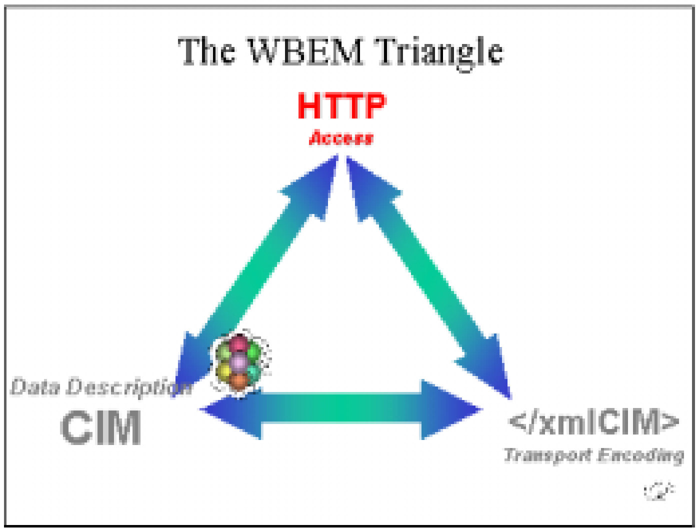
Web Server Interface

Design Specification

Version 1.0



Authors

Warren Otsuka
Bapu Patil

Hewlett-Packard Company

Preface

This is version 1.0 of "Web Server Interface Design Specification Document" for PEGASUS CIM SERVER implementation.

If you have any questions or comments regarding this document, please contact

Warren Otsuka	Email: warren_otsuka@hp.com	(408) 447-5287
Bapu Patil	Email: bapu_patil@hp.com	(408) 447-3209

Web Server Interface Design

Overview

The current PEGASUS CIM Server implementation provides communication between a CIM client and CIM Server via a socket connection using the default port 5988. The CIM Server accepts connections and process xmlCIM requests. In some cases, we do not want the Clients to directly connect to port 5988 because of firewall restrictions and CIM Clients will not be able to connect to port 5988. In that case we want to allow CIM client to go through port 80, i.e. through a web server.

For the PEGASUS CIM Server to coexist with a web server there must be a means for CIM requests to be forwarded from a web server to the CIM Server for processing. This is where we need an interface that will take requests forwarded by the Web Server, process them and send the requests to CIM Server.

Design Objective

This design provides a way for CIM client applications to access CIM information regardless of whether a web server is running on the system. The design provides access to the CIM Server via CIM Servlet using the Java Servlet Specification 2.2. CIM clients who wish to communicate with CIM Server running in conjunction with a web server may send XmlCIM data over HTTP to the CIM Servlet. The CIM SERVER will run either as a standalone HTTP server as it currently does or it can be configured to accept xmlCIM requests from a web server. Both modes can not be supported at the same time.

It is assumed that the web server will be on the same system as CIM Server. Since the web server will be listening on port 80, the CIM Server can not use the port 80. In case CIM Server is configured to use port 80, it will get a networking error when it tries to bind to port 80. In order for the CIM Server and the web server to run concurrently the default CIM Server port number must be changed to a port other than 80.

