Genesys Mobile Services Deployment Guide

Security and Access Control

12/18/2025

Contents

Security and Access Control

This page discusses deployment topology and advanced configuration that will allow you to secure your Genesys Mobile Services solution.

Note: Although some load balancing considerations are discussed on this page with respect to solution architecture, configuration recommendations are provided on the Configuring Apache Load Balancer page.

Overview of Security, Access Control, and Load Balancing

Genesys Mobile Services makes contact center functionality accessible through a set of REST- and CometD-based APIs. Since these APIs can be used both by clients residing inside and outside the enterprise network, it is important to understand how to protect data that is travelling between solution components. The Genesys Mobile Services solution is designed to work with your existing security infrastructure, relying on third-party components (security proxies) to provide encryption and authorization capabilities.

Deployment requirements

Genesys Mobile Services should always be deployed behind an HTTP security gateway (proxy) performing:

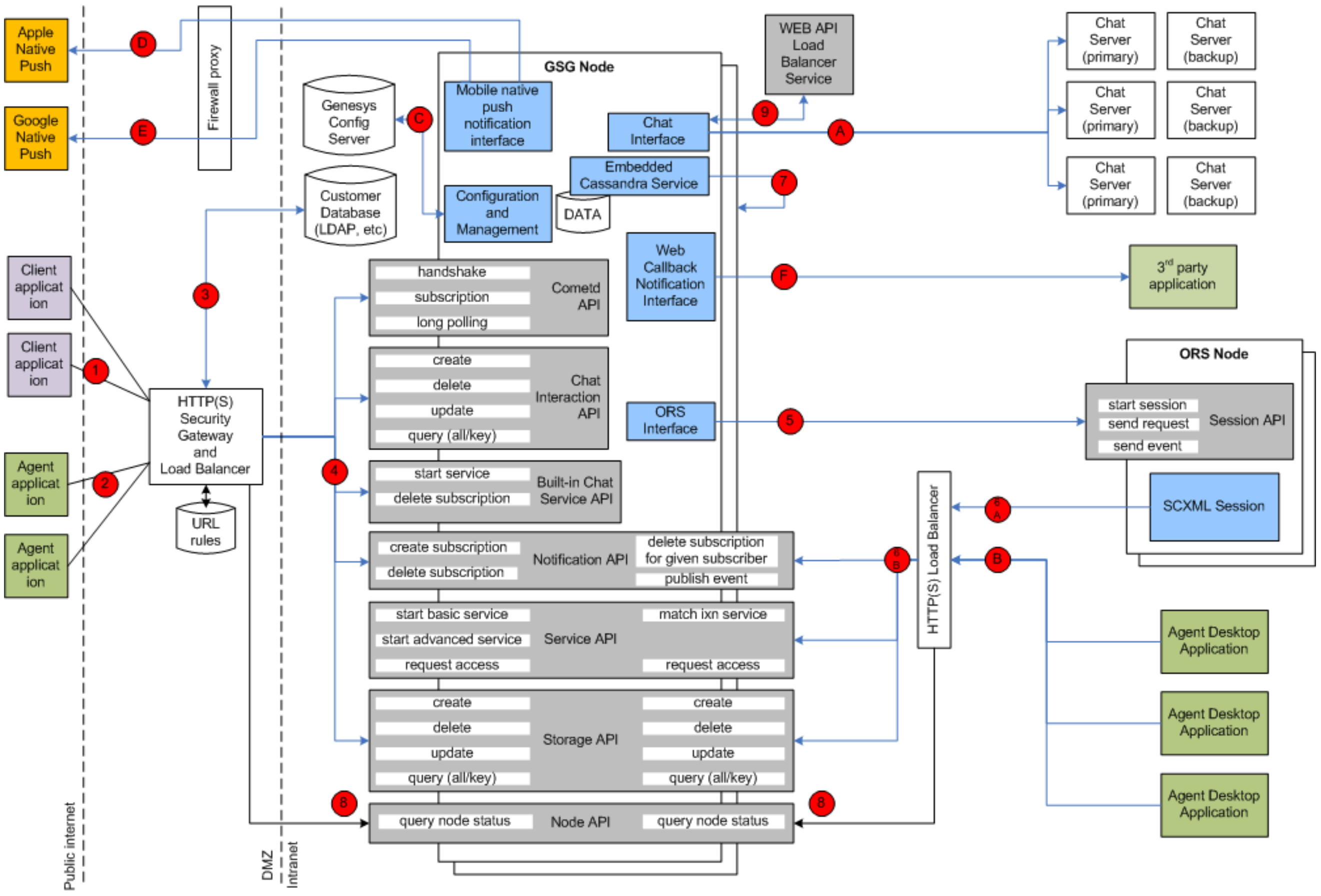
- client authentication (optional)
- TCP port and URL access control
- load balancing functionality, distributing the load between multiple Genesys Mobile Services nodes responsible for processing API requests
- HTTP connection encryption (SSL)

