**M S Ramaiah Institute of Technology**

(An Autonomous Institute, Affiliated to VTU)

MSR Nagar, MSRIT post, Bangalore-54.

A **Design** for

**BUILDING A WEB SERVICE FRAMEWORK (STOW-RS) FOR MEDICAL IMAGING (DICOM)**

Submitted By:

**NEHA PATHAPATI**

**(USN:** 1MS12CS143)

**COMPANY:** PHILIPS

**BUSINESS UNIT:** HEALTH SYSTEMS

*in partial fulfillment for the award of the degree of*

# *Bachelor of Engineering in Computer Science & Engineering*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**M.S.RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

[www.msrit.edu](http://www.msrit.edu), **Feb 2016**

DESIGN

**INTRODUCTION**

The goal of our project is to build a STOW – RS (Store Over the Web – Restful Service) service to facilitate third party clients to post DICOM (Digital Imaging and Communication in Medicine) files (all scans are in compliance with the DICOM standard) to the PACS (Picture Archival and Communication System).

This Design document begins with the description of the modules of STOW-RS, followed by its architectural design, and finally the data flow and sequence diagrams to illustrate it.

**NUMBER OF MODULES:**

There are 5 modules in STOW-RS.

**DESCRIPTION OF MODULES:**

1. **Accept STOW requests from third party clients**

The client requests are received through the REST API. All request messages are HTTP/1.1 multipart messages. HTTP Request field Content-Type is used in the header lines by the client in an HTTP/1.1 transaction to indicate the type of data being sent to the Service. The request message can be DICOM or metadata and bulk data depending on the "Content-Type", and is encapsulated in a multipart request body.

1. **Validation of STOW request and DICOM file header**

Firstly, the format of Study Instance ID in the STOW – RS request URL should be validated. Secondly, the Study Instance ID of the STOW- RS request URL should be checked for consistency with the Study Instance ID field of the DICOM file header. Thirdly, the header of the DICOM file, consisting of the 128 bytes of preamble, 4 bytes of prefix, and the data set (comprising of data elements) should be checked.

1. **Ensuring security**

Mechanism for storing a DICOM file will be through the HTTPS protocol, using DICOM UIDs for Study Instance level. For enabling the HTTPS protocol, HTTP BASIC Authorization over SSL and SSL Client Certificates must be used. Authentication and authorization of the client attempting to store DICOM files on the STOW – RS service must be implemented.

1. **Storing the files**

The STOW – RS service creates new resources for the given DICOM instances on the Server or appends them to an existing resource on the Server. One or more DICOM instances associated with one or more study instance unique identifiers (SUID) are stored in the appropriate file store location. The Storage Library is responsible for this action.

1. **Issuing a HTTP Response Message**

The RESTful Service shall return an HTTP status line, including a status code and associated textual phrase for the entire set of stored SOP Instances, followed by a message body containing the Store Instances Response Module.

If the status for all instances included in the POST request is Success, the RESTful Service shall return an "HTTP 200 - Success “response code.

If the status for all instances included in the POST request is Failure, the RESTful Service shall return an appropriate failure status line with a response code. If there are instance specific errors, the response code shall be a 409 and the response payload shall contain the Store Instances Response Module, which contains additional information regarding instance errors.

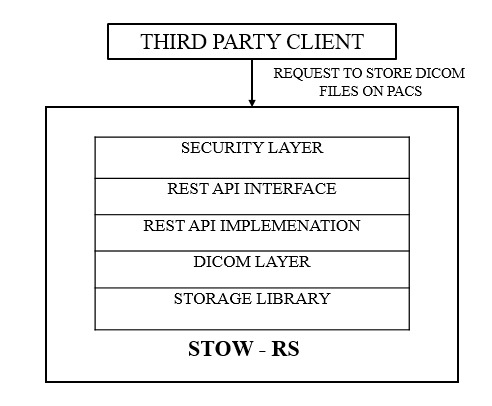
In all other conditions, the RESTful Service shall return an "HTTP 202 - Accepted" response code. The response payload may contain Store Instances Response Module, which specifies additional information regarding instance warnings or failures.