Architecture

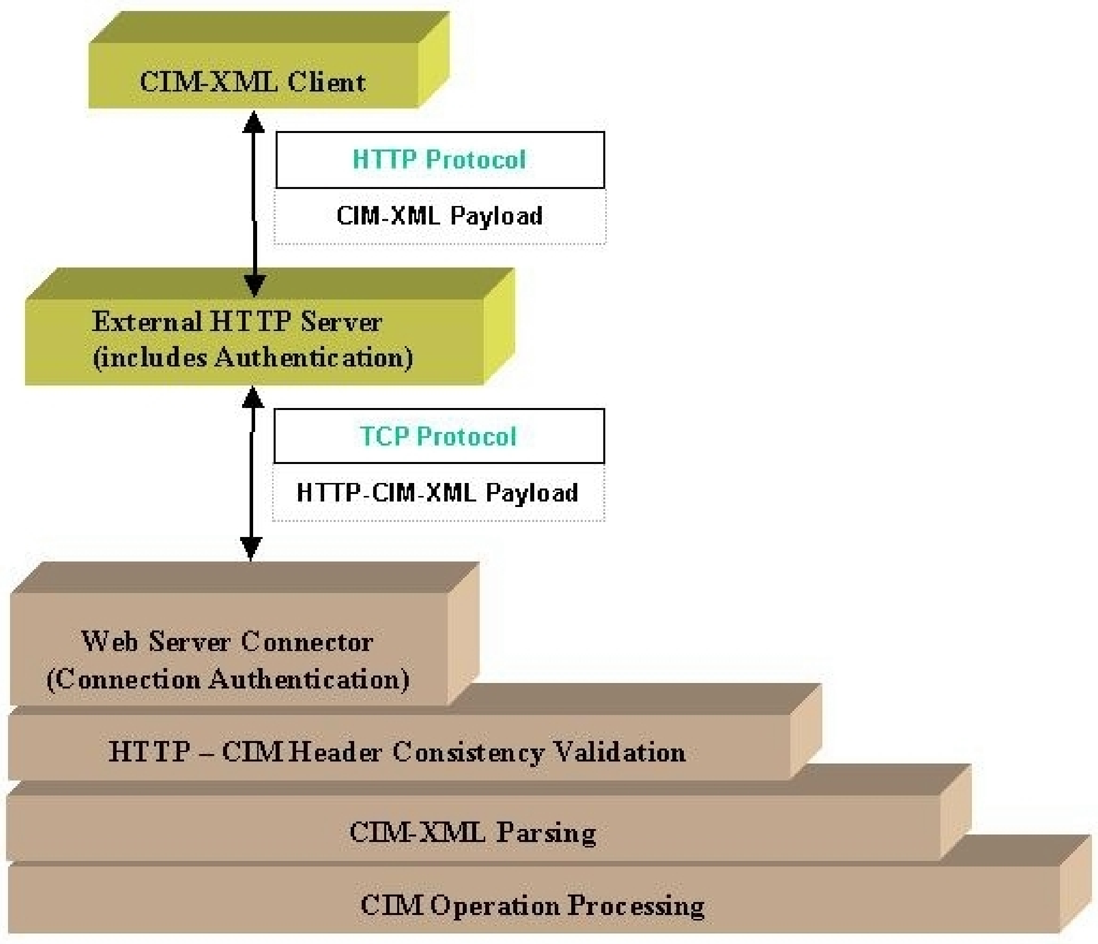


Figure 1: Proposed Architecture

Figure 1 shows the general flow of request through a Web server running on the same where CIM Server is running. In this CIM Clients request can go through web server and then to CIM Server.

Figure 2 shows more details to previous figure. Figure 2 includes a Servlet that will process CIM Requests that came through Web Server and then forwarded to CIM Server.