In addition, the HTTP security gateway (proxy) could perform following functions:

- protect against denial of service (DoS) attacks
- authenticate requests using HTTP Basic/Digest, oAuth or other authentication/authorization protocol
- manage client access using IP-based access control
- rate limit API traffic using quotas
- inspect packets for threats and sensitive data

Genesys Mobile Services Deployment Topology

The following image shows architecture and communication links between different components in a typical Genesys Mobile Services solution. A table with recommendations on how to secure these connections follows immediately after.



List of Connections Utilized by a Typical Genesys Mobile Services Deployment

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
Security Gateway and Load Balancer in Front of Genesys Mobile Services Nodes (Customer Client Side)	(1)	User/ customer facing application inside or outside of the corporate firewall: mobile, web or ... based application	Cometd based notification, Chat, Service and Storage APIs exposed through Genesys Mobile Services component	HTTP/ HTTPS	Any deployment-specific port convenient for client applications. For example: 80. See third party gateway/ proxy documentation if you need to	Client is typically a hard-coded server port as part of the API access URL	Client applications should access Genesys Mobile Services APIs through SSL-protected HTTP connections with optional basic/

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
					change/ configure this port. See also GMS->Options->/server/ external_url_base configuration option in Configuration Manager		digest/ oAuth/... client authentication. If a client application has no access to user credentials, then anonymous access is supported but will result in a lower level of security. Clients should be blocked from accessing certain URLs of the API according to the access rules below.
Security Gateway and Load Balancer in Front of Genesys Mobile Services Nodes (Agent Client Side)	(2)	Agent-facing application inside or outside of the corporate firewall: mobile, web or ... based application	CometD-based notification, Chat, Service and Storage APIs exposed through Genesys Mobile Services components	HTTP/ HTTPS	Any deployment-specific port convenient for client applications. For example: 81 See also *GMS/server/ external_url_base* configuration option in Configuration Manager. **Note:** Use a different port from connection (1).	Client is typically a hard-coded server port as part of the API access URL.	Agent applications residing outside of the enterprise intranet can also access Genesys Mobile Services APIs through an SSL-protected HTTP connection. Agent authentication can be performed

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							against Configuration Server, or other service such as LDAP, by the security gateway. Deployments with lower security requirements can allow API access without agent authentication, relying only on uniqueness of the service ID supplied by the application. In this case, it is assumed that only trusted applications would gain access to service ID. Agent authentication is encouraged especially if used outside of the corporate network. Clients should be blocked from accessing certain URLs of the API

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							according to the access rules below.
Enterprise Authentication Server (LDAP, etc)	(3)	Security gateway and load balancer in front of Genesys Mobile Services nodes (client side)	API user client authentication	HTTP/ HTTPS or deployment specific	Deployment-specific port. See third party documentation.	Port is hard coded as part of the security gateway configuration.	Use an appropriate level of security, as required by the Enterprise authentication server.
Two or More Genesys Mobile Services Nodes	(4)	Security gateway and load balancer in front of Genesys Mobile Services nodes (client side)	CometD-based notification, Chat, Service, Storage APIs	HTTP/ HTTPS	Default: 8080. Configured inside *<Genesys Mobile Services deployment directory>/launcher.xml: <parameter name="http_port" displayName="jetty.port" mandatory="false">* See also the *GMS->Options->/server/ external_url_base* configuration option in Configuration Manager.	Port is hard coded as part of the security gateway configuration.	Jetty container hosting Genesys Mobile Services can be configured to accept SSL-protected connections from security and load balancing gateway. Mutual authentication can also be enabled if required. See below for more information about how to configure SSL connector in Jetty. HTTP basic authentication for security and load

