**M S Ramaiah Institute of Technology**

(An Autonomous Institute, Affiliated to VTU)

MSR Nagar, MSRIT post, Bangalore-54.

A **Software Requirement Specification (SRS)** 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**

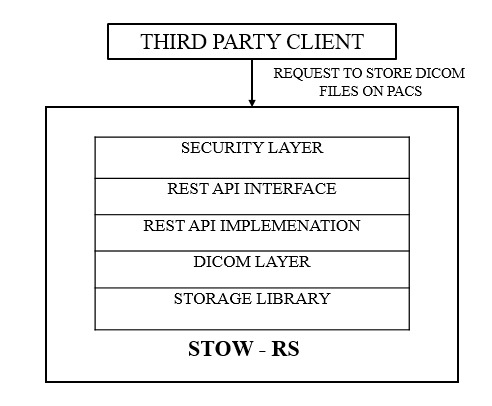
[www.msrit.edu](http://www.msrit.edu), **Feb 2016**

**SOFTWARE REQUIREMENT SPECIFICATION (SRS)**

**1. PRODUCT OVERVIEW**

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).

A block diagram of the 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:

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

This project aims to build a web-based service for accessing and presenting DICOM supported as per the DICOM. (Defined by National Electrical Manufacturers Association (NEMA)).

**2. EXTERNAL INTERFACE REQUIREMENTS**

1. **User Interface**

The STOW-RS service can be accessed by the third party clients using a simple REST API. A RESTful Web service is a Web service implemented using REST architecture and HTTP.

1. **Hardware Interface**

This information is proprietary to Philips. Hence, it cannot be disclosed.

1. **Software Interface**

This information is proprietary to Philips. Hence, it cannot be disclosed.

1. **Communication Interface**

Communication between the third party client and the STOW-RS service occurs by HTTP messages.

The DICOM file to be stored is encapsulated into a HTTP multipart request message and sent across to the server (STOW – RS service). All request messages are HTTP/1.1 multipart messages. For DICOM instances being sent by the client, the HTTP Request field Content-Type is set as: **Content-Type: application/dicom**. The DICOM Request Message has a multipart body, where: **Content-Type: multipart/related; type=application/dicom; boundary = {Message Boundary}.** The multipart request body contains every instance to be stored. Each instance is in a separate part of the multipart body. Each part in the multipart body represents a DICOM SOP Instance with the following HTTP headers: Content-Type: application/dicom.

The STOW RESTful Service will return a 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 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 a Store Instances Response Module, which specifies additional information regarding instance warnings or failures.

**3. FUNCTIONAL REQUIREMENTS**

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

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.

The STOW - RS service must be able to process storage requests with the following HTTP header format:

* Resource

{SERVICE}/studies[/{StudyInstanceUID}]

Where,

{SERVICE} is the base URL for the service. This may be a combination of scheme (either HTTP or HTTPS), host, port, and application;

{StudyInstanceUID} (Optional) is the study instance UID for a single study. If not specified, instances can be from multiple studies. If specified, all instances shall be from that study, and instances not matching the StudyInstanceUID shall be rejected.

* Method

POST

* Headers

Content-Type - The representation scheme being posted to the RESTful service. The types allowed for this request header are as follows:

* Multipart/related; type=application/dicom; boundary = {MessageBoundary}
* Multipart/related; type=application/dicom+xml; boundary = {MessageBoundary}.
* The DICOM Request Message has a multipart body.
* Content-Type:

Multipart/related; type=application/dicom; boundary = {MessageBoundary}.

Specifies that the post is XML metadata and bulk data.

* The multipart request body contains every instance to be stored. Each instance is in a separate part of the multipart body.
* Each part in the multipart body represents a DICOM SOP Instance with the following HTTP headers:
* Content-Type: application/dicom.
* The XML Metadata and Bulk Data Request Message has a multipart body.
* Content-Type:

Multipart/related; type=application/dicom+xml; boundary = {MessageBoundary}

* The multipart request body contains all the metadata and bulk data to be stored.
* If the number of bulk data parts does not correspond to the number of unique BulkDataURIs in the metadata then the entire message is invalid and will generate an error status line.
* Each body part is either DICOM XML metadata or a bulk data item from a SOP Instance sent as part of the Store operation.

The first part of the multipart message must be XML metadata and contains the following HTTP headers:

* Content-Type: application/dicom+xml; transfer-syntax = {TransferSyntaxUID}

Subsequent items will contain the following HTTP headers (order is not guaranteed):

* Content-Type: application/dicom+xml; transfer-syntax={TransferSyntaxUID}

An uncompressed bulk data element encoded in Little Endian binary format with the following headers:

* Content-Type: application/octet-stream

Content-Location: {BulkDataURI}

A compressed pixel data object from a SOP Instance in the Study with the following headers:

* Content-Type: {MediaType}

Content-Location: {BulkDataURI}

Metadata and its associated bulk data shall always be sent in the same POST request.