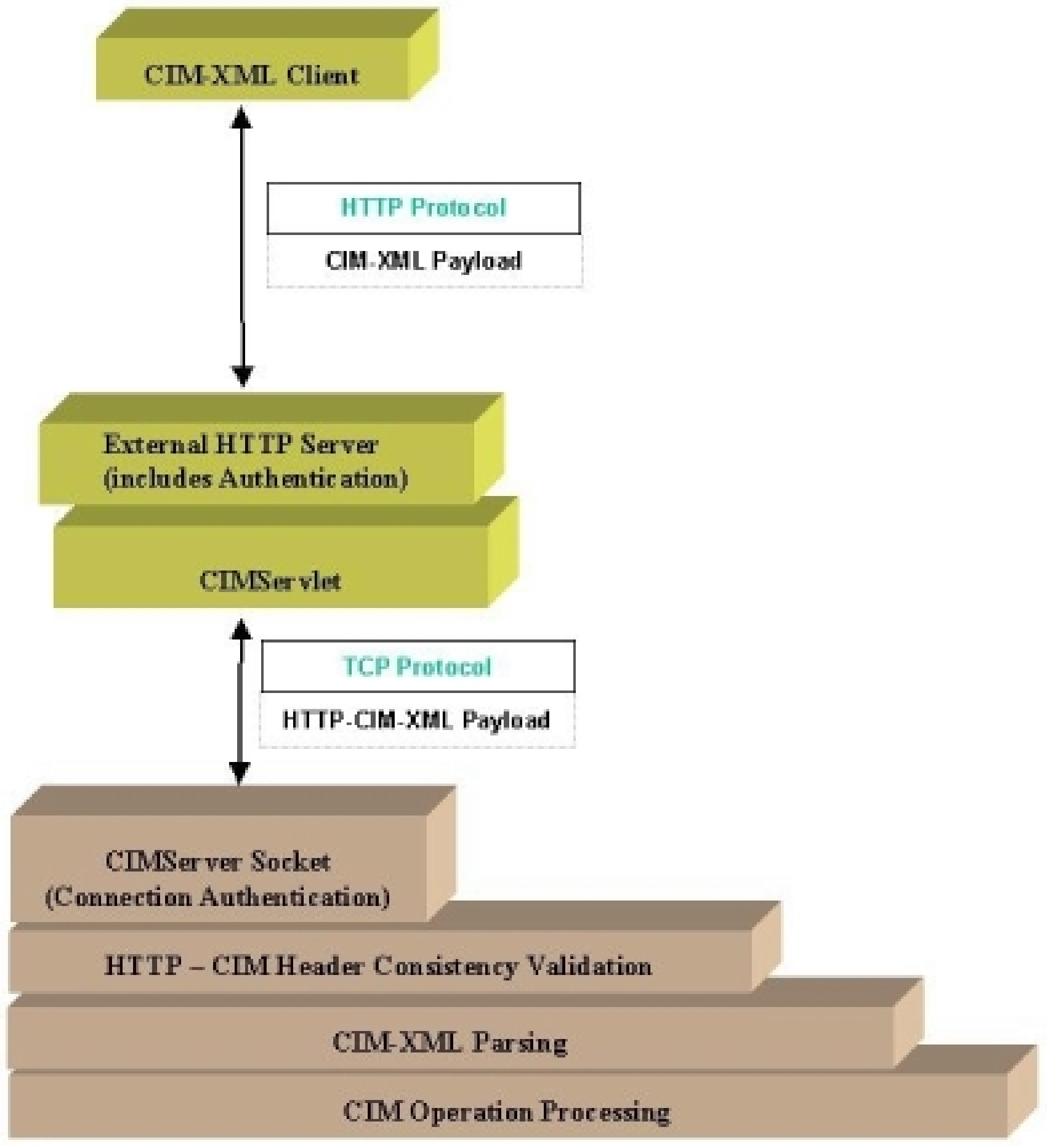


Figure 2: Web Server Request Flow

CIM Servlet provides an interface to CIM Client applications to access CIM information via Web Server. That is, it Provide access to CIMOM via a Web Server. CIM clients those wishes to communicate with CIMOM will send XmlCIM data over HTTP interface and these requests are forwarded to CIM Servlet by the Web Server and CIM Servlet parses CIM client request and extract the HTTP headers and XmlCIM payload. Using this HTTP request information, it then reconstructs the HTTP headers and passes the headers and the xmlCIM payload to CIM Server over a socket connection.

CIM Servlet Design

CIM Servlet interface architecture will include the implementations for doPost() and doOptions() method calls of HTTP Servlet interface. The standard Java Servlet APIs such as doPost(), and doOptions () method calls will be overridden by CIM Servlet. The CIM Servlet will also include (override) init() method to initialize the servlet and any other required interfaces such as reading configuration files etc.. The CIM Servlet will connect to CIM Server using Socket interface.

CIMServlet design will include Connection Pool (a Java Class) which will maintain list of socket connection to CIM Server. For each of the HTTP operation calls (doOptions() and doPost()) that are requested to CIM Servlet will borrow a connection from the Connection Pool and use the socket connection to process the request. Once the requesting processing done and response is sent back to the client Servlet will return the Socket connection back to the Pool.

For every request, CIM Servlet will perform:

- Borrow a socket connection from the Connection Pool.
- Reads the HTTP requests input data sent by the CIM clients.
- Write the request data over the socket.
- Read the socket data (response) from CIM Server.
- Form the HTTP response and send it to CIM Clients.
- Return the connection back to the Pool.

