JMX Security and Configuration

⚠ Up to AIE 4.2, this page was called "JMX Administration."

Overview

In order to support remote/automated monitoring of AIE, a Java Management Extensions (JMX) web console and a JMX Remote Method Invocation (RMI) registry are exposed. Tools such as jconsole can be used to connect to the JMX server for monitoring and management.

View incoming links. **>>**

Security Guide

- Overview
- Enabling Remote JMX Authentication
    - Providing Credentials to acquire ExposedApi
- 3rd party JMX Management Tools

Enabling Remote JMX Authentication

By default, the remote JMX connectivity via RMI does not require authentication. In order to authenticate against a selected group of users, go to the **<project_dir>\conf\properties\core-app\attivio.core-app.properties** file and setting for jmx.auth.provider.bean.name property to identify AuthenticationProvider bean to use with remote clients. If this property is not specified, AIE will attempt to use the default security authentication provider bean, similar to Deploy Webapp feature.

<project_dir>\conf\properties\core-app\attivio.core-app.properties

```
# Uncomment to specify the bean name of AuthenticationProvider for remote jmx
# See documentation on JMX authentication for expectations
# Note: Some AIE exposed APIs were implemented with Remote jmx
#
#jmx.auth.provider.bean.name=
```

Credentials to JMX Console

Use **mx4j.credentials.username** and **mx4j.credentials.password** (see below) to specify credentials to JMX console.

<project_dir>\conf\properties\core-app\attivio.core-app.properties

```
# Uncomment to add authentication to web interfaces (mx4j) to jmx
# The password can be encrypted with the following command and then wrapped in *{encryptedValuehere}
#
# aie-exec encrypt --password "mypassword"
#mx4j.credentials.username=admin
#mx4j.credentials.password=mypassword
```

Providing Credentials to acquire ExposedApi

Beside establishing JMX connectivity explicitly to get registered managed bean, AIE Service Factory framework also exposes implementations of Exposed API via JMX. Using these services remotely when JMX authentication is turned on thus require credentials. See the following sample code that acquired a ConfigurationAPI for example:

```
public static void main(String[] args) throws Exception {

  Platform.instance.setProjectName("demoproject");
  System.setProperty("AIE_ZOOKEEPER", "localhost:16980");
  ServiceFactory.setFactoryParams(new JmxAuthParams() {
    @Override
    public String getUsername() {
      return "foo";
    }
    @Override
    public String getPassword() {
      return "dev";
    }
  });
  ConfigurationApi api = ServiceFactory.getService(ConfigurationApi.class);
  System.out.println(api.getApplicationName());
}
```

See ServiceFactory and ServiceDiscoveryHelper .

Configuring SSL For JMX

The transport for JMX can be configured to use SSL for an added layer of security. Add **jmx.ssl.enable=true** to any property file, or uncomment it from **<project-dir>/conf/properties/core-app/attivio.core-app.properties**. If you have already generated key and certificate information when configuring SSL for the AIE Administrator, there is nothing more to do. If you want JMX configured for SSL but not the AIE Administrator, then generate key and certificate information as described in the first four steps of Enabling SSL HTTPS for AIE Web Applications. (There is no need to configure Jetty in this case).

<project_dir>\conf\properties\core-app\attivio.core-app.properties

```
# Uncomment to force web access to JMX over SSL
# The same keystore is used as is configured for 'SSL for the AIE Administrator'
#jmx.ssl.enable=true
```

Note: If JMX SSL is enabled in the application, be aware that connecting to JMX will require a few extra steps than if JMX SSL is not enabled. If you would like to connect using tools such as jconsole you need to add the following system properties when you start up that tool to ensure that you can connect securely:

example_jconsole_script.sh

```
$JAVA_HOME/bin/jconsole \
-J-Djavax.net.ssl.keyStore=/path/to/keystore/configured/for/attivio.ks \
-J-Djavax.net.ssl.keyStorePassword=configuredPassword \
-J-Djavax.net.ssl.trustStore=/path/to/truststore/configured/for/attivio.ks \
-J-Djavax.net.ssl.trustStorePassword=configuredPassword
```

If you wish to create your own client using java you can set the system properties when you start your process by passing in the arguments you see above (minus the "-J") or simply set the properties in your code before attempting to establish the secure JMX connection:

ExampleJMXClient.java

```
System.setProperty("javax.net.ssl.keyStore","/path/to/keystore/configured/for/attivio.ks");
System.setProperty("javax.net.ssl.keyStorePassword","configuredPassword");
System.setProperty("javax.net.ssl.trustStore","/path/to/truststore/configured/for/attivio.ks");
System.setProperty("javax.net.ssl.trustStorePassword","configuredPassword");
```

3rd party JMX Management Tools

Many management tools such as jconsole can connect to remote processes in order to view and manage system configuration and statistics. By default, AIE creates a RMI endpoint for remote JMX management. In order to connect your 3rd party tool to the AIE JMX server, use the following URI to connect. Note: this URI assumes you used the default 17000 baseport.

```
service:jmx:rmi:///jndi/rmi://localhost:17004/jmxrmi
```