SafeNet Authentication Service

Integration Guide

Using RADIUS Protocol for Apereo CAS

gemalto
security to be free

Document Part Number: 007-013673-001, Rev. A

Release Date: August 2017

Contents

Third-Party Software Acknowledgement

This document is intended to help users of Gemalto products when working with third-party software, such as Apereo Central Authentication Service (CAS).

Material from third-party software is being used solely for the purpose of making instructions clear. Screen images and content obtained from third-party software will be acknowledged as such.

Description

SafeNet Authentication Service (SAS) delivers a fully automated, versatile, and strong authentication-as-a-service solution.

With no infrastructure required, SafeNet Authentication Service provides smooth management processes and highly flexible security policies, token choice, and integration APIs.

Apereo CAS is a single sign-on protocol for the web. Its purpose is to permit a user to access multiple applications while providing their credentials (such as user ID and password) only once. It also allows web applications to authenticate users without gaining access to the users' security credentials.

This document describes how to:

- Deploy multi-factor authentication (MFA) options in Apereo CAS using SafeNet one-time password (OTP) authenticators managed by SafeNet Authentication Service.
- Configure Apereo CAS to work with SafeNet Authentication Service in the RADIUS mode.

It is assumed that the Apereo CAS environment is already configured and working with static passwords prior to implementing multi-factor authentication using SafeNet Authentication Service.

Apereo CAS can be configured to support multi-factor authentication in several modes. The RADIUS protocol will be used for the purpose of working with SafeNet Authentication Service

Applicability

The information in this document applies to:

- **SafeNet Authentication Service (SAS)**—SafeNet's cloud-based authentication service
- **SafeNet Authentication Service – Service Provider Edition (SAS-SPE)**—A server version that is used by Service Providers to deploy instances of SafeNet Authentication Service
- **SafeNet Authentication Service – Private Cloud Edition (SAS-PCE)**—A server version that is used to deploy the solution on-premises in the organization

Environment

The integration environment that was used in this document is based on the following software versions:

- **SafeNet Authentication Service – Private Cloud Edition (SAS-PCE)**
- **Apereo CAS**—Version 4.2

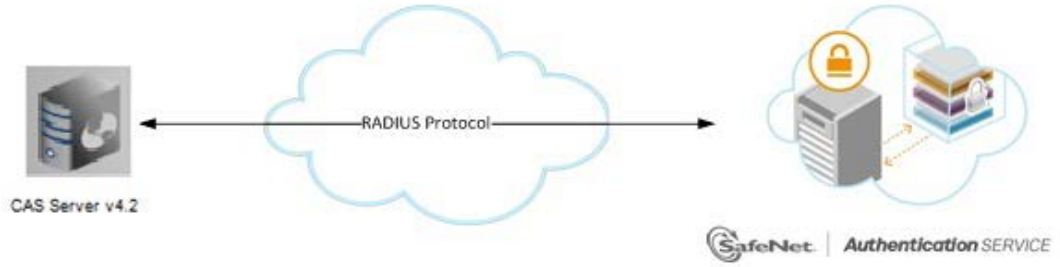
Audience

This document is targeted to system administrators who are familiar with Apereo CAS, and are interested in adding multi-factor authentication capabilities using SafeNet Authentication Service.

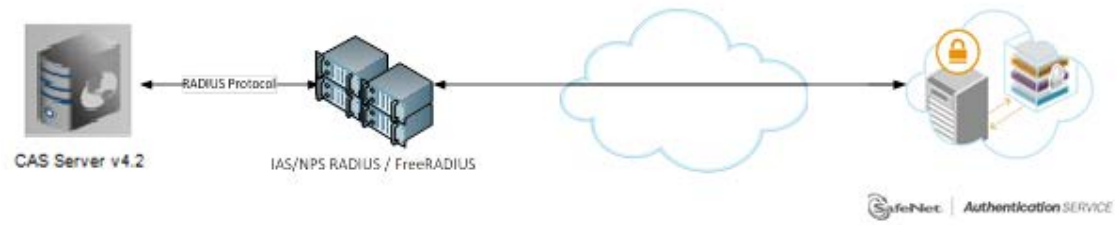
RADIUS-based Authentication using SafeNet Authentication Service Cloud

SafeNet Authentication Service (SAS) Cloud provides two RADIUS mode topologies:

- **SAS cloud hosted RADIUS service**—A RADIUS service that is already implemented in the SAS cloud environment and can be used without any installation or configuration requirements.