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							balancing gateway can also be configured with the help of Genesys Professional Services. See Restricting Ports for more information about Port Number control.
Two or More ORS Nodes	(5)	Two or more Genesys Mobile Services nodes	ORS scxml session start, stop, send event, etc	HTTP	Default: 7210. Configured in Configuration Manager, inside *<Genesys Mobile Services deployment directory>/launcher.xml: <parameter name="http_port" displayName="jetty.port" mandatory="false"/>.*	Each Genesys Mobile Services node should have a connection configured in Configuration Manager to each ORS application used by the Genesys Mobile Services solution. See *Applications->Genesys Mobile Services->Connections->ORS(1..n)->Server Info->Host and Listening Ports->http.*	
Security	(6A)	ORS node	Invoking	HTTP	Deployment	Hard	Security

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
and Load Balancing Gateway in Front of Genesys Mobile Services Nodes		running SCXML session	Genesys Mobile Services APIs: Service (match-interaction), Notification and Storage APIs.		specific. See 3rd party documentation	coded as part of the Genesys Mobile Services API URL inside SCXML session. Could be configured in Configuration Manager and read be SCXML session	proxy should be configured to block access to the API URLs according to the access rules below.
Two or More Genesys Mobile Services Nodes	(6B)	Security and load balancing gateway in front of Genesys Mobile Services nodes	Invoking Genesys Mobile Services APIs: Service (match-interaction), Notification and Storage APIs.	HTTP	Default: 8080. Configured in Configuration Manager, inside *<Genesys Mobile Services deployment directory>/launcher.xml: <parameter name="http_port" displayName="jetty.port" mandatory="false">*	See 3rd party security and load balancing gateway configuration for URL and port mapping	Jetty container hosting Genesys Mobile Services could be configured to accept SSL protected connections from security and load balancing gateway. Mutual authentication could also be enabled if required. See below for more information about how to configure SSL connector in Jetty. HTTP basic authentication for security

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							and load balancing gateway can also be configured with the help of Genesys Professional Services. See Restricting Ports for more information about Port Number control.
Cassandra Instance Embedded into Genesys Mobile Services Node	(7)	Local and remote Genesys Mobile Services nodes accessing distributed Cassandra storage	Cassandra client API	TCP/IP	Defaults: • rpc_port: 9159 • storage_port: 6934 • ssl_storage_port: 6935 • JMX port: 9192 Configuration: see *<Genesys Mobile Services install dir>/etc/ cassandra.yaml*. Check the following parameters: listening, encryption_options. Enable inter-node encryption to protect data travelling between Genesys Mobile Services nodes. see *<Genesys Mobile*	Uses local connection to embedded Cassandra instance.	Most of the transient service session related data is stored in Cassandra database that uses memory and file system of the Genesys Mobile Services node. See *<Genesys Mobile Services install directory>/data* folder. Files there should be protected from unauthorized access. Access to Cassandra ports used

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
					Services install dir>/launcher.xml for JMX configuration.		for synchronization between Genesys Mobile Services nodes and client access should only be allowed from Genesys Mobile Services nodes. Enable internode encryption in Cassandra configuration. More about Cassandra encryption below.
Genesys Mobile Services Server Node	(8)	Security and load balancing gateway in front of the two or more Genesys Mobile Services nodes	Genesys Mobile Services Node API - health check performed by load balancing gateway to improve switchover in case of a Genesys Mobile Services node failure	HTTP	Default port: 8080. Configured inside *<Genesys Mobile Services deployment directory>/launcher.xml: <parameter name="http_port" displayName="jetty.port" mandatory="false">* See also *GMS->Options->/server/external_url_base* configuration option in Configuration Manager.	Hard coded in security and load balancing gateway configuration. See third party documentation for information how to change it.	No special security is required for this connection. Read-only operation. See Restricting Ports for more information about Port Number control.
Genesys WEB API Server	(9)	Genesys Mobile Services	Chat server load	HTTP	Default port: 9002. Configured	Read dynamically from	No special security is required