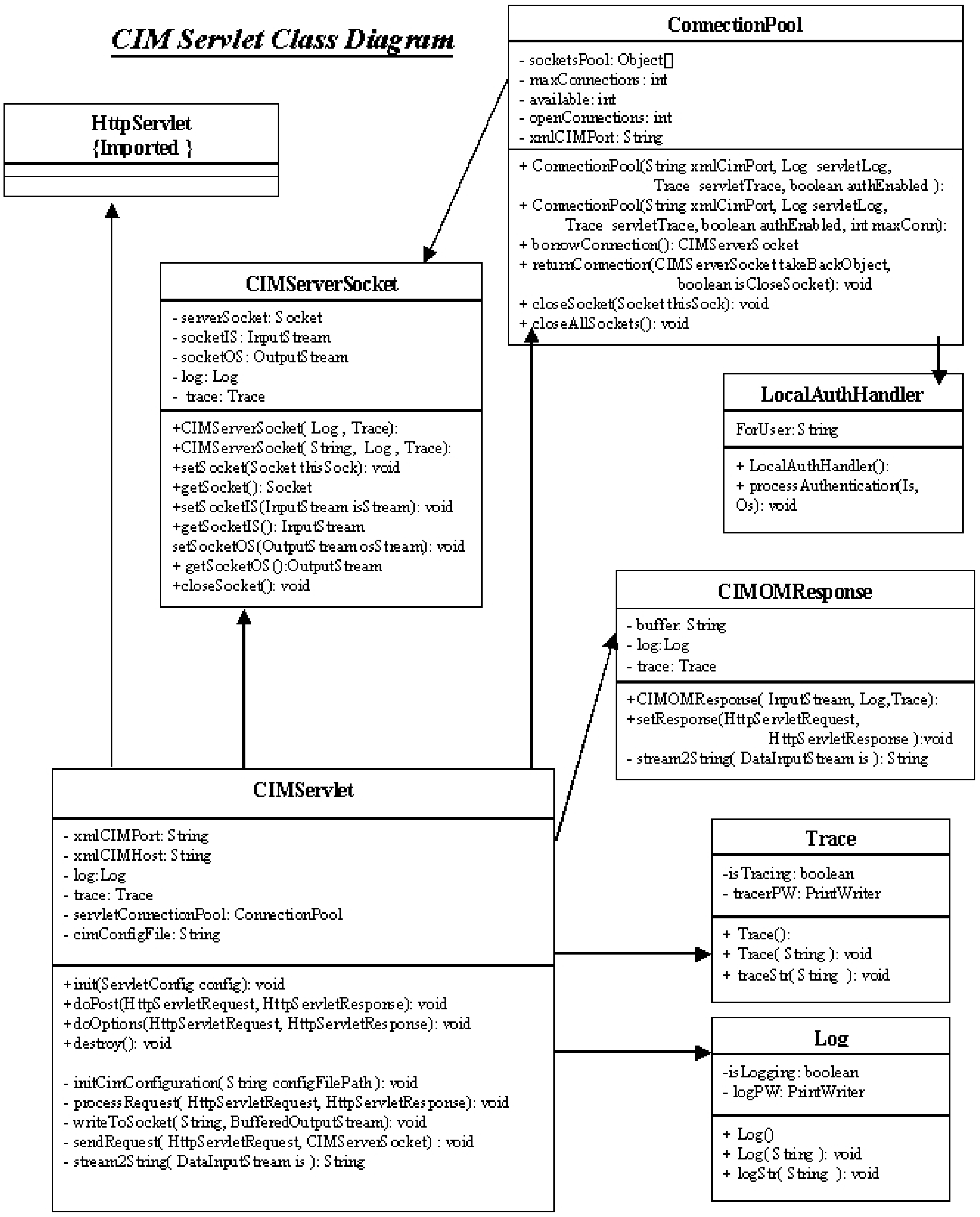

CIM Servlet Class Diagram
HttpServlet
{Imported }
ConnectionPool
- socketsPool: Object[]
- maxConnections : int
- available: int
- openConnections: int
- xmlCIMPort: String
+ ConnectionPool(String xmlCimPort, Log servletLog, Trace servletTrace, boolean authEnabled):
+ ConnectionPool(String xmlCimPort, Log servletLog, Trace servletTrace, boolean authEnabled, int maxConn):
+ borrowConnection(): CIMServerSocket
+ returnConnection(CIMServerSocket takeBackObject, boolean isCloseSocket): void
+ closeSocket(Socket thisSock): void
+ closeAllSockets(): void
CIMServerSocket
- serverSocket: Socket
- socketIS: InputStream
- socketOS: OutputStream
- log: Log
- trace: Trace
+CIMServerSocket(Log , Trace):
+CIMServerSocket(String, Log , Trace):
+setSocket(Socket thisSock): void
+getSocket(): Socket
+setSocketIS(InputStream isStream): void
+getSocketIS(): InputStream
setSocketOS(OutputStream osStream): void
+ getSocketOS ():OutputStream
+closeSocket(): void
LocalAuthHandler
ForUser: String
+ LocalAuthHandler():
+ process Authentication(Is, Os): void
CIMOMResponse
- buffer: String
- log:Log
- trace: Trace
+CIMOMResponse(InputStream, Log,Trace):
+setResponse(HttpServletRequest, HttpServletResponse):void
- stream2String(DataInputStream is): String
CIMServlet
- xmlCIMPort: String
- xmlCIMHost: String
- log:Log
- trace: Trace
- servletConnectionPool: ConnectionPool
- cimConfigFile: String
+init(ServletConfig config): void
+doPost(HttpServletRequest, HttpServletResponse): void
+doOptions(HttpServletRequest, HttpServletResponse): void
