User authentication

Summary: This section describes how to configure **user authentication** for the CAST dashboards.

Introduction

The CAST dashboards have various **authentication modes** available for use:

Mode	Description
Default authentication	This mode is **active by default** and relies on simple username/password authentication defined in the **application-security-default.xml** configuration file within the web application.
Standard LDAP	This mode is **inactive by default** and allows users to authenticate with a **standard LDAP server**. For example: • Basic Active Directory domain controllers • Other members of the Active Directory family such as ADAM / AD LDS • Non-Microsoft directories such as Apache, Oracle, Novell etc. Note that the configuration of Standard LDAP mode requires detailed knowledge of your environment's LDAP implementation - i.e. an LDAP administrator should help with this.
SAML	This mode is **inactive by default** and allows users to authenticate via **SAML**. Note that the configuration of SAML mode requires detailed knowledge of your environment - i.e. SAML administrator should help with this.

- CAST recommends using **Standard LDAP** because this avoids having to manually manage individual usernames and passwords via the **Default authentication** mode.
- Only **one mode** can be active at a time.
- If you are using **CAST Dashboards 2.5**, this step is not required and you can skip it - authentication is configured in the wizard installer.

Authentication mode activation

The activation of the available authentication modes is governed by a .**properties** configuration file within the web application:

```
WAR 1.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\security.properties

WAR  2.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\classes\application.properties

ZIP  2.x
<unpacked_zip>\configurations\application.properties
```

In the **.properties** configuration file, activation is handled by the following line. In the "out of the box" state, the **default** security mode is active as shown below. Only **one mode** can be active at a time:

```
security.mode=default
```

To activate a mode, change the following line to the required security mode:

- **default** -> The initial mode when you deploy the application
- **ldap** -> Set this mode for authentication over LDAP(S)
- **saml** -> Set this mode for authentication over SAML

For example, to change from the **Default authentication** security mode to **Standard LDAP:**

```
security.mode=ldap
```

Following any changes you make, **save the .properties** file and then **restart** the web application so that the changes are taken into account.

Default authentication mode

This mode is enabled **by default** "out of the box" with the following username and case sensitive password (usernames are NOT case sensitive):

Username	Password	Group	Notes
guest	my_password	NoGroup	See the section below for more information about **groups**.

If you would like to alter the password for this existing username, or you would like to add additional username/passwords, you need to modify the following file with a text editor:

```
WAR 1.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\users.properties

WAR  2.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\classes\users.properties

ZIP  2.x
<unpacked_zip>\configurations\users.properties
```

This file contains the following line which defines the usernames that can access the Dashboard:

```
guest=my_password,NoGroup,enabled
```

- A user is defined on a single line.
- If the username or password contains special characters (non US-ANSI characters) such as é,è,à,ç,ù, etc., you must ensure that your text editor saves the **user.properties** file with **iso-8859-1** encoding.
- The **enabled/disabled** entry must ALWAYS be placed at the **very end of the line**.
- A user added in this file **does not have permission to access any data**, therefore you must either grant this user specific **roles** or grant access to the data via an **authorization**.

Users

Adding a new user

To add a new user, add an additional line into the **users.properties** file. The following examples will add in a username "jhu" with the password "password" with **no group configuration**:

```
guest=my_password,NoGroup,enabled
jhu=password,NoGroup,enabled
```

Following any changes you make, **save the users.properties file** and then **restart** your application server so that the changes are taken into account.

Note that when you add a new user, the user will initially **not have access to any data** - an error will be displayed when the user attempts to log in. You must therefore either:

- configure an authorization (see **Data authorization**) specific to the new user to grant the user access to data
- or grant the user (or the group the user belongs to) the **ADMIN role** which has access to all data and therefore does not require an authorization configuration (but you should use this role with caution!)

Removing an existing user

To remove an existing user, remove the corresponding line from the **users.properties file**. Following any changes you make, **save the users.properties file** and then **restart** your application server so that the changes are taken into account.

Editing an existing user

To edit an existing user, edit the corresponding line in the **users.properties file**. Following any changes you make, **save the users.properties file** an d then **restart** your application server so that the changes are taken into account.

