs:element name="GetAvailableScreenResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="GetAvailableScreenResult"
      nillable="true" type="tns:Rectangle" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="GetOutputLocation">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="preferredProtocols" nillable="true"
      xmlns:q1="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
      type="q1:ArrayOfstring" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="GetOutputLocationResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="GetOutputLocationResult"
      nillable="true" type="xs:anyURI" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="NotifyStateChanged">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="state" type="tns:State" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:simpleType name="State">
  <xs:restriction base="xs:string">
    <xs:enumeration value="IDLE" />
    <xs:enumeration value="INPROGRESS" />
    <xs:enumeration value="SUSPENDED" />
```

```
      <xs:enumeration value="COMPLETED" />
      <xs:enumeration value="CANCELED" />
      <xs:enumeration value="EXIT" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="State" nillable="true" type="tns:State" />
  <xs:element name="NotifyStateChangedResponse">
    <xs:complexType>
      <xs:sequence />
    </xs:complexType>
  </xs:element>
  <xs:element name="NotifyStatus">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="status" nillable="true"
        type="tns:Status" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="Status">
    <xs:sequence>
      <xs:element minOccurs="0" name="StatusType" type="tns:StatusType" />
      <xs:element minOccurs="0" name="CodeValue" type="xs:int" />
      <xs:element minOccurs="0" name="CodingSchemeDesignator" nillable="true"
      type="xs:string" />
      <xs:element minOccurs="0" name="CodeMeaning" nillable="true"
      type="xs:string" />
      <xs:element minOccurs="0" name="ContextIdentifier" nillable="true"
      type="xs:string" />
      <xs:element minOccurs="0" name="MappingResource" nillable="true"
      type="xs:string" />
      <xs:element minOccurs="0" name="ContextGroupVersion" nillable="true"
      type="xs:string" />
      <xs:element minOccurs="0" name="ContextGroupExtensionFlag"
      nillable="true" type="xs:string" />
      <xs:element minOccurs="0" name="ContextGroupLocalVersion" nillable="true"
      type="xs:string" />
      <xs:element minOccurs="0" name="ContextGroupExtensionCreatorUID"
      nillable="true" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Status" nillable="true" type="tns:Status" />
  <xs:simpleType name="StatusType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="INFORMATION" />
      <xs:enumeration value="WARNING" />
      <xs:enumeration value="ERROR" />
      <xs:enumeration value="FATALERROR" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="StatusType" nillable="true" type="tns:StatusType" />
  <xs:element name="NotifyStatusResponse">
    <xs:complexType>
      <xs:sequence />
    </xs:complexType>
  </xs:element>
  <xs:element name="NotifyDataAvailable">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="data" nillable="true"
```

```
      type="tns:AvailableData" />
      <xs:element minOccurs="0" name="lastData" type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="AvailableData">
  <xs:sequence>
    <xs:element minOccurs="0" name="ObjectDescriptors" nillable="true"
    type="tns:ArrayOfObjectDescriptor" />
    <xs:element minOccurs="0" name="Patients" nillable="true"
    type="tns:ArrayOfPatient" />
  </xs:sequence>
</xs:complexType>
<xs:element name="AvailableData" nillable="true" type="tns:AvailableData" />
<xs:complexType name="ArrayOfObjectDescriptor">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="ObjectDescriptor"
    nillable="true" type="tns:ObjectDescriptor" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfObjectDescriptor" nillable="true"
type="tns:ArrayOfObjectDescriptor" />
<xs:complexType name="ObjectDescriptor">
  <xs:sequence>
    <xs:element minOccurs="0" name="ClassUID" nillable="true"
    type="tns:UID" />
    <xs:element minOccurs="0" name="MimeType" nillable="true"
    type="tns:MimeType" />
    <xs:element minOccurs="0" name="Modality" nillable="true"
    type="tns:Modality" />
    <xs:element minOccurs="0" name="TransferSyntaxUID" nillable="true"
    type="tns:UID" />
    <xs:element minOccurs="0" name="DescriptorUuid" nillable="true"
    type="tns:UUID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ObjectDescriptor" nillable="true"
type="tns:ObjectDescriptor" />
<xs:complexType name="MimeType">
  <xs:sequence>
    <xs:element minOccurs="0" name="Type" nillable="true" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="MimeType" nillable="true" type="tns:MimeType" />
<xs:complexType name="Modality">
  <xs:sequence>
    <xs:element minOccurs="0" name="Modality" nillable="true"
    type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Modality" nillable="true" type="tns:Modality" />
<xs:complexType name="UUID">
  <xs:sequence>
    <xs:element minOccurs="0" name="Uuid" nillable="true" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="UUID" nillable="true" type="tns:UUID" />
<xs:complexType name="ArrayOfPatient">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Patient"
```

