**ECAS Authentication Provider Setup for USM Back-end**

# Introduction

Setting-up and ECAS Authentication Provider for the USM back-end (i.e. the USM- Administration RESTful web-services) and configuring the USM back-end for ECAS based user authentication is possible yet not necessarily desired as the ECAS authentication mechanism is too intrusive for RESTful web-services as it involved a redirection to an ECAS server login page targeted at human beings and not to systems.

In this document we however present how to [install the ECAS Authentication Provider](#_toc30) and how to [configure the USM Back-end for ECAS authentication](#_toc56) on the basis that:

* The information provided may be used to restrict access to the USM front-end (more suitable for ECAS authentication since it is targeted to human beings) to end-users authenticated via ECAS.
* The information provided may be used to restrict access to both the USM front-end and back-end in the case where both components (front and back) are packaged in a single WAR (using the back-end WAR as an overlay for the front-end) or in a single EAR (presumably with shared session configuration)

# Assumptions

As ECAS is an European Commission product that only exists inside the European Commission it is assumed that the ECAS Authentication Provider and the USM Back-end are to be deployed onto an Oracle Weblogic Application Server instance (primary J2EE platform used by the European Commission).

# References

1. **ECAS Documentation**: <https://webgate.ec.europa.eu/CITnet/confluence/display/IAM/ECAS>
2. **ECAS Repository**: <https://webgate.ec.europa.eu/CITnet/confluence/display/IAM/ECAS+Forge>
3. **ECAS Client** **Installation and Configuration Guide – Basic***:* [*https://webgate.ec.europa.eu/CITnet/svn/ecas-public/clients/java/tags/4.3.1/doc/ECAS%20Client%20Installation%20and%20Configuration%20Guide%20-%20Basic.pdf*](https://webgate.ec.europa.eu/CITnet/svn/ecas-public/clients/java/tags/4.3.1/doc/ECAS Client Installation and Configuration Guide-Basic.pdf)

# ECAS Authentication Provider Installation Procedure

1. Make a backup copy of your Weblogic domain configuration (config.xml and start-sup shell scripts)
2. Download the latest/desired ECAS Client version (currently 4.3.1) from the ECAS Repository (see [References](#_toc25)). This includes:
   * **ecas-weblogic-10.3-authprovider-4.3.1.jar**: The ECAS Client JAR EcasIdentityAsserterV2 for WebLogic Server 10.3 and above
   * **ecas-demo.ear**: The demo application (optional)
   * **security.properties**: A customized resource bundle to provide descriptions for the EcasIdentityAsserterV2 provider specific page in the WebLogic admin console.
   * **log4j-1.2.15.jar**: a patched log4j JAR
   * **log4j.xml**: The example log4j configuration file (to be adapted to your environment)
3. Download the ECAS Client *Installation and Configuration Guide – Basic* from the ECAS Repository (see [References](#_toc25)).
4. Install the ECAS Client as described in the *Installation and Configuration Guide – Basic* document. In particular, do make sure that you follow the instructions provided in Section 4.3. *Security requirements* of the aforementioned document to:
   * **Enable strong cryptography**
   * **Disable SSLv3** (see [Sample WebLogic start-up script](#_toc136))
   * **Disable SSLv3 for WebLogic-generated Web Service clients** (necessary on Weblogic 12.1.1 but not on version 12.1.2)
5. After completing the installation of the ECAS Client, make sure you take a note of the fact that the Weblogic administration console is now accessible only via the <http://localhost:7001/console/login/LoginForm.jsp> URL (unless your weblogic administrator user is defined in ECAS user database).
6. If your Weblogic server/domain is deployed outside the European Commission, you will most probably integrate with an ECAS Mockup Server instance rather than with the (real) ECAS server instance with which the ECAS Client is (pre-)configured to integrate. If this is the case, you should open the Weblogic administration console (see above), navigate to the ***Security Realms >myrealm >Providers >EcasIdentityAsserterV2*** screen and update the following properties in the ***Provider Specific*** panel (see [Sample WebLogic configuration](#_toc161):

* **Assurance Level**: LOW (unless you need to accept internal EC users only)
* **Accept Strengths**: BASIC
* **Ecas Base Url**: base URL of your ECAS Server Mockup instance (e.g. [*https://svm-midway.athens.intrasoft-intl.private:7012*](https://svm-midway.athens.intrasoft-intl.private:7012/))
* **Ecas Server Direct Host Name**: host name (or IP address) of your ECAS Server Mockup instance (e.g. *svm-midway.athens.intrasoft-intl.private*)
* **Ecas Server Direct One Way Ssl Port**: port number of your ECAS Server Mockup instance (e.g. *7012*)

1. Restart your Weblogic server instance for the latest changes to take effect.
2. Thats' all folks!

# Configure the USM Back-end for ECAS authentication

Configuring the USM Back-end for ECAS authentication is described in general terms in section 6. *PROTECT YOUR WEB APPLICATION* of the ECAS Client *Installation and Configuration Guide – Basic* (see [References](#_toc25)).