Disabling a user without removing it from the users.properties file

To disable a user, change the **enabled** parameter to **disabled**:

```
jhu=password,NoGroup,disabled
```

Following any changes you make, **save the users.properties file** and then **restart** your application server so that the changes are taken into account.

User groups

Users can be grouped together to facilitate authorization assignments (see **Data authorization**) - for example, a set of users can be assigned to a group and that group can then be authorized to view the required data instead of having to authorize individual users. Groups are defined in the following file:

```
WAR 1.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\users.properties

WAR  2.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\classes\users.properties

ZIP  2.x
<unpacked_zip>\configurations\users.properties
```

Adding a new group

In this example we will add the group "**CIO**" and associate the existing user "**jhu**" to that group:

```
jhu=password,NoGroup,enabled
```

Replace the **NoGroup** entry with the name of the group "**CIO**", ensuring that **enabled** is always at the end of the line:

```
jhu=password,CIO,enabled
```

If other users should also be members of this group, add them in the same way:

```
jhu=password,CIO,enabled
dch=password,CIO,enabled
```

A user can be a member of several groups. The following defines the existing user **"jhu"** as member of the **"CIO"** and **"Users"** groups - i.e. comma separated group names:

```
jhu=password,CIO,Users,enabled
```

Following any changes you make, **save the users.properties file** and then **restart** your application server so that the changes are taken into account.

Note that when a group name is defined in the users.properties file, the group is automatically created. The group does not need to be declared or defined anywhere else.

Standard LDAP mode

Note that the configuration of the Standard LDAP mode requires detailed knowledge of your environment's LDAP implementation - i.e. an LDAP administrator should help with this.

This mode is **not enabled by default** "out of the box". It may be used with any LDAP compatible corporate directory. It allows users to login to the dashboard with their corporate LDAP credentials. LDAP groups can also be used for **authorization assignments** and for **role assignments**. CAST has provided place holder parameters, so you must change these before authentication will work correctly. To do so, modify the following configuration file within the web application:

```
WAR 1.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\security.properties

WAR  2.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\classes\application.properties

ZIP  2.x
<unpacked_zip>\configurations\application.properties
```

This file contains the following section which defines the required parameters:

```
WAR 1.x

# Parameters for ldap mode
# ------------------------
security.ldap.url=ldap://directory.example.com/
security.ldap.account.dn=cn=serviceaccount,dc=example,dc=com
security.ldap.account.password=password
security.ldap.account.key=
security.ldap.usersearch.base=dc=example,dc=com
security.ldap.usersearch.filter=(&(objectClass=user)(sAMAccountName={0}))
security.ldap.groupsearch.base=dc=example,dc=com
security.ldap.groupsearch.filter=(&(objectClass=group)(member={0}))

WAR  2.x and ZIP  2.x

## SPRING SECURITY LDAP CONFIG
# LDAP url, in the form ldap://HOST:PORT
security.ldap.url=ldap://directory.example.com/
# The ldap base where users and groups can be found
security.ldap.base=dc=example,dc=com
# The DN for accessing the LDAP repository
security.ldap.manager.dn=CN=serviceaccount,OU=RESOURCES,OU=FR,DC=example,DC=com
# The associated password. You can encrypt this using the aip encryption tool
security.ldap.manager.password=password
# The attribute containing the user's login
# NOTE: Unused, it might be useful later to map a DN to a user's full name
# security.ldap.user.nameattribute=sAMAccountName
# The base for the user search which can be left empty
# NOTE: No need to add the initial base, it will be taken into account
security.ldap.usersearch.base=OU=RESOURCES,OU=FR
# The Filter for user search
security.ldap.usersearch.filter=(&(objectClass=user)(sAMAccountName={0}))
# The attribute of a group entry to obtain the role name
security.ldap.groupsearch.roleAttribute=cn
# The base for the group search
# NOTE: No need to add the initial base, it will be taken into account
security.ldap.groupsearch.base=
# The filter to use for the group search
security.ldap.groupsearch.filter=(&(objectClass=group)(member={0}))
# Performance fix for nested groups on AD
#security.ldap.groupsearch.filter=(&(objectClass=group)(member:1.2.840.113556.1.4.1941:={0}))
#security.ldap.groupsearch.maxSearchDepth=1
```