```
    nillable="true" type="tns:Patient" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfPatient" nillable="true"
type="tns:ArrayOfPatient" />
<xs:complexType name="Patient">
  <xs:sequence>
    <xs:element minOccurs="0" name="AssigningAuthority" nillable="true"
    type="xs:string" />
    <xs:element minOccurs="0" name="DateOfBirth" type="xs:dateTime" />
    <xs:element minOccurs="0" name="ID" nillable="true" type="xs:string" />
    <xs:element minOccurs="0" name="Name" nillable="true" type="xs:string" />
    <xs:element minOccurs="0" name="ObjectDescriptors" nillable="true"
    type="tns:ArrayOfObjectDescriptor" />
    <xs:element minOccurs="0" name="Sex" nillable="true" type="xs:string" />
    <xs:element minOccurs="0" name="Studies" nillable="true"
    type="tns:ArrayOfStudy" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Patient" nillable="true" type="tns:Patient" />
<xs:complexType name="ArrayOfStudy">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Study"
    nillable="true" type="tns:Study" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfStudy" nillable="true" type="tns:ArrayOfStudy" />
<xs:complexType name="Study">
  <xs:sequence>
    <xs:element minOccurs="0" name="ObjectDescriptors" nillable="true"
    type="tns:ArrayOfObjectDescriptor" />
    <xs:element minOccurs="0" name="Series" nillable="true"
    type="tns:ArrayOfSeries" />
    <xs:element minOccurs="0" name="StudyUID" nillable="true"
    type="tns:UID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Study" nillable="true" type="tns:Study" />
<xs:complexType name="ArrayOfSeries">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="Series"
    nillable="true" type="tns:Series" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfSeries" nillable="true" type="tns:ArrayOfSeries" />
<xs:complexType name="Series">
  <xs:sequence>
    <xs:element minOccurs="0" name="ObjectDescriptors" nillable="true"
    type="tns:ArrayOfObjectDescriptor" />
    <xs:element minOccurs="0" name="SeriesUID" nillable="true"
    type="tns:UID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="Series" nillable="true" type="tns:Series" />
<xs:element name="NotifyDataAvailableResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="NotifyDataAvailableResult"
      type="xs:boolean" />
    </xs:sequence>
```

```
  </xs:complexType>
</xs:element>
<xs:element name="GetData">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="objects" nillable="true"
      type="tns:ArrayOfUUID" />
      <xs:element minOccurs="0" name="acceptableTransferSyntaxes"
      nillable="true" type="tns:ArrayOfUID" />
      <xs:element minOccurs="0" name="includeBulkData" type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="ArrayOfUUID">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="UUID"
    nillable="true" type="tns:UUID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfUUID" nillable="true" type="tns:ArrayOfUUID" />
<xs:complexType name="ArrayOfUID">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="UID"
    nillable="true" type="tns:UID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfUID" nillable="true" type="tns:ArrayOfUID" />
<xs:element name="GetDataResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="GetDataResult" nillable="true"
      type="tns:ArrayOfObjectLocator" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="ArrayOfObjectLocator">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="ObjectLocator"
    nillable="true" type="tns:ObjectLocator" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfObjectLocator" nillable="true"
type="tns:ArrayOfObjectLocator" />
<xs:complexType name="ObjectLocator">
  <xs:sequence>
    <xs:element minOccurs="0" name="Length" type="xs:long" />
    <xs:element minOccurs="0" name="Offset" type="xs:long" />
    <xs:element minOccurs="0" name="TransferSyntax" nillable="true"
    type="tns:UID" />
    <xs:element minOccurs="0" name="URI" nillable="true" type="xs:anyURI" />
    <xs:element minOccurs="0" name="Locator" nillable="true"
    type="tns:UUID" />
    <xs:element minOccurs="0" name="Source" nillable="true"
    type="tns:UUID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ObjectLocator" nillable="true" type="tns:ObjectLocator" />
<xs:element name="ReleaseData">
  <xs:complexType>
    <xs:sequence>
```