In a nutshell, performing this configuration entails at the minimum the execution of the following three actions:

1) Add a security constraint to the **web.xml** deployment descriptor defined in the Administration-Rest-Service as depicted below

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns="http://java.sun.com/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  version="3.0">  <session-config>  <session-timeout>30</session-timeout>  </session-config>  **<security-constraint>**  **<web-resource-collection>**  **<web-resource-name>protected</web-resource-name>**  <description>  This is the protected REST services.  </description>  **<url-pattern>\*</url-pattern>**  **</web-resource-collection>**  **<auth-constraint>**  <description>  Requires users to be authenticated but  does not require them to be authorized.  </description>  **<role-name>\*</role-name>**  **</auth-constraint>**  <user-data-constraint>  <description>Encryption is not required for this area.</description>  <transport-guarantee>NONE</transport-guarantee>  </user-data-constraint>  **</security-constraint>**  </web-app> |

*Configuring the* *USM Back-end web.xml deployment descriptor*

2) Add an allow-all-roles option to the **weblogic.xml** deployment descriptor defined in the Administration-Rest-Service as depicted below

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <weblogic-web-app xmlns="http://xmlns.oracle.com/weblogic/weblogic-web-app">  <context-root>usm-administration</context-root>  <container-descriptor>  **<!-- Allow some resources to be accessed by**  **authenticated users who do not possess any role -->**  **<allow-all-roles>true</allow-all-roles>**  <prefer-web-inf-classes>true</prefer-web-inf-classes>  </container-descriptor>  </weblogic-web-app> |

*Configuring the USM Back-end weblogic.xml deployment descriptor*

3) Build and deploy the Administration-Rest-Service without executing the integrations tests as depicted below

|  |
| --- |
| ~/dev/mare/unionvms/trunk/USM/java/administration $ **mvn clean install -Pswagger,weblogic -DskipTests** |

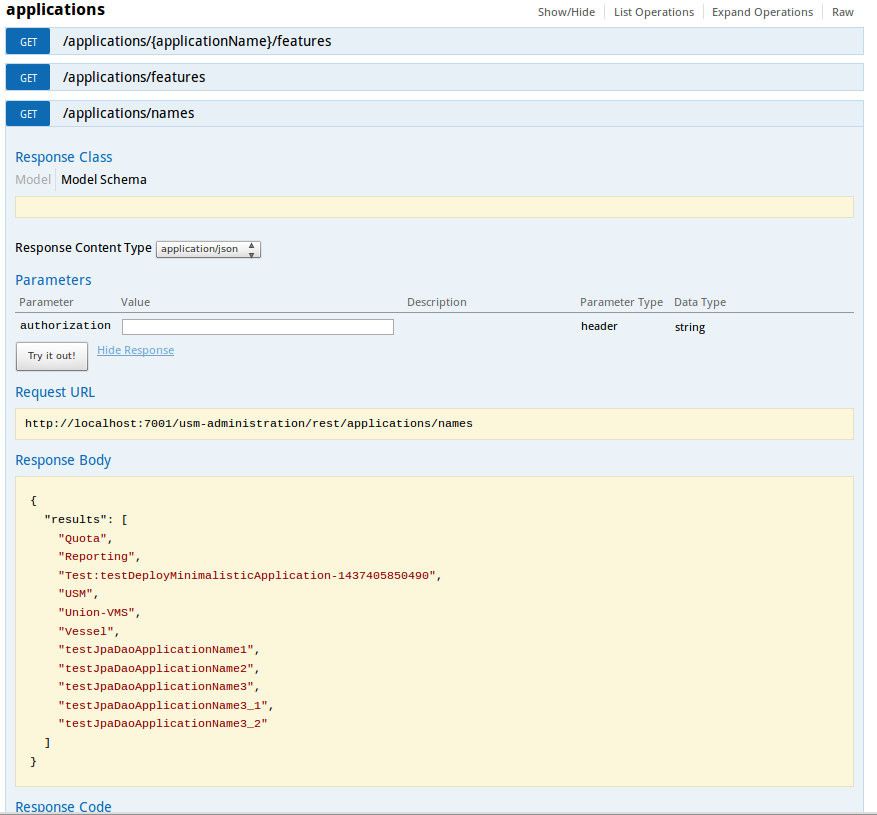
*Building and deploying the USM Back-end*

After executing the above three steps you should be able to the USM back-end RESTful web-services without providing any JWT token in the **authorisation** header field (customarily used for user authentication), simply by entering the service URL (e.g. <http://localhost:7001/usm-administration/rest/applications/names>) directly in your browser address bar.