+destroy(): void
- initCimConfiguration(String configFilePath): void
- processRequest(HttpServletRequest, HttpServletResponse): void
- writeToSocket(String, BufferedOutputStream): void
- sendRequest(HttpServletRequest, CIMServerSocket) : void
- stream2String(DataInputStream is): String
Trace
-isTracing: boolean
- tracerPW: PrintWriter
+ Trace():
+ Trace(String): void
+ traceStr(String): void
Log
-isLogging: boolean
- logPW: PrintWriter
+ Log()
+ Log(String): void
+ logStr(String): void

Fig 3: CIMServlet Class Diagram

CIM Servlet Interfaces:

init():

Depending on the Web Server, the Servlet init() is called in one of the following stages:

- When the Web Server starts.
- Just before the first service (doPost, doGet etc) request of CIM Servlet.
- The request of a Web Server Administrator.

During the initialization, Servlet will read Configuration file to set the values for Log, Trace, Port, etc.

doPost():

This gets called when the client sends HTTP "POST" request. For every HTTP request that come into doPost() of the CIMServlet, this does the following:

- Borrow a socket connection from the Connection Pool.
- Reads the HTTP requests input data sent by the CIM clients.
- Write the request data over the socket.
- Read the socket data (response) from CIM Server.
- Form the HTTP response and send it to CIM Clients.
- Return the connection back to the Pool.

```
public void doPost( HttpServletRequest req,   HttpServletResponse resp)
                        throws ServletException, IOException
{
        // Process the HTTP input
        processRequest(req, resp);
}
```