```
      <xs:element minOccurs="0" name="objects" nillable="true"
      type="tns:ArrayOfUUID" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ReleaseDataResponse">
  <xs:complexType>
    <xs:sequence />
  </xs:complexType>
</xs:element>
<xs:element name="GetAsModels">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="objects" nillable="true"
      type="tns:ArrayOfUUID" />
      <xs:element minOccurs="0" name="classUID" nillable="true"
      type="tns:UID" />
      <xs:element minOccurs="0" name="supportedInfoSetTypes" nillable="true"
      type="tns:ArrayOfMimeType" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="ArrayOfMimeType">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="MimeType"
    nillable="true" type="tns:MimeType" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfMimeType" nillable="true"
type="tns:ArrayOfMimeType" />
<xs:element name="GetAsModelsResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="GetAsModelsResult" nillable="true"
      type="tns:ModelSetDescriptor" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="ModelSetDescriptor">
  <xs:sequence>
    <xs:element minOccurs="0" name="FailedSourceObjects" nillable="true"
    type="tns:ArrayOfUUID" />
    <xs:element minOccurs="0" name="InfosetType" nillable="true"
    type="tns:MimeType" />
    <xs:element minOccurs="0" name="Models" nillable="true"
    type="tns:ArrayOfUUID" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ModelSetDescriptor" nillable="true"
type="tns:ModelSetDescriptor" />
<xs:element name="ReleaseModels">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="models" nillable="true"
      type="tns:ArrayOfUUID" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ReleaseModelsResponse">
  <xs:complexType>
```

```
    <xs:sequence />
  </xs:complexType>
</xs:element>
<xs:element name="QueryModel">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="models" nillable="true"
      type="tns:ArrayOfUUID" />
      <xs:element minOccurs="0" name="xPaths" nillable="true"
      xmlns:q2="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
      type="q2:ArrayOfstring" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="QueryModelResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="QueryModelResult" nillable="true"
      type="tns:ArrayOfQueryResult" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType name="ArrayOfQueryResult">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="QueryResult"
    nillable="true" type="tns:QueryResult" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfQueryResult" nillable="true"
type="tns:ArrayOfQueryResult" />
<xs:complexType name="QueryResult">
  <xs:sequence>
    <xs:element minOccurs="0" name="Model" nillable="true" type="tns:UUID" />
    <xs:element minOccurs="0" name="Result" nillable="true"
    type="tns:ArrayOfXPathNode" />
    <xs:element minOccurs="0" name="XPath" nillable="true"
    type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="QueryResult" nillable="true" type="tns:QueryResult" />
<xs:complexType name="ArrayOfXPathNode">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="XPathNode"
    nillable="true" type="tns:XPathNode" />
  </xs:sequence>
</xs:complexType>
<xs:element name="ArrayOfXPathNode" nillable="true"
type="tns:ArrayOfXPathNode" />
<xs:complexType name="XPathNode">
  <xs:sequence>
    <xs:element minOccurs="0" name="NodeType"
    xmlns:q3="http://schemas.datacontract.org/2004/07/System.Xml.XPath"
    type="q3:XPathNodeType" />
    <xs:element minOccurs="0" name="Value" nillable="true"
    type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:element name="XPathNode" nillable="true" type="tns:XPathNode" />
<xs:element name="QueryInfoSet">
  <xs:complexType>
```