**ALGORITHM DESIGN:**

1. Initialization of DICOM library.
2. Initialization of logging mechanism.
3. Host STOW-RS service on port 443 to listen for incoming STOW-RS requests.
4. A STOW-RS request arrives.
5. Validate format of Study Instance ID from STOW-RS request URL. If format is invalid, abort STOW-RS service.
6. Retrieve Study Instance ID, Series Instance ID, and SOP Instance ID from DICOM file header (inputted by client to store).
7. Store the DICOM file at a predefined location in the hierarchy of the form: StudyUID/SeriesID/SopUID.
8. Construct appropriate HTTP Response Message (status code + status message) and return to client.

**ARCHITECTURAL DESIGN**

A block diagram of STOW-RS service is shown in Figure.1:



**Figure 1: ARCHITECTURE OF STOW-RS**

A brief overview of the architecture of STOW is as follows:

1. The third – party client is the entity external to the healthcare organization that wishes to store DICOM files from the PACS.
2. The Security Layer is responsible for – authenticating, that is, verifying the identity of the external client wanting to store a DICOM file.
3. The REST Interface is the REST API used to enable the client to access STOW – RS service.
4. The REST Implementation layer refers to the actual implementation of the REST API interface.
5. The DICOM Library is used to validate the DICOM information being pushed into PACS by the client.
6. The Storage Library is responsible for the physical storing of DICOM files into the PACS.

The STOW-RS can be decomposed into the following modules:

1. Accept STOW requests from third party clients

The client requests are received through the REST API. All request messages are HTTP/1.1 multipart messages. HTTP Request field Content-Type is used in the header lines by the client in an HTTP/1.1 transaction to indicate the type of data being sent to the Service. The request message can be DICOM or metadata and bulk data depending on the "Content-Type", and is encapsulated in a multipart request body.

1. Validation of STOW request and DICOM file header

Firstly, the format of Study Instance ID in the STOW – RS request URL should be validated. Secondly, the Study Instance ID of the STOW- RS request URL should be checked for consistency with the Study Instance ID field of the DICOM file header. Thirdly, the header of the DICOM file, consisting of the 128 bytes of preamble, 4 bytes of prefix, and the data set (comprising of data elements) should be checked.

1. Ensuring security

Mechanism for storing a DICOM file will be through the HTTPS protocol, using DICOM UIDs for Study Instance level. For enabling the HTTPS protocol, HTTP BASIC Authorization over SSL and SSL Client Certificates must be used. Authentication and authorization of the client attempting to store DICOM files on the STOW – RS service must be implemented.

1. Storing the files

The STOW – RS service creates new resources for the given DICOM instances on the Server or appends them to an existing resource on the Server. One or more DICOM instances associated with one or more study instance unique identifiers (SUID) are stored in the appropriate file store location. The Storage Library is responsible for this action.

1. Issuing a HTTP Response Message

The RESTful Service shall return an HTTP status line, including a status code and associated textual phrase for the entire set of stored SOP Instances, followed by a message body containing the Store Instances Response Module.

If the status for all instances included in the POST request is Success, the RESTful Service shall return an "HTTP 200 - Success “response code.

If the status for all instances included in the POST request is Failure, the RESTful Service shall return an appropriate failure status line with a response code. If there are instance specific errors, the response code shall be a 409 and the response payload shall contain the Store Instances Response Module, which contains additional information regarding instance errors.

In all other conditions, the RESTful Service shall return an "HTTP 202 - Accepted" response code. The response payload may contain Store Instances Response Module, which specifies additional information regarding instance warnings or failures.

