DICOM Application Hosting API

UML Class Diagram Example

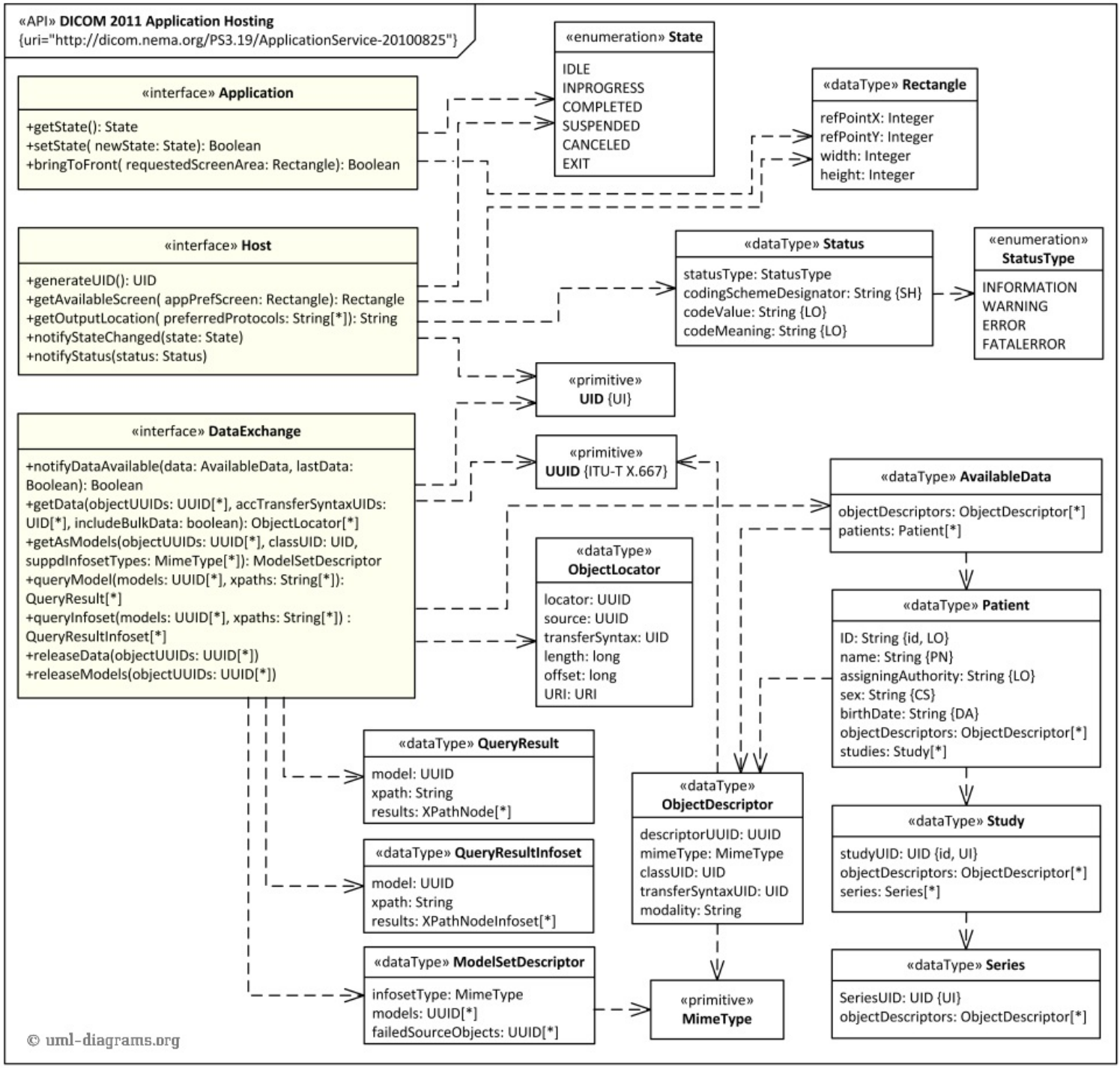
An example of class diagram representing **DICOM Application Hosting API**. The DICOM Application Hosting API was published in Digital Imaging and Communications in Medicine (DICOM) Standard, Part 19 Application Hosting (PS 3.19-2011). Example shown below is our own interpretation of the API, please refer to the DICOM Standard for official description and diagrams.

The Part 19 of the DICOM Standard defines an application programming interface (API) between two software applications - **Hosting System** and **Hosted Application**, both applications exchanging medical data while located on the same system.

The **Hosting System (Host)** (aka Host Application) is an 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. Host application provides the hosted one with data, such as a set of medical images and related metadata.

The **Hosted Application (Application)** is an application launched and controlled by a hosting system, which may also utilize some services offered by the hosting system. Hosted application processes provided medical data, potentially returning back some newly generated data, for example in the form of another set of images and/or structured reports, to the hosting 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.



DICOM Application Hosting API UML class diagram example.

There are three interfaces defined by the DICOM Application Hosting API.

The **Application** interface encapsulates the hosted application, and is invoked by the hosting system to control the hosted application. The getState() method allows to query hosted application of its current state. Hosting system could request hosted application to switch to a new state using setState() method. By calling bringToFront() method, the hosting system could ask hosted application to make its GUI visible as the topmost window, and to gain focus.

The **Host** interface 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. For example, hosting system could generate new DICOM UID for the hosted application to use. The hosted application may inform the hosting system of notable events that occur during execution by invoking notifyStatus() method.

The **DataExchange** interface is an interface which is used by both the hosting system and the hosted application to exchange medical data directly or indirectly, and thus must be implemented by both. 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. Model-based methods can work with a variety of models which can be either some abstraction of the data, or can be a model of some native format, including models defined by the DICOM Standard. Particular models are identified by a UID and described as XML Infosets, even though the original data might never be actually represented in XML form.

The DICOM Application Hosting API uses **ArrayOf[Type]** wrapper class to represent an array of a specific type "to enable cross-platform

compatibility". Our example UML diagram provided here uses standard UML multiplicity instead.

Noticed a spelling error? Select the text using the mouse and press Ctrl + Enter.

by Kirill Fakhroutdinov

This document describes **UML 2.5** and is based on **OMG™ Unified Modeling Language™ (OMG UML®) 2.5** specification ***[UML 2.5 FTF - Beta 1]***.
All UML diagrams were created in **Microsoft Visio** 2007-2016 using ***UML 2.2 stencils***. You can send your comments and suggestions to webmaster at **webmaster@uml-diagrams.org**.