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
Node		node	balancing and high availability API		in Configuration Manager. See GMS->Connections->WEB API Server->Server Info->Listening Ports->default port	Configuration Manager configuration: GMS->Connections->WEB API Server. See also GMS->Options->chat/ chat_load_balancer_url_path in Configuration Manager.	for this connection. Read only operation.
Genesys Chat Server - N+1 Primary/ Backup Pairs	(A)	Genesys Mobile Services node	FlexChat protocol	TCP/IP, TLS encrypted if required	Default port: 4856. Configured in Configuration Manager. See *GMS->Connections->WEB API Server->Connections->Chat Server (primary)->Server Info->Listening Ports->webapi port*.	Read dynamically before each chat API call (refresh/ connect/ etc). See: *GMS->Options->chat/ chat_load_balancer_url_path and GMS->Connections->Web API Server* in Configuration Manager. ⚠ You must confirm TLS is supported and configured for this connection.	Enable TLS encryption if a higher level of protection is required. This is a Platform SDK-based connection supporting standard Genesys security.
Security and Load Balancing Gateway in Front of Genesys Mobile Services Nodes	(B)	Agent desktop application	Invoking Genesys Mobile Services APIs: Service, Notification and Storage APIs.	HTTP/ HTTPS	Deployment specific. See third party documentation.	Hard coded as part of the Genesys Mobile Services API URL inside agent desktop client code.	Security proxy should be configured to block access to the API URLs according to the access rules

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
						Could be configured in Configuration Manager and read dynamically by desktop application.	below.
Genesys Configuration Server	(C)	Genesys Mobile Services node	Reading Genesys Mobile Services node configuration and reacting on changes	TCP/IP protected by TLS is required	Default: 2020. Configured in Configuration Manager configuration file.	Configuration is in *<Genesys Mobile Services installation directory>/startServer.bat* parameters **-app', '-host' and '-port**. See *launcher.xml* in the same directory for description of the parameters.	Enable TLS encryption if a higher level of protection is required. TLS is a Platform SDK-based connection supporting standard Genesys security. Specify a secure Configuration Server port for the connection.
Apple Mobile Native Message Push Service	(D)	Genesys Mobile Services node	Sending messages to mobile clients using Apple's devices	TCP/IP, binary	host: *gateway.push.apple.com*, port 2195; Controlled by Apple	See Apple Notification for more details on how to configure Genesys Mobile Services to access Apple Push Notification Service.	Apple requires an SSL/ TLS-enabled connection with client side certificates. Make sure your firewall allows access to the *https://gateway.push.appl* URL from all Genesys Mobile Services nodes if

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							you are planning to use this interface.
Google Mobile Native Message Push Service (also called Google Cloud Messaging)	(E)	Genesys Mobile Services node	Sending messages to mobile clients using Android devices	HTTPS	Default: 80. Configured by Google.	See GCM Service for more details on how to configure Genesys Mobile Services to access Google Cloud Messaging	Google require SSL protected connection with client side certificates. Make sure your firewall allows access to the *https://android.googleapis* *gcm/send* URL from all Genesys Mobile Services nodes if you are planning to use this interface.
Third Party Application Inside Enterprise Network	(F)	Genesys Mobile Services node	Sending Web Callback Notification messages to a third party application subscribed for notifications	HTTP	Third party application specific. Provided by the application as a web callback URL parameter when the subscription is created.	Client configuration: Genesys Mobile Services node will use the web callback URL provided by the third party application when a subscription is created.	Currently HTTPS is not supported for this connection. Used only for notifications within a corporate network. Configure SSL proxy if stronger protection is needed.

