DICOM Hosted Application Life Cycle

UML State Machine Diagram Example

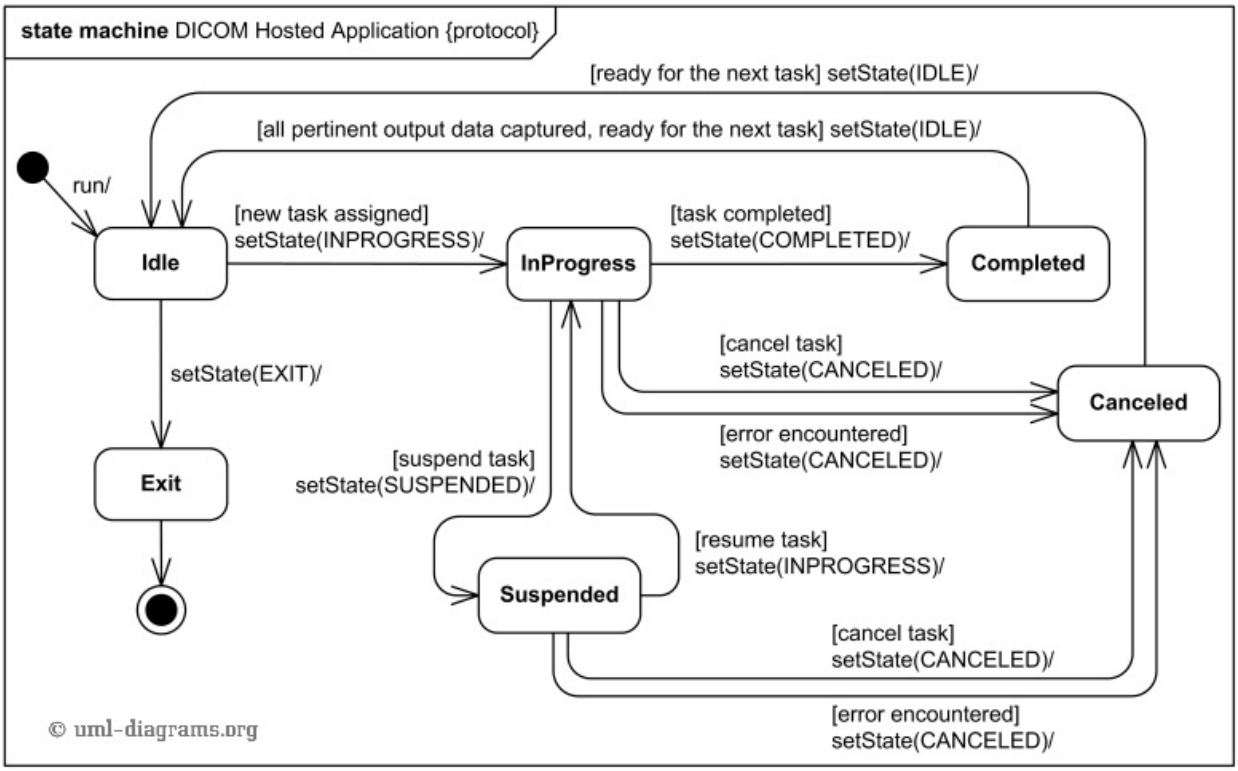
An example of UML **protocol state machine diagram** for **DICOM Application Hosting API**. The DICOM Application Hosting API was published in Digital Imaging and Communications in Medicine (DICOM) Standard, Part 19 Application Hosting (PS 3.19-2011). Example shown below is our own interpretation of 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. 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. An end user launches the hosting system, and all subsequent interactions start in the hosting system.



DICOM hosted application life cycle UML protocol state machine diagram example.

Hosting system initializes hosted application by issuing a run or exec command or its equivalent. Once the hosted application is initialized, its state is becomes the **Idle** state. Application is required to notify host of any state changes by calling the notifyStateChanged() method but this method itself is not part of any state transition.

While in the IDLE state the hosted application is awaiting for a new task assignment from the hosting system.

Once the hosted application started performing some task assigned, it changes its state to **InProgress**.

After the application has completed processing it moves into **Completed** state. In this state it waits for the host to access all output data from hosted application, and releases all resources after that.

Application could also be **Suspended**. In this state processing of the current task is stopped and as many resources as possible are released. Still, the hosted application should keep as much information as needed to be able to resume processing.

Application could transition to the **Canceled** state either by incoming request to cancel task or in the case when the application

encounters a non fatal error that prevents further processing of the current task, while still capable to process another task.

While in the **Canceled** state application stops all processing and should release all resources. There is no chance to resume processing of a canceled task from this state.

The **Exit** state is the terminal state of the hosted application.

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

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**.