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.

1. **Issuing a HTTP Response Message**

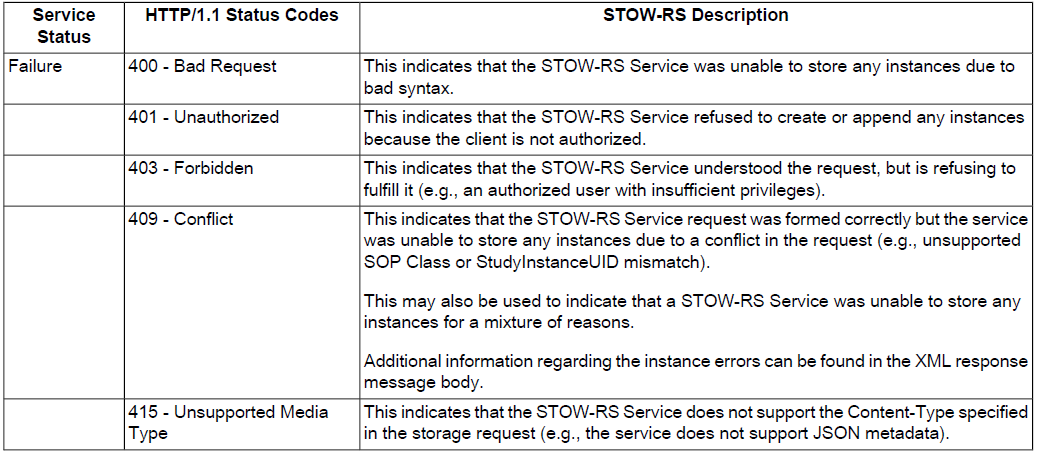
The RESTful Service shall return an HTTP status line, including a status code and associated textual phrase for the entire set of stored SOP Instances, followed by a message body containing the Store Instances Response Module.

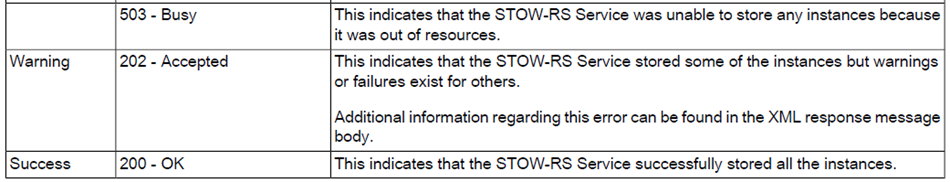
If the status for all instances included in the POST request is Success, the RESTful Service shall return an "HTTP 200 - Success “response code.

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 standard HTTP response codes are listed in Table .1:

**Table 1: HTTP/1.1 STANDARD RESPONSE CODE**





**4. SOFTWARE SYSTEM ATTRIBUTES**

1. **Reliability**

The STOW – RS service must create new resources or append to existing resources the incoming DICOM instances. Storing of the same instance should not occur multiple times. The DICOM files must be accepted only from an authorized and an authenticated client. The request URL and the DICOM file header must by validated. The adequate HTTP response message must be returned to the client after completion of the STOW operation. All operations must be logged for further reference. The service must meet all of the functional requirements without any unexpected behavior. It should be tested and debugged completely. All exceptions should be well handled.

1. **Availability**

The STOW – RS service should always be up and running. An authorized client must be able to post DICOM files using STOW – RS service at any time instance.

1. **Security**

Storing of DICOM files will be through HTTPS protocol, using DICOM UIDs for Study Instance level. The client must be authorized and authenticated before he is allowed to post a DICOM file using the STOW – RS service. It should also not be possible for the DICOM files of third party clients stored to be accessed by unauthorized personnel’s within our organization.

1. **Portability**

The STOW-RS service should be able to function upon any CPU configuration.

1. **Maintainability**

The service should be written clearly, concisely, and optimally. The code must be well documented with explanatory comment lines. Modularity of code has to be ensured for easy and accelerated version upgrades and changes.

1. **Performance**

The STOW – RS service must run on Windows Server for best performance. The entire sequence of operations – receiving of DICOM files from clients, validation, security check, storing files, logging, and sending back an appropriate HTTP response message should be completed within a few milliseconds.

**5. PERFORMANCE REQUIREMENT**

Performance benchmarking is yet to be done**.** However, the service is required to function within very few milliseconds.

**6. DATABASE REQUIREMENT**

Not applicable. We will not be using a database.

**7. DESIGN CONSTRAINTS**

Some of the dll’s used are in 32-bit, and hence. They have to be converted to 64-bit.

**8. OTHER REQUIREMENTS**

**Logging**

All errors, warnings, exceptions, and informational messages have to be logged in a custom location for further reference.

**Auditing**

A security-relevant chronological record that provides documentary evidence of the sequence of activities that have occurred needs to be maintained. This record, called the audit trail can be used to for reexamination of events occurred and for security purposes.