You first need to change the following parameters marked in red to match the URL and the service account required to connect to your LDAP directory:

WAR 1.x	• **security.ldap.url**=**ldap://directory.example.com/** • **security.ldap.account.dn**=**cn=serviceaccount,dc=example,dc=com** • **security.ldap.account.password**=**password**
WAR 2.x and ZIP 2.x	• **security.ldap.url**=**ldap://directory.example.com/** • **security.ldap.base**=**dc=example,dc=com** • **security.ldap.manager.dn**=**cn=serviceaccount,ou=resources,ou=fr,dc=example,dc=com** • **security.ldap.account.password**=**password**

It is possible to encrypt the LDAP service account password to avoid entering values in clear text, please see **Encrypt login and password for database and LDAP** for more information about this.

You then need to change the following parameters marked in red related to searching the **users/groups** in your directory - specifically if you are leveraging groups to manage **data authorization**:

WAR 1.x	• **security.ldap.usersearch.base** > The root tree node from which **users** should be searched • **security.ldap.usersearch.filter** > The criteria for searching users: you must change "**user**" and "**sAMAccountName**" to match your directory structure • **security.ldap.groupsearch.base** > The root tree node from which **groups** should be searched • **security.ldap.groupsearch.filter** > The criteria for searching users: you must change "**group**" and "**member**" to match your directory structure
WAR 2.x and ZIP 2.x	• **security.ldap.usersearch.base** > The root tree node from which **users** should be searched • **security.ldap.usersearch.filter** > The criteria for searching users: you must change "**user**" and "**sAMAccountName**" to match your directory structure • **security.ldap.groupsearch.roleAttribute** > The attribute of a **group entry** to obtain the role name • **security.ldap.groupsearch.base** > The root tree node from which **groups** should be searched • **security.ldap.groupsearch.filter** > The criteria for searching users: you must change "**group**" and "**member**" to match your directory structure

For some LDAP servers:

- the **security.ldap.usersearch.filter** parameter may take the following form "security.ldap.usersearch.filter=(&(objectClass=**inetOrgPerson**)(**uid**={0}))"
- the **security.ldap.groupsearch.filter** parameter may take the following form "security.ldap.groupsearch.filter=(&(objectClass=**groupOfNames**)(**member**={0}))"

Following any changes you make, save the **.properties** file and then **restart** your application server so that the changes are taken into account. Users should now be able to access the dashboard using their corporate LDAP login - authentication is therefore the responsibility of the corporate LDAP directory.

Note that:

- enabling **Standard LDAP** mode will **disable** the **Default Authentication** mode
- By default, the log mechanism is not configured to provide any logging information to debug Active Directory authentication issues - if you have encountered issues activating Active Directory authentication, please enable DEBUG mode as described in **Configuring the Log and Audit Trail**.
- by default LDAP users will initially **not have access to any data** - an error will be displayed when the user attempts to log in. You must therefore either:
  - configure an Authorization (see **Data authorization**) specific to the user (or to a group the user belongs to) to grant the user access to data
  - or grant the user (or the group the user belongs to) the **ADMIN role** which has access to all data and therefore does not require an authorization configuration (but you should use this role with caution!)

Notes about Groups

- Users can be grouped together to facilitate authorization assignments (see **Data authorization**) - for example, a set of users can be assigned to a group and that group can then be authorized to view the required data instead of having to authorize individual users. In Standard LDAP mode, Groups are retrieved directly from the LDAP directory as configured in the .properties file.
- **Nested groups are supported**, both for authorization assignments (see **Data authorization**) and for **role assignments**. For instance, if user **jdoe** is member of **groupA**, which is member of **groupB** which is used to define an authorization or role, then **jdoe** will be attributed the **groupB** authorizations/roles.