```
      <xs:sequence>
        <xs:element minOccurs="0" name="models" nillable="true"
        type="tns:ArrayOfUUID" />
        <xs:element minOccurs="0" name="xPaths" nillable="true"
        xmlns:q4="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
        type="q4:ArrayOfstring" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="QueryInfoSetResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="QueryInfoSetResult" nillable="true"
        type="tns:ArrayOfQueryResultInfoSet" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="ArrayOfQueryResultInfoSet">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="QueryResultInfoSet"
      nillable="true" type="tns:QueryResultInfoSet" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfQueryResultInfoSet" nillable="true"
  type="tns:ArrayOfQueryResultInfoSet" />
  <xs:complexType name="QueryResultInfoSet">
    <xs:sequence>
      <xs:element minOccurs="0" name="Model" nillable="true" type="tns:UUID" />
      <xs:element minOccurs="0" name="Result" nillable="true"
      type="tns:ArrayOfXPathNodeInfoSet" />
      <xs:element minOccurs="0" name="XPath" nillable="true"
      type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="QueryResultInfoSet" nillable="true"
  type="tns:QueryResultInfoSet" />
  <xs:complexType name="ArrayOfXPathNodeInfoSet">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="XPathNodeInfoSet"
      nillable="true" type="tns:XPathNodeInfoSet" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfXPathNodeInfoSet" nillable="true"
  type="tns:ArrayOfXPathNodeInfoSet" />
  <xs:complexType name="XPathNodeInfoSet">
    <xs:sequence>
      <xs:element minOccurs="0" name="InfoSetValue" nillable="true"
      type="xs:base64Binary" />
      <xs:element minOccurs="0" name="NodeType"
      xmlns:q5="http://schemas.datacontract.org/2004/07/System.Xml.XPath"
      type="q5:XPathNodeType" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="XPathNodeInfoSet" nillable="true"
  type="tns:XPathNodeInfoSet" />
</xs:schema>
```

B.2.2.2 Referenced Definitions

The following is the content of XPathNodeType.xsd:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.datacontract.org/2004/07/System.Xml.XPath"
elementFormDefault="qualified"
targetNamespace="http://schemas.datacontract.org/2004/07/System.Xml.XPath"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="XPathNodeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Root" />
      <xs:enumeration value="Element" />
      <xs:enumeration value="Attribute" />
      <xs:enumeration value="Namespace" />
      <xs:enumeration value="Text" />
      <xs:enumeration value="SignificantWhitespace" />
      <xs:enumeration value="Whitespace" />
      <xs:enumeration value="ProcessingInstruction" />
      <xs:enumeration value="Comment" />
      <xs:enumeration value="All" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="XPathNodeType" nillable="true" type="tns:XPathNodeType" />
</xs:schema>
```

The following is the content of Types.xsd:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/2003/10/Serialization/"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="anyType" nillable="true" type="xs:anyType" />
  <xs:element name="anyURI" nillable="true" type="xs:anyURI" />
  <xs:element name="base64Binary" nillable="true" type="xs:base64Binary" />
  <xs:element name="boolean" nillable="true" type="xs:boolean" />
  <xs:element name="byte" nillable="true" type="xs:byte" />
  <xs:element name="dateTime" nillable="true" type="xs:dateTime" />
  <xs:element name="decimal" nillable="true" type="xs:decimal" />
  <xs:element name="double" nillable="true" type="xs:double" />
  <xs:element name="float" nillable="true" type="xs:float" />
  <xs:element name="int" nillable="true" type="xs:int" />
  <xs:element name="long" nillable="true" type="xs:long" />
  <xs:element name="QName" nillable="true" type="xs:QName" />
  <xs:element name="short" nillable="true" type="xs:short" />
  <xs:element name="string" nillable="true" type="xs:string" />
  <xs:element name="unsignedByte" nillable="true" type="xs:unsignedByte" />
  <xs:element name="unsignedInt" nillable="true" type="xs:unsignedInt" />
  <xs:element name="unsignedLong" nillable="true" type="xs:unsignedLong" />
  <xs:element name="unsignedShort" nillable="true" type="xs:unsignedShort" />
  <xs:element name="char" nillable="true" type="tns:char" />
  <xs:simpleType name="char">
    <xs:restriction base="xs:int" />
  </xs:simpleType>
  <xs:element name="duration" nillable="true" type="tns:duration" />
  <xs:simpleType name="duration">
    <xs:restriction base="xs:duration">
      <xs:pattern value="\-?P(\d*D)?(T(\d*H)?(\d*M)?(\d*(\.\d*)?S)?)?" />
      <xs:minInclusive value="-P10675199DT2H48M5.4775808S" />
```

```
      <xs:maxInclusive value="P10675199DT2H48M5.4775807S" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="guid" nillable="true" type="tns:guid" />
  <xs:simpleType name="guid">
    <xs:restriction base="xs:string">
      <xs:pattern value="[\da-fA-F]{8}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F]{4}-[\da-fA-F]{12}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:attribute name="FactoryType" type="xs:QName" />
  <xs:attribute name="Id" type="xs:ID" />
  <xs:attribute name="Ref" type="xs:IDREF" />
</xs:schema>
```

The following is the content of ArrayOfString.xsd:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
elementFormDefault="qualified"
targetNamespace="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="ArrayOfstring">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="string"
      nillable="true" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ArrayOfstring" nillable="true" type="tns:ArrayOfstring" />
</xs:schema>
```