Appendix B: Security in ApplinX

End-To-End Security

ApplinX multi-tier architecture supports end-to-end security by utilizing encryption and industry-standard, secured protocols within each layer of communication. The following document details the security measures that are available for each layer, as well as additional security mechanisms available in other ApplinX components.

For SOA and Web Enablement Solutions

Host <> ApplinX Server

Communication between ApplinX Server and the host can be encrypted using SSL V3. Both client and server authentication are supported. SSL X509 certificate is stored using standard Keystore implementations (JCEKS).

This feature is available for any host that supports SSL V3 communication, however, this has only been tested on Mainframe hosts. It is also possible to use the secured protocol SSH V2 (instead of the VT protocol).

To configure an SSL connection between the host and ApplinX server:

- Refer to Configuring the SSL Connection.

For Web Enablement Solutions

ApplinX Server <> ApplinX Base Object/ApplinX Web Applications

Communication between ApplinX Server and the ApplinX Base Object (which resides on the web server or application server) can be encrypted using SSL (this layer of security includes encryption only, without authentication).

To configure an SSL connection between ApplinX server and the Web Server/Application Server:

1. In the Server Properties, General tab, select the **Secured port** checkbox.
2. In the Web Application Configuration Manager set the Session server URL to the SSL port (e.g. applinxs://localhost:23443).

   It is possible to set the address dynamically from within the Base Object by using the method setServerURL in the ApplinX SessionConfig object within the GXBasicContext file, gx_initSessionConfig method (IPv4 and IPv6 address formats are supported).

   Note:
   When you want to enable connecting to the HTTPS port only from the ApplinX server machine, add the following system variable when starting the server:
   `-Dcom.sabratec.applinx.securesocket.localonly=true`

Web Server / Application Server <> End User Browser

Communication between the end user web browser and the Web server or application server can be secured using HTTPS, or a firewall. This feature is not related to ApplinX and should be supported by the Web Server / Application Server.

The ApplinX .NET Framework also supports authentication of end users using NT authentication, as this is a built-in feature in the Microsoft .NET environment.

For SOA Solutions

ApplinX Server <> Procedure Clients

The Java and .NET Procedure Clients (binary SOA clients), can be encrypted using HTTPS (SSL).

To configure an SSL connection between ApplinX server and a Procedure Client

1. Follow the instructions in the following topic: Creating a Secure SSL Connection between a Procedure Client and ApplinX Server.

   Note:
   When you want to enable connecting to the HTTPS port only from the ApplinX server machine, add the following system variable when starting the server:
   `-Dcom.sabratec.applinx.securesocket.localonly=true`

ApplinX Server <>Web Services (WS-Stack)

To configure an SSL connection between ApplinX server and a Web Service (WS-Stack)

1. In the Server Properties, General tab, select the **Secured port** checkbox. ApplinX will automatically connect to WS-Stack using a secure connection.

   Note:
   When you want to enable connecting to the HTTPS port only from the ApplinX server machine, add the following system variable when starting the server:

```
    -Dcom.sabratec.applinx.securesocket.localonly=true
```

Web Services (WS-Stack) <> Web Service Client

This layer can be encrypted using HTTPS (SSL). Refer to http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.htmland also to the relevant commented section in <ApplinXInstallationDirectory>/conf/server.xml.

Development Time

ApplinX allows managing password-protected users, groups and their permissions. It is possible to define certain permissions to a group, and then associate users with this group, giving the user the permissions defined for this group or to define specific users permissions. Each user/group can be assigned with read/write permissions at the application or folder level. The users' definitions are saved in an encrypted configuration file.

It is also possible to define users based on NT domain users (NT authentication).

Connection Pools

It is possible to specify passwords of host users as part of the connection information sets of connection pools (to enable connection pooling with automatic login to the host application). These passwords are encrypted and saved in the application's repository database.

Running ApplinX Server with a Java Policy File

In order to run the ApplinX server with a Java security manager enabled, the following flags should be appended to the Start_Process_Parameters in the <ApplinX installation>\bin\start-gxserver.bat file, or to the JAVA_OPTS in the <ApplinX installation>\bin\start-gxserver.sh file or to the Start_Process_Parameters in the GXApplinXService.ini file:

```
-Djava.security.manager -Djava.security.policy=./conf/catalina.policy
```