Using LDAPS (LDAP over SSL)

If your LDAP server requires that you use LDAPS (LDAP over SSL) then you must ensure that the following is done:

- Use a **ldaps://** URL in the **security.ldap.url** parameter in the .properties file.
- The LDAP server's SSL certificate or a parent certificate (CA) also needs to be imported into the truststore for the default Java implementation (i.e. JRE) used by the web application server. To do this, you need to use the **keytool** command line utility (provided with the JRE - see **https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html** for more information) on the workstation on which the web application server is running. For example:

```
%JAVA_HOME%\bin\keytool -importcert -alias [alias] -keystore [path-to-jre/lib/security/cacerts] -file [path-to-certificate-file]
```

Note that you may be prompted for the password of the keystore. By default this password is set to "**changeit**".

SAML mode

This mode is **not enabled by default** "out of the box".

Prerequisites

Before you can configure your CAST AIP web applications to use SAML authentication, the following prerequisites must already be in place:

CAST AIP web applications deployed and functioning	The CAST AIP web applications must be deployed and functioning before you can proceed. In particular you must ensure that any **roles** and **data authorizations** are already configured.
Apache Tomcat / standalone ZIP configured for HTTPS	The Apache Tomcat host server / standalone ZIP deployments must be configured to use the HTTPS protocol. See: • **Configuring Apache Tomcat to use secure https protocol** • **Modify the user access port for 2.x JAR or ZIP deployments**
FederationMetadata.xml	This file must be provided by your IT administrators before you can proceed.
Key pair generation	A public/private key pair must be generated on the Apache Tomcat host server in a dedicated keystore to allow encrypted communication with the backend identity provider. See below for more information. **Note:** Dashboard supports the SAML Keystore file, which is generated using the SHA256 algorithm.

Supported versions of SAML

Version	Supported
2.0	✓
1.1	✗
1.0	✗

Configuration process

Request FederationMetadata.xml

You must request the **FederationMetadata.xml** file from your IT administrators. When you have received the file, you should store it in a location that can be accessed from the web application, for example, within the Apache Tomcat installation location or within the unpacked ZIP. For example:

```
Windows: D:/apache-tomcat/conf/FederationMetadata.xml
Linux: /opt/apache-tomcat/conf/FederationMetadata.xml
```

Key pair generation

A public/private key pair must be generated on the Apache Tomcat host server in a dedicated keystore to allow encrypted communication with the backend identity provider. This keystore should be specific to the SAML configuration. To do so, you need to use the **keytool** command line utility provided with the JRE (see for example **https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html** for more information about how to use this tool (this link refers to Oracle's Java implementation, and other JRE providers will have their own instructions, which you should use)) on the workstation on which the web application server is running. For example:

```
%JAVA_HOME%\keytool -genkeypair -alias <some-alias> -keyalg RSA -sigalg SHA256withRSA -keysize 2048 -
validity 3650 -keypass <keypass> -keystore <samlKeystore.jks> -storepass <storepass>
```

Where:

-alias	Choose an alias that is specific to the key pair.

-keyalg, -sigalg, -keysize, -validity	Choose these options according to your own requirements (see for example **https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html** for more information about how to use this tool (this link refers to Oracle's Java implementation, and other JRE providers will have their own instructions, which you should use))
-keypass	This configured a password that is used to protect the private key of the generated key pair. The value must be at least 6 characters.

-keystore	Choose a keystore location in which to store the key pair, for example:
	`Windows: D:/keystore/samlKeystore.jks`
	`Linux: /opt/apache-tomcat/conf/samlKeystore.jks`

-storepass	Choose a password to protect the keystore.

Activate and configure the authentication mode in the web application

Activation and configuration of the SAML authentication mode is governed by a **.properties** configuration file within the web application:

```
WAR 1.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\security.properties

WAR  2.x
CATALINA_HOME\webapps\<dashboard>\WEB-INF\classes\application.properties

ZIP  2.x
<unpacked_zip>\configurations\application.properties
```

To activate the SAML authentication mode, change the following line. For example, to change from the **Default authentication** security mode to **SAML**, do as follows. Change:

```
security.mode=default
```

to:

```
security.mode=saml
```

Save the **.properties** file.