Customer Facing API Access Rules for Security Gateway in Front

of Genesys Mobile Services

The following access rules should be used as a model for your environment, allowing a list of services provided by your Genesys Mobile Services deployment to be accessed while restricting the use of internal-only services. **Note:** Only allow access to the limited set of services listed by name.

```
 Allow access from the end user application or proxy - mobile, web, etc:
 -- async notifications over HTTP API:
 If client is using cometd transport (typically for chat):
 {base url:port}/genesys/cometd
 -- service API:
 {base url:port}/genesys/{version}/service/{service name}
 {base url:port}/genesys/{version}/service/{id}/storage
 {base url:port}/genesys/{version}/service/{id}
 {base url:port}/genesys/{version}/service/{id}/{request name}
 -- chat media API:
 {base url:port}/genesys/{version}/service/{id}/ixn/chat
 {base url:port}/genesys/{version}/service/{id}/ixn/chat/refresh
 {base url:port}/genesys/{version}/service/{id}/ixn/chat/disconnect
 {base url:port}/genesys/{version}/service/{id}/ixn/chat/startTyping
 {base url:port}/genesys/{version}/service/{id}/ixn/chat/stopTyping
 {base url:port}/genesys/{version}/service/{id}/ixn/chat/*

 Only when client need direct access allow (in most cases only ORS/scxml need it):
 -- storage API
 {base url:port}/genesys/{version}/storage/{ttl}
 {base url:port}/genesys/{version}/storage/{data id}/{ttl}
 {base url:port}/genesys/{version}/storage/{data id}
 {base url:port}/genesys/{version}/storage/{data id}/{key}
 -- callback (management) API
 {base url:port}/genesys/{version}/service/callback/{callback-execution-name}/{service_id}
 {base url:port}/genesys/{version}/service/callback/{callback-execution-name}/{service_id}
 {base url:port}/genesys/{version}/service/callback/{callback-execution-
name}/availability?{timestamp=value}
 {base url:port}/genesys/{version}/admin/callback/
queues?target={target_name}&end_time={iso_end_time}

 Allow access from load balancer for node health check:
 {base url:port}/genesys/{version}/node

 Allow access from the intranet:
 -- admin UI
 {base url:port}/genesys/admin/*
 {base url:port}/genesys/{version}/admin/*
 {base url:port}/genesys/{version}/reports/*
 {base url:port}/genesys/{version}/statistic/*
 -- datadepot (typically accessed from ORS/scxml):
 {base url:port}/genesys/{version}/datadepot/{tenantId}/{activityName}/dates
 {base url:port}/genesys/{version}/datadepot/{tenantId}/activities
 {base url:port}/genesys/{version}/datadepot/{tenantId}/{activityName}/aggregates
 {base url:port}/genesys/{version}/datadepot/{tenantId}/{activityName}/aggregates/unique
 -- all built-in services (except chat)
 {base url:port}/genesys/{version}/service/request-access
 {base url:port}/genesys/{version}/service/match-interaction
 -- notification API (typically accessed from ORS/scxml):
 {base url:port}/genesys/{version}/notification/subscription
 {base url:port}/genesys/{version}/notification/subscription/{id}
 {base url:port}/genesys/{version}/notification/publish
```

Mobile Native Push Notification Configuration Details

Two major mobile platforms are currently supported: Android and Apple. Details about Genesys Mobile Services configuration could be found on the Push Notification Service page of the API Reference.

Client Side Port Definitions for Genesys Framework Connections

The following connections support client side port definition functionality:

- Connection to Configuration Server - This port definition is defined as part of the installation, and as an environment variable.
- HTTP Connections between Genesys Mobile Services nodes – This port definition is defined as part of the configuration data associated with the Genesys Mobile Services Application object, and is used to create the connections between Genesys Mobile Services nodes.
- HTTP Connection from Genesys Mobile Services to ORS – This port definition is defined as part of the configuration data associated with the Genesys Mobile Services Application object and is used to create the connection with ORS.
- Genesys Mobile Services to Chat Server Connections - this port definition is defined as part of the configuration data associated with the Genesys Mobile Services application object and is used to create the connection with Chat Server.

Note: Connections from client applications (CometD and REST API requests) do *not* have client side port definitions.

Hiding Sensitive Data in Logs

Genesys recommends using log data filtering to hide sensitive configuration data in log files. The given product will then use that filter to determine what content should be filtered or masked when creating log files. The only real interface to Genesys Mobile Services are APIs, and having to configure filter criteria every time you add a new parameter or API is cumbersome and complicated. Therefore, Genesys Mobile Services supports filtering and masking data in log files for any API parameter that matches the following naming convention:

- Filter: Any parameter name that is prefixed with XXX will be filtered out of the log files entirely. Example: *XXX_FilterParam*
- Mask: Any parameter name that is prefixed with MMM will be masked in the log files, showing in the log with the letters MMM. Example: *MMM_MaskParam*