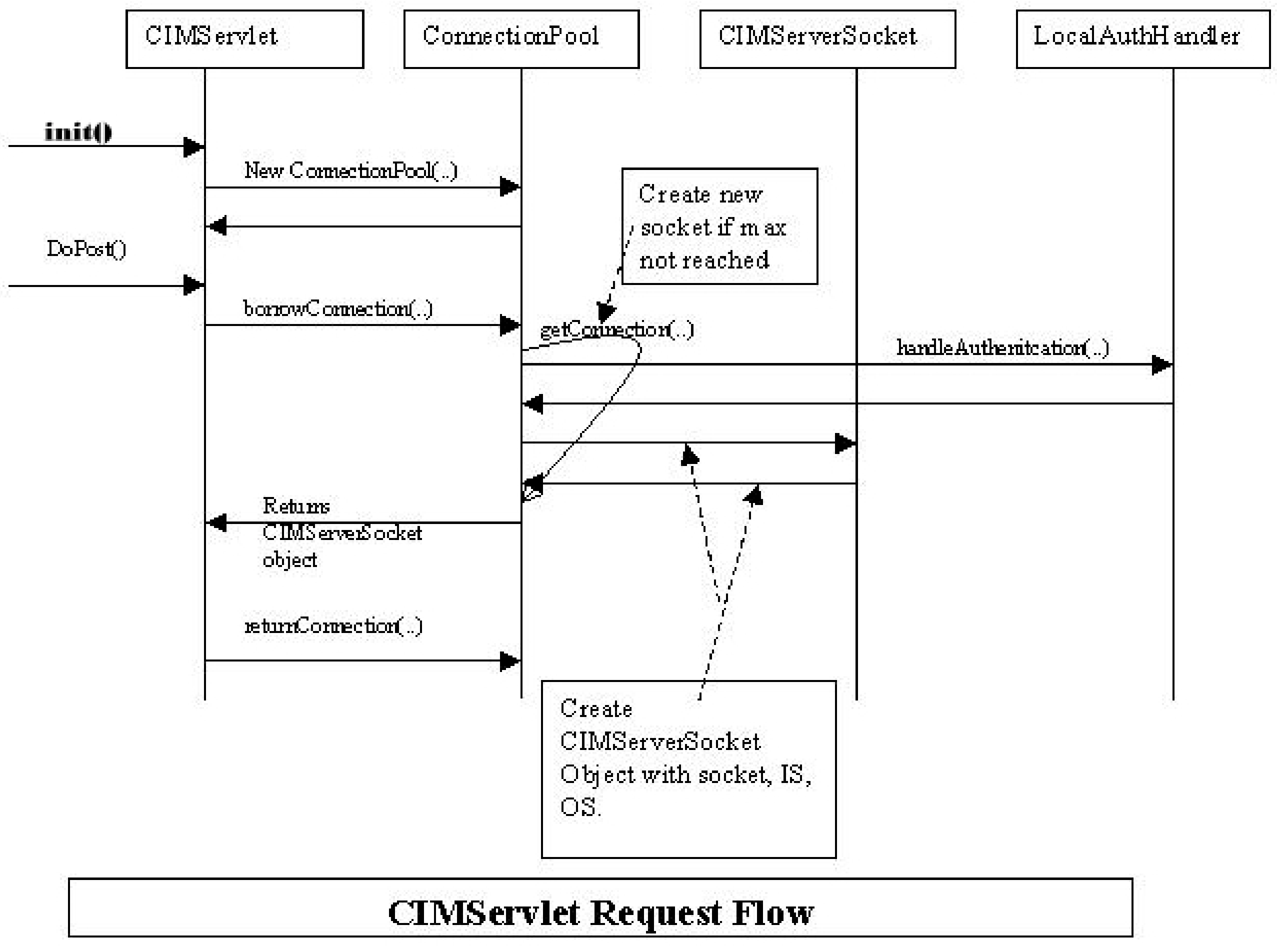


Fig 4: doPost() Method Sequence Diagram

doOptions():

doOptions() method returns features supported by CIM Server. This gets called when the client sends HTTP "OPTIONS" request. It passes the request to CIM Server.

For every HTTP request that come into doOptions() of the CIMServlet, this does the following:
For every request, CIM Servlet will perform:

- Borrow a socket connection from the Connection Pool.
- Reads the HTTP requests input data sent by the CIM clients.
- Write the request data over the socket.
- Read the socket data (response) from CIM Server.
- Form the HTTP response and send it to CIM Clients.
- Return the connection back to the Pool.

```
public void doOptions( HttpServletRequest req,   HttpServletResponse resp)
                        throws ServletException, IOException
{

          // Process the HTTP input
          processRequest(req, resp);

}
```

CIM Servlet Access Security:

HTTP Protocol provides built-in authentication support – based on a simple challenge/response, username/password model. One could have either Basic authentication or Digest authentication or could have both depending on what the client application supports.

Authentication by Web Servers

Web Server can be setup in such way that accesses to certain resources (files, directories, servlets, etc.) can be restricted. In such a case Web Server maintains a database of usernames and passwords and identifies certain resources as protected. When a user requests access to a restricted resource, the server responds with a challenge asking username and password. At this time users enter a username and password. This is again sent back to the web server. If the submitted username and password match the information in the server's database, access is granted. The server handles the whole authentication process itself.

Pros and Cons

- Web Server has to maintain user information. This makes WBEM Admin to work with Web Server admin in order to add/delete users. There may be additional other Web Server users in password. This way WBEM could better have its own password file and easy to maintain, no need to depend on Web Server.

Dependencies:

M-POST method call support:

- As of now, most Web Servers don't support HTTP M-POST method calls. The CIM Servlet depends on Web Server (such as Apache) and Servlet Engine (such as Tomcat) to support HTTP M-POST method calls to handle M-POST calls. Since Web Server and Servlet Engines do not handle the call, this version of CIM Servlet will not support M-POST call as well. As the Web Servers support this feature in the future, the CIM Servlet will include the implementation. If CIM Clients make M-POST call, the invocation will fail in case Web Server not supporting and Web Server will return "501 Not Implemented". In such a case, CIM Clients should retry using POST with appropriate modifications. This is again part of CIM over HTTP specification.