Configure SAML authentication

Find the SAML parameters section in the .**properties** configuration file and modify each uncommented line to match the items you have already configured. Save the **.properties** file when complete.

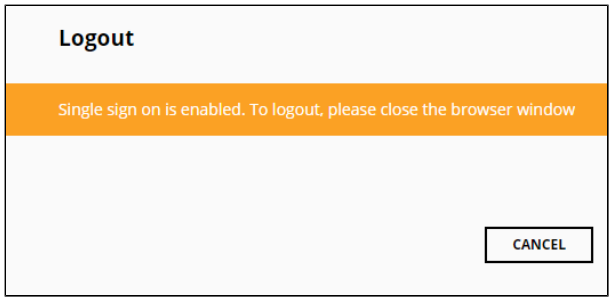
```
WAR 1.x

# Parameters for saml mode
# ------------------------
# idp metadata file
security.saml.idp.metadata.location=file:/opt/apache-tomcat/conf/FederationMetadata.xml
# attribute name for group in saml response
security.saml.idp.metadata.group.attribute.name=http://schemas.xmlsoap.org/claims/Group
# Key store path
security.saml.keystore.path=file:/opt/apache-tomcat/conf/myKeystore.jks
# key store password
security.saml.keystore.password=changeit
# Key alias
security.saml.key.alias=somealias
# Key password
security.saml.key.password=changeit
# is Single Logout implemented in the customer IDP ?
security.saml.single.logout=true

WAR and ZIP  2.x

## SPRING SECURITY SAML CONFIG
# or a classpath resource using "classpath:myMetadataFile.xml" for example
# NB : when using an HTTPS metadata source, you must first add the public certificate to the keystore
security.saml.metadata.source=
# Specify the filename of the keystore to use for the SAML certificates
# The file must be placed inside the security.config.folder
security.saml.keystore.filename=sample.keystore
# Specify the default alias in the keystore for the certificate
security.saml.keystore.default-alias=somealias
# Specify the keystore and alias password
security.saml.keystore.password=changeit
# The XML attribute containing the user's name
# If this attribute is missing or empty, the user ID will be used
security.saml.attribute.username=
# The XML attribute containing the user's group in the SAML response
security.saml.attribute.group=
# is Single Logout implemented in the customer IDP ?
security.saml.single.logout=true
```

1.x = security.saml.idp.metadata.location **2.x = security.saml.metadata.source**	Location of the **FederationMetadata.xml** file. For example: • **Windows:** D:/apache-tomcat/conf/FederationMetadata.xml • **Linux:** /opt/apache-tomcat/conf/FederationMetadata.xml
1.x = security.saml.idp.metadata.group.attribute.name **2.x = security.saml.attribute.username** **2.x = security.saml.attribute.group**	Name of the **username** and/or **group attribute** (please discuss with your IT administrators about this option). Note that for group attributes, you should normally use (to be confirmed with your SAML admins): **http://schemas.xmlsoap.org/claims/Group**, for example: • 1.x = security.saml.idp.metadata.group.attribute.name=**http://schemas.xmlsoap.org/claims/Group** • 2.x = security.saml.attribute.group=**http://schemas.xmlsoap.org/claims/Group**
1.x = security.saml.keystore.path **2.x = security.saml.keystore.filename**	Location of the **keystore** you created previously.
security.saml.keystore.password	The keystore **password** you created previously (corresponds to the **-storepass** option for keytool)

1.x = security.saml.key.alias **2.x = security.saml.keystore.default-alias**	The keystore **alias** you created previously.
1.x = security.saml.key.password **2.x = security.saml.keystore.password**	The key **password** you created previously (corresponds to the **-keypass** option for keytool).
security.saml.single.logout	If SAML authentication is in operation, but no Single Logout service is provided in the IdP, you can force the dashboard to handle this situation gracefully and display a message explaining what to do by setting the option to **true** (default): 

Restart Apache Tomcat / ZIP file

Now restart your Apache Tomcat server or the web application ZIP file so that the changes you made are taken into account.