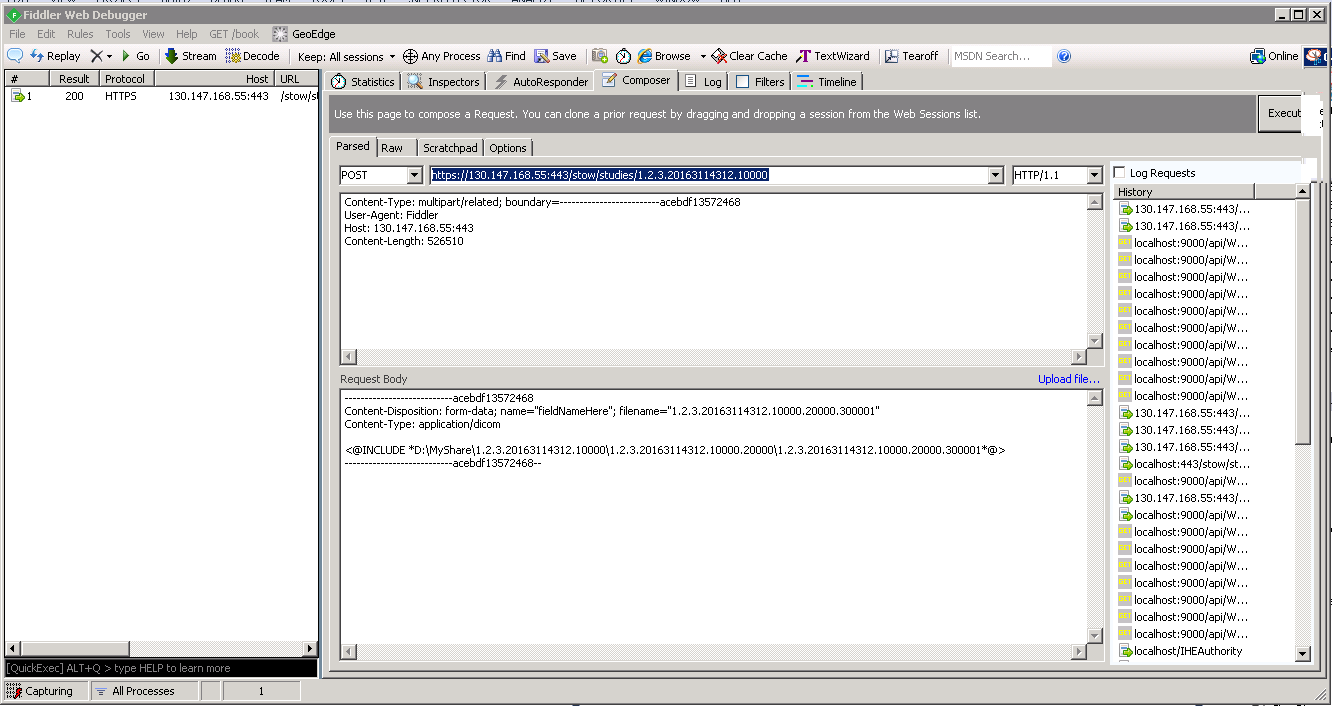
The STOW-RS service is structured in such a way that each of the modules have to be executed in order and successfully to store the DICOM files of the third party clients onto the PACS. The failure of any one module will cause the service to fail for that particular client request. Hence, there is a complete dependency between the modules.

**GRAPHICAL USER INTERFACE**

There is no typical GUI developed, as STOW-RS forms the server side of the web service.

However, during development of STOW-RS, the client accessing the STOW-RS service is simulated/mocked using the Fiddler Tool. The client’s request to store a DICOM file on the STOW server is triggered using the Fiddler Tool. The HTTP response returned by the server to the client is also viewed using Fiddler.

A snapshot of the Fiddler tool on the client side is as follows:



There are 6 fields and 1 control for the client in this tool, and they are:

1. HTTP Method

Here, it is set to POST as STOW-RS allows third party clients to store DICOM files on the STOW server.

1. Request URL

It refers to the URL of the server where the STOW-RS service is hosted.

An example of the request URL would be:

<https://130.147.168.55:443/stow/studies/1.2.3.20163114312.10000>

Here,

* https is used for secure http communication (configured using client and server SSL certificates, and by binding the thumbprint of server certificate to the default HTTPS port of 443).
* “13.147.168.55” refers to the IP address of the server over which the STOW-RS service is hosted.
* “stow” refers to the name of the controller of STOW-RS.
* “1.2.3.20163114312.10000” refers to the Study Instance UID.

1. HTTP Version

It refers to the version of HTTP protocol and set to 1.1, that is, HTTP/1.1.

1. Upload File

This control allows the client to browse and upload the file he wants to post using STOW-RS.

1. HTTP Request Header

It contains the following four fields:

* Content type – which specifies the type of content the client is sending to server. It has to be set to multipart/related.
* Boundary – to distinguish between the multiple parts of HTTP request message. Each part contains a DICOM instance.
* Host – the IP address and the port number of the STOW-RS server.
* Content Length – This field indicates the number of bytes of data the client is uploading using STOW-RS.