In the policy file (specified in the path above) the following permissions are set inside a grant section (if a different policy file is used, one should add the following manually):

```
permission java.net.SocketPermission "localhost:2323" , "listen,resolve,accept";
permission java.net.SocketPermission "localhost:*" , "resolve,accept";
permission java.net.SocketPermission "<host name>:<host port>" , "connect,resolve";
permission java.io.FilePermission "${com.sabratec.gxhome}/-", "read, write, delete";
permission java.io.FilePermission "${catalina.home}/-", "read";
permission java.io.FilePermission "${java.home}/../-", "read";
permission java.io.FilePermission "${java.io.tmpdir}/" , "read, delete, write";
permission java.io.FilePermission "${java.io.tmpdir}/-" , "read, delete, write";

//ApplinX Xstream usage. Used mostly by ApplinX configuration persist to XML
permission java.lang.RuntimePermission "accessClassInPackage.sun.misc";
permission java.lang.RuntimePermission "accessClassInPackage.sun.reflect";
permission java.lang.RuntimePermission "accessClassInPackage.sun.io";
permission java.lang.RuntimePermission "accessClassInPackage.sun.logging.*";
permission java.lang.RuntimePermission "defineClassInPackage.org.apache.jasper.runtime";
permission java.lang.RuntimePermission "accessDeclaredMembers";
permission java.lang.RuntimePermission "createClassLoader";
permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
permission java.lang.RuntimePermission "reflectionFactoryAccess";
permission java.io.SerializablePermission "enableSubclassImplementation";
permission java.lang.RuntimePermission "getClassLoader";
// For using Log4J
permission java.lang.RuntimePermission "defineClassInPackage.java.lang";

// Used for showing the server icon in the system tray. Uncomment if needed.
// permission java.lang.RuntimePermission "loadLibrary.GXUtil";
// permission java.lang.RuntimePermission "modifyThreadGroup";

permission java.io.SerializablePermission "enableSubstitution";
permission java.sql.SQLPermission "setLog";
permission java.util.PropertyPermission "com.sabratec.*", "read,write";
permission java.util.PropertyPermission "com.softwareag.*", "read,write";
permission java.util.PropertyPermission "*", "read";
permission java.util.PropertyPermission "org.apache.adb.properties", "read,write";
permission java.util.PropertyPermission "javax.xml.registry.ConnectionFactoryClass", "write";
```

To allow ApplinX to integrate with WSStack, the following are needed:

```
permission java.net.SocketPermission "<ip of host to where the WSStack is deployed>:*", "accept,resolve";
permission java.net.SocketPermission "<machine name to where WSStack is deployed: WSStack Http port>", "connect,resolve";
permission java.lang.RuntimePermission "getProtectionDomain";
permission java.lang.RuntimePermission "modifyThread";
permission java.lang.RuntimePermission "getenv.WS_STACK_HOME";

// In case WSStack is run in the server’s process, the following are also needed
permission java.net.NetPermission "specifyStreamHandler";
permission java.lang.RuntimePermission "shutdownHooks";
permission java.lang.RuntimePermission "defineClassInPackage.org.apache.jasper.runtime";
permission java.lang.RuntimePermission "modifyThread";
permission java.lang.RuntimePermission "setContextClassLoader";
```

Note:
lines with a close that starts with a single ’<’ character, should be edited according to the text inside the close.

When ApplinX is running with SSL support, the following should be added as well:

```
permission java.io.FilePermission "${java.home}/jre/bin/keytool" ,"execute";
```

Blocking Access to URLs in an ApplinX Web Application

You can block access to each URL of a web application by using a property file that contains the URLs of the web application pages and the access to each URL. The following access roles are available:

- **Unauthenticated**
  User that did not pass the authentication process and has the minimal access permission.
- **User**
  User that has regular permissions
- **Admin**
  User with the highest permissions.

To enable blocking of selected URLs

1. Uncomment the `filter` section in file *web.xml* of the web application.

```
<!-- <filter>
    <filter-name>GXCheckURLProxyFilter</filter-name>
    <filter-class>com.sabratec.applinx.j2ee.framework.web.filters.GXCheckURLProxyFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>GXCheckURLProxyFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping> -->
```

2. Update the property file *URLPermissions.properties* to set the URL permission. Asterisk notation is supported for directories and file extensions.

3. Set the role of the user in a session by setting the method `setSessionRole` in class inherited from `GXAbstractWebContext`.

The table below shows the default access for the URLs of ApplinX applications:

Resource Pattern	**Role**
`z_resourceReader.jsp`	All (Unauthenticated)
`run_printlet.jsp`	All (Unauthenticated)
`config/*`	Administrator
`log/*`	Administrator
`z_editConfig.jsp`	Administrator
`.class`	Administrator
`Index.jsp`	All (Unauthenticated)
`__PREVIOUS_VERSION`	Administrator

All URLs not defined in the property files can be accessed only by an authenticated user. When one file is set to two roles, the last definition in the property file has priority.

By using asterisk notation to define directories and/or group of files for a role (e.g. `config/*`) you can exclude one file from the group by providing a more exact definition (e.g. `config/web.xml`).

This feature is available when creating a new web application. If you want to use this feature for an existing web application, you will need to manually add the filter code to the file *web.xml* and also add the property file *URLPermissions.properties* to the application.