Modify application-security-saml.xml file - only required in 2.x releases

If you are using **CAST Dashboards 2.0**, please ensure that you modify the **application-security-saml.xml** file located here:

```
WAR  2.x
CATALINA_HOME\webapps\<deployed_war>\WEB-INF\classes\security

ZIP  2.x
<unpacked_zip>\configurations\security
```

First you need to update the "`metadataGenerator`" to match the location of your Dashboard deployment. Locate the following section:

```
<!-- Define basic information regarding WEBI as a Service Provider -->
<bean id="metadataGenerator" class="org.springframework.security.saml.metadata.MetadataGenerator">
        <property name="entityId" value="https://localhost:8080/saml/metadata"/>
        <property name="extendedMetadata" ref="extendedMetadata"/>
        <property name="includeDiscoveryExtension" value="false"/>
        <property name="keyManager" ref="keyManager"/>
</bean>
```

Change the line `<property name="entityId" value="https://localhost:8080/saml/metadata"/>` to match your own deployment. Some examples for the "value" parameter are given below:

```
<property name="entityId" value="https://<my_server_dns_name>/saml/metadata"/>
<property name="entityId" value="https://<my_server_dns_name>/<deployed_war>/saml/metadata"/>
<property name="entityId" value="https://<my_server_dns_name><:custom_ssl_port>/<deployed_war>/saml/metadata"
/>
<property name="entityId" value="https://<my_server_ip_address>/saml/metadata"/>
<property name="entityId" value="https://<my_server_ip_address>/<deployed_war>/saml/metadata"/>
<property name="entityId" value="https://<my_server_ip_address><:custom_ssl_port>/<deployed_war>/saml
/metadata"/>
```

Next update the `successRedirectHandler` to configure the redirect after a successful login. Locate the following section:

```
<bean id="successRedirectHandler"
        class="org.springframework.security.web.authentication.SavedRequestAwareAuthenticationSuccessHandler"
>
        <property name="alwaysUseDefaultTargetUrl" value="true"/>
        <!-- the default landing url after login successful -->
        <property name="defaultTargetUrl" value="/engineering/index.html"/>
</bean>
```

Change the line `<property name="defaultTargetUrl" value="/engineering/index.html"/>` to match your own deployment. For example:

```
Engineering Dashboard: <property name="defaultTargetUrl" value="/engineering/index.html"/>
Health Dashboard: <property name="defaultTargetUrl" value="/portal/index.html"/>
Combined Health/Engineering: <property name="defaultTargetUrl" value="/welcome.html"/>
```

Save the file and restart your Apache Tomcat server or the web application ZIP file so that the changes you made are taken into account.

Generate spring_metadata

Now browse to the following URL to generate the **spring_metadata**:

```
WAR file deployment:
https://tomcat/<deployed_war>/saml/metadata

ZIP fil deployment:
https://localhost/saml/metadata
```

This will download a file called **spring_saml_metadata.xml**. Send this file to your IT administrators who will then register it in the ADFS allowing users to login to the web application.

Tips

- Attempting to use the login button in the **static/default.html** page in the CAST Health Dashboard, Engineering Dashboard and the RestAPI will fail when SAML mode is configured: this button is only configured to use **basic authentication**. If you need to use any of the options provided in the **static/default.html** page (which all require a login), you must ensure that you login to the dashboard in the conventional way, and THEN access the **static/default.html** page in your browser.
- ADFS are very sensitive: if badly set, authentication will fail.
- By default, the log mechanism is not configured to provide any logging information to debug SAML authentication issues - if you have encountered issues activating SAML authentication, please enable DEBUG mode as described in **Configuring the Log and Audit Trail**.

Notes about Groups

- SAML groups can also be used for **authorization assignments** (for example, a set of users can be assigned to a group and that group can then be authorized to view the required data instead of having to authorize individual users) and for **role assignments**. In SAML mode, Groups are retrieved directly from the SAML directory.
- **Nested groups are supported**, both for **authorization assignments** and for **role assignments**. For instance, if user **jdoe** is member of **groupA**, which is member of **groupB** which is used to define an authorization or role, then **jdoe** will be attributed the **groupB** authorizations /roles.