Server-side modules are Java classes that load when a Wowza Streaming Engine™ media server software application starts. They provide much of the functionality that controls the streaming process. This article explores the various types of modules available in Wowza Streaming Engine, including built-in, utility, and custom modules.

**About server-side modules**

Server-side modules are configured on a per-application basis. They load dynamically when the Wowza Streaming Engine application starts. Server-side modules are useful for controlling Adobe HDS, Apple HLS, Microsoft Smooth Streaming, MPEG-DASH, and RTSP/RTP streaming. They also provide remote methods that can be called from Adobe Flash Player.

Typically, modules are bound to .jar files in the Wowza Streaming Engine installation. Modules can leverage third-party libraries or built-in Java functionality if the dependent .jar files are copied to the `lib` folder in the Wowza Streaming Engine installation. You control whether and how modules are used by editing them on the **Modules** tab of the application in Wowza Streaming Engine Manager.
**Note:** Items on this page should be configured by advanced users only.

**Modules** Java classes that extend an application’s functionality. The list below defines an order-dependent list of modules to be loaded for a given application. The modules are loaded dynamically when the application instance is loaded. The base (ModuleCore) module must be included by the application for it to operate properly.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Fully Qualified Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>Base</td>
<td>com.wowza.wms.module.ModuleCore</td>
</tr>
<tr>
<td>logging</td>
<td>Client Logging</td>
<td>com.wowza.wms.module.ModuleClientLogging</td>
</tr>
<tr>
<td>flvPlayback</td>
<td>FLV Playback</td>
<td>com.wowza.wms.module.ModuleFLVPlayback</td>
</tr>
<tr>
<td>ModuleCoreSecurity</td>
<td>Core Security Module for Applications</td>
<td>com.wowza.wms.security.ModuleCoreSecurity</td>
</tr>
</tbody>
</table>

**Notes:**

- If you add a module to an application while it’s running, you must restart the application for the changes to take effect.
- If you modify an installed module, you must restart Wowza Streaming Engine for the changes to take effect.

**Use built-in modules**

The following modules are built in to Wowza Streaming Engine and are used by applications for basic functionality.

<table>
<thead>
<tr>
<th>Module</th>
<th>Class</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>Class</th>
</tr>
</thead>
</table>


<table>
<thead>
<tr>
<th>Module</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModuleCore</td>
<td>com.wowza.wms.module.ModuleCore</td>
</tr>
<tr>
<td>ModuleClientLogging</td>
<td>com.wowza.wms.module.ModuleClientLogging</td>
</tr>
<tr>
<td>ModuleFLVPlayback</td>
<td>com.wowza.wms.module.ModuleFLVPlayback</td>
</tr>
</tbody>
</table>
ModuleCoreSecurity com.wowza.wms.security.ModuleCoreSecurity
Use utility modules

Several ready-to-use utility modules are also available to for Wowza Streaming Engine. The Wowza IDE is not required to implement these modules.

These utility modules are designed for Wowza Streaming Engine. They offer such useful functions as adding an audio track to a video-only stream, which some CDNs require, and sending connection and stream stats to Google Analytics. Most are backward-compatible, however, some aren’t fully supported in Wowza Media Server™ software.

Every utility module is documented by an article that provides a link to download the module and includes configuration instructions. Each article also links to the module’s source code on GitHub. See Module examples for a complete list of all utility module articles.

Use custom modules

Finally, you can develop your own module classes—and custom methods—using the Wowza IDE.

Here’s a sample custom module that uses the event method onAppStart and the custom method doSomething:
To use a custom module like this, compile it, package it into .jar file, and put it in the `install-dir`/lib folder. Then, add it to an application by going to the application’s Module tab in Wowza Streaming Engine Manager, clicking Add Module, and specifying the module’s Name, Description, and Fully Qualified Class Name. When you’re done, the new module appears in the list and is ready for the application to use.

```
package com.mycompany.module;

import com.wowza.wms.module.*;
import com.wowza.wms.client.*;
import com.wowza.wms.amf.*;
import com.wowza.wms.request.*;

public class MyModule extends ModuleBase {
    public void onAppStart(IApplicationInstance appInstance) {
        getLogger().info("onAppStart");
    }

    public void doSomething(IClient client, RequestFunction function, AMFDataList params) {
        getLogger().info("doSomething");
    }
}
```

To use a custom module, the code snippet above is used. Modules are Java classes that extend an application’s functionality. The list below defines an order-dependent list of modules to be loaded for a given application. The modules are loaded dynamically when the application instance is loaded. The base (ModuleCore) module must be included by the application for it to operate properly.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Fully Qualified Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>Base</td>
<td>com.wowza.wms.module.ModuleCore</td>
</tr>
<tr>
<td>logging</td>
<td>Client Logging</td>
<td>com.wowza.wms.module.ModuleClientLogging</td>
</tr>
<tr>
<td>flvPlayback</td>
<td>FLVPlayback</td>
<td>com.wowza.wms.module.ModuleFLVPlayback</td>
</tr>
<tr>
<td>ModuleCoreSecurity</td>
<td>Core Security Module for Applications</td>
<td>com.wowza.wms.security.ModuleCoreSecurity</td>
</tr>
<tr>
<td>MyModule</td>
<td>This is MyModule</td>
<td>com.mycompany.module.MyModule</td>
</tr>
</tbody>
</table>

Troubleshoot custom module connections to external web services
When running custom module code with Wowza Streaming Engine version 4.7.8 or later that connects to external web services, you may have SSL failures that result in exceptions being reported in Wowza Streaming Engine logs. Log messages may look similar to this:

```
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException
  : PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
```

An error like the one above is likely due to missing SSL certificates in your Java installation. This may occur when the web service is using a self-signed certificate or a certificate that is issued by an internal certificate authority.

To resolve this issue, import the web service’s certificates into your Java cacerts keystore. You may need to run the following commands as an administrator or with elevated privileges.

1. Get the root and intermediate certificates for the domains using the openssl tool and save them to a file.
   For example, if you were having trouble reaching `example.com` you would use the following command:
   ```
   openssl s_client -showcerts -connect example.com:443 > example.crt
   ```

2. Navigate to your Java installation directory using the terminal or command prompt.

3. Import the generated crt file into your cacerts keystore of your Java installation with the keytool command.
   ```
   keytool -importcert -keystore /path-to-jre-9.0.4/lib/security/cacerts -storepass changeit -file example.crt -alias example-root
   ```

4. Verify that the certificates have been imported with the keytool and list command.
   ```
   keytool -list -v
   ```

5. Restart your Wowza Streaming Engine service for the changes to go into effect.