# Samples

## USM Back-end access with ECAS authentication

The below example highlights the possibility to use one of the USM back-end RESTful web-services (nicely documented by-the-way), without providing a JWT token in the **authorisation** header field.



## WebLogic domain configuration

### Sample WebLogic start-up script

The below example highlights the mandatory configuration to disable SSLv3, defined in the Weblogic server start-up script ($DOMAIN\_HOME/bin/startWebLogic.sh)

|  |
| --- |
| # START WEBLOGIC  echo "starting weblogic with Java version:"  ${JAVA\_HOME}/bin/java ${JAVA\_VM} -version  **export ECAS\_OPTIONS="-Dweblogic.wsee.client.ssl.usejdk=true -DUseSunHttpHandler=true -Dhttps.protocols=TLSv1 -Dweblogic.security.SSL.enableJSSE=true -Dweblogic.ssl.JSSEEnabled=true -Dweblogic.security.SSL.protocolVersion=TLS1"**  **export JAVA\_OPTIONS=$JAVA\_OPTIONS:$ECAS\_OPTIONS**  if [ "${WLS\_REDIRECT\_LOG}" = "" ] ; then  echo "Starting WLS with line:"  echo "${JAVA\_HOME}/bin/java ${JAVA\_VM} ${MEM\_ARGS} -Dweblogic.Name=${SERVER\_NAME} -Djava.security.policy=${WL\_HOME}/server/lib/weblogic.policy ${JAVA\_OPTIONS} ${PROXY\_SETTINGS} ${SERVER\_CLASS}"  ${JAVA\_HOME}/bin/java ${JAVA\_VM} ${MEM\_ARGS} -Dweblogic.Name=${SERVER\_NAME} -Djava.security.policy=${WL\_HOME}/server/lib/weblogic.policy ${JAVA\_OPTIONS} ${PROXY\_SETTINGS} ${SERVER\_CLASS}  else  echo "Redirecting output from WLS window to ${WLS\_REDIRECT\_LOG}"  ${JAVA\_HOME}/bin/java ${JAVA\_VM} ${MEM\_ARGS} -Dweblogic.Name=${SERVER\_NAME} -Djava.security.policy=${WL\_HOME}/server/lib/weblogic.policy ${JAVA\_OPTIONS} ${PROXY\_SETTINGS} ${SERVER\_CLASS} >"${WLS\_REDIRECT\_LOG}" 2>&1  fi |

*Sample Weblogic start-up script*

### Sample WebLogic configuration

The below example displays the suggested configuration for *EcasIdentityAsserterV2* in the Weblogic domain configuration file ($DOMAIN\_HOME/config/config.xml)*.*

|  |
| --- |
| <sec:authentication-provider xmlns:sch="https://www.cc.cec/cas/schemas" xsi:type="sch:**ecas-identity-asserter-v2Type**">  <n1:name xmlns:n1="http://www.bea.com/ns/weblogic/90/security">EcasIdentityAsserterV2</n1:name>  <sch:control-flag>**OPTIONAL**</sch:control-flag>  <sch:ecas-base-url>**https://svm-midway.athens.intrasoft-intl.private:7012**</sch:ecas-base-url>  <sch:accept-strengths>**BASIC**</sch:accept-strengths>  <sch:assurance-level>**LOW**</sch:assurance-level>  <sch:ecas-server-direct-host-name>**svm-midway.athens.intrasoft-intl.private**</sch:ecas-server-direct-host-name>  <sch:ecas-server-direct-one-way-ssl-port>**7012**</sch:ecas-server-direct-one-way-ssl-port>  </sec:authentication-provider> |

*Suggested configuration for the EcasIdentityAsserterV2*

The below example highlights the mandatory configuration for *DefaultAuthenticator* in the Weblogic domain configuration file ($DOMAIN\_HOME/config/config.xml)*.*

|  |
| --- |
| <sec:authentication-provider xsi:type="wls:**default-authenticatorType**">  <sec:control-flag>**OPTIONAL**</sec:control-flag>  <wls:minimum-password-length>5</wls:minimum-password-length>  </sec:authentication-provider> |

*Required configuration for the DefaultAuthenticator*