1. HTTP Request Body

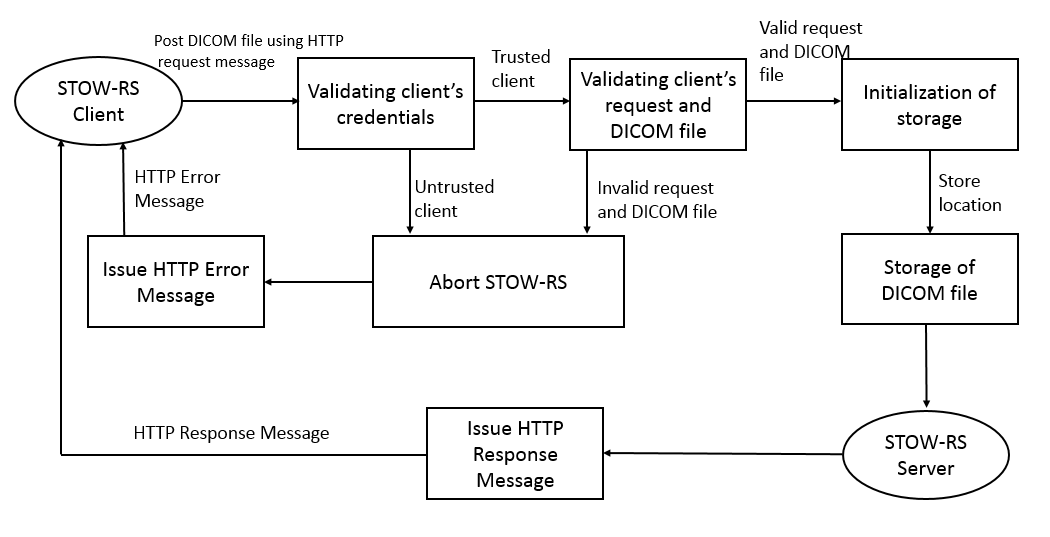
The contents of the HTTP Request are enclosed inside a pair of boundaries (defined in HTTP Request Header). The various fields of the HTTP Request Body are:

* The Content-Disposition response-header field is used as a means for the server to suggest a default filename if the client requests that the content is saved to a file.
* The content type is set to application/dicom to signal that the client is sending DICOM instances to the server. It can be application/dicom+xml or application/json depending on the data being sent.

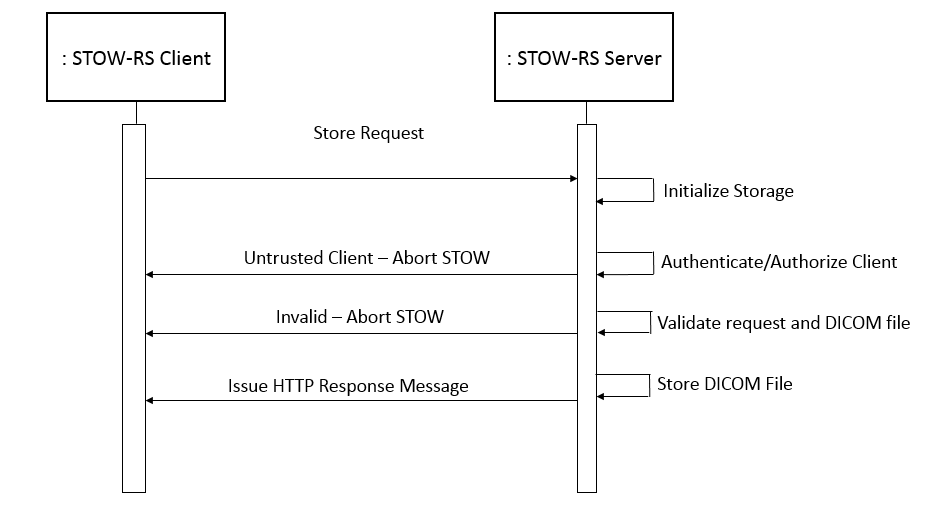
7). Execute

This allows the client to send the STOW-RS request to the server.

**DATA FLOW DIAGRAM**



**SEQUENCE DIAGRAM**



**REFERENCE**

Digital Imaging and Communication (DICOM) standard 2015 published by National Electrical Manufacturers Association.