- **Local RADIUS hosted on-premises**—A RADIUS agent that is implemented in the existing customer's RADIUS environment. The agent forwards the RADIUS authentication requests to the SAS cloud environment. The RADIUS agent can be implemented on a Microsoft NPS/IAS or FreeRADIUS server.



This document demonstrates the solution using the SAS cloud hosted RADIUS service.

For more information on how to install and configure SAS Agent for IAS/NPS, refer to: http://www2.gemalto.com/sas-downloads/docs/007-012390-002_SAS_Agent_for_NPS_1.30_ConfigurationGuide_RevD.pdf

For more details on how to install and configure FreeRADIUS, refer to the *SafeNet Authentication Service FreeRADIUS Agent Configuration Guide*.

RADIUS-based Authentication using SafeNet Authentication Service-SPE and SafeNet Authentication Service-PCE

For both on-premises versions, SafeNet Authentication Service (SAS) can be integrated with the following solutions that serve as local RADIUS servers:

- **Microsoft Network Policy Server (MS-NPS)** or the legacy **Microsoft Internet Authentication Service (MS-IAS)**—SafeNet Authentication Service is integrated with the local RADIUS servers using a special on-premises agent called SAS Agent for Microsoft IAS and NPS.

  For more information on how to install and configure the SAS Agent for Microsoft IAS and NPS, refer to the following document:

  http://www2.gemalto.com/sas-downloads/docs/007-012390-002_SAS_Agent_for_NPS_1.30_ConfigurationGuide_RevD.pdf

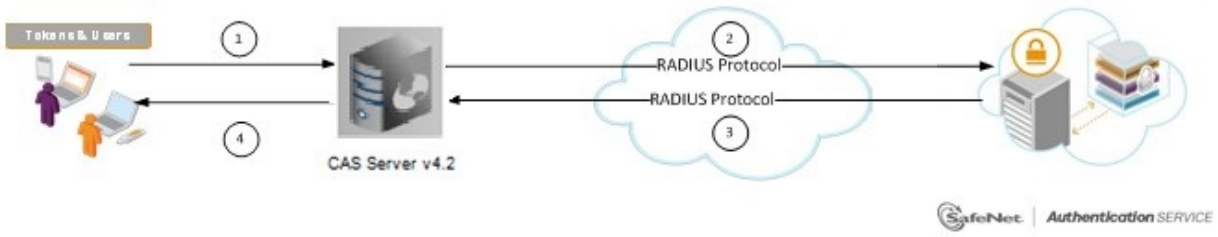
- **FreeRADIUS**—The SAS FreeRADIUS Agent is a strong authentication agent that is able to communicate with SAS through the RADIUS protocol.

  For more information on how to install and configure the SAS FreeRADIUS Agent, refer to the Gemalto Support Portal.

RADIUS Authentication Flow using SafeNet Authentication Service

SafeNet Authentication Service (SAS) communicates with a large number of VPN and access-gateway solutions using the RADIUS protocol.

The image below describes the data flow of a multi-factor authentication transaction for Apereo CAS.



1. A user attempts to log on to Apereo CAS using an OTP authenticator.
2. Apereo CAS sends a RADIUS request with the user's credentials to SafeNet Authentication Service (SAS) for validation.
3. The SAS authentication reply is sent back to Apereo CAS.
4. The user is granted or denied access to the Apereo CAS based on the OTP value calculation results from SAS.