Configuring SSL Connection Listener for a Jetty Container

Beginning with Jetty 7.3.1, the preferred way to configure SSL parameters for the connector is by configuring the *SslContextFactory* object and passing it to the connector's constructor. Jetty has two SSL connectors–the *SslSocketConnector* and the *SslSelectChannelConnector*. The *SslSocketConnector* is built on top of the Jetty *SocketConnector* which is Jetty's implementation of a blocking connector. It uses Java's *SslSocket* to add the security layer. The *SslSelectChannelConnector* is an extension of Jetty's *SelectChannelConnector* which uses non-blocking IO. For its security layer, it uses java nio *SslEngine*. You can configure these two connectors similarly; the difference is in the implementation. The following is an example of an *SslSelectChannelConnector* configuration. You can configure an *SslSocketConnector* the same way, just change the value of the class to *org.eclipse.jetty.server.ssl.SslSocketConnector*.

```
 <Call name="addConnector">
    <Arg>
      <New class="org.eclipse.jetty.server.ssl.SslSelectChannelConnector">
        <Arg>
          <New class="org.eclipse.jetty.http.ssl.SslContextFactory">
            <Set name="keyStore"><SystemProperty name="jetty.home" default="." />/etc/
keystore</Set>
            <Set name="keyStorePassword">OBF:1vny1zlo1x8e1vnw1vn61x8g1zlu1vn4</Set>
            <Set name="keyManagerPassword">OBF:1u2u1wml1z7s1z7a1wnl1u2g</Set>
            <Set name="trustStore"><SystemProperty name="jetty.home" default="." />/etc/
keystore</Set>
            <Set name="trustStorePassword">OBF:1vny1zlo1x8e1vnw1vn61x8g1zlu1vn4</Set>
          </New>
        </Arg>
        <Set name="port">8443</Set>
        <Set name="maxIdleTime">30000</Set>
      </New>
    </Arg>
 </Call>
```

Choosing a Directory for the Keystore

The keystore file in the example above is given relative to the Jetty home directory. For production, choose a private directory with restricted access to keep your keystore in. Even though it has a password on it, the password may be configured into the runtime environment and is vulnerable to theft. You can now start Jetty the normal way (make sure that *jcert.jar, jnet.jar* and *jsse.jar* are on your classpath) and SSL can be used with a URL like: https://localhost:8443/

Setting the Port for HTTPS

Remember that the default port for HTTPS is 443 not 80, so change 8443 to 443 if you want to be able to use URLs without explicit port numbers. For a production site it normally makes sense to have an *HttpListener* on port 80 and a *SunJsseListener* on port 443. Because these are privileged ports, you might want to use a redirection mechanism to map port 80 to 8080 and 443 to 8443, for example. The most common mistake at this point is to try to access port 8443 with HTTP rather than HTTPS.

Redirecting HTTP requests to HTTPS

To redirect HTTP to HTTPS, the webapp should indicate it needs CONFIDENTIAL or INTEGRAL connections from users. This is done in web.xml:

```
<web-app>
...
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Everything in the webapp</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
</web-app>
```

Then you need to tell the plain HTTP connector that if users try to access that webapp with plain HTTP, they should be redirected to the port of your SSL connector (the "confidential port"):

```
<Call name="addConnector">
 <Arg>
    <New class="org.eclipse.jetty.nio.SelectChannelConnector">
       ...
       <Set name="confidentialPort">443</Set>
    </New>
 </Arg>
</Call>
```

Using the Denial of Service Filter

The Denial of Service (DoS) filter limits exposure to request flooding, whether malicious, or as a result of a misconfigured client. The DoS filter keeps track of the number of requests from a connection per second. If the requests exceed the limit, Jetty rejects, delays, or throttles the request, and sends a warning message. The filter works on the assumption that the attacker might be written in simple blocking style, so by suspending requests you are hopefully consuming the attacker's resources. The DoS filter is related to the QoS filter, using Continuations to prioritize requests and avoid thread starvation. See the Jetty documentation for more information: http://wiki.eclipse.org/Jetty/Reference/DoSFilter

The DoS filter is configured in the GMS `web.xml` file, located in the `webapp/WEB-INF` directory. The default configuration is:

```
<filter>
<filter-name>DoSFilter</filter-name>
<filter-class>org.eclipse.jetty.servlets.DoSFilter</filter-class>
<init-param>
<param-name>maxRequestsPerSec</param-name>
<param-value>100</param-value>
</init-param>
<init-param>
<param-name>ipWhitelist</param-name>
<param-value>127.0.0.1</param-value>
</init-param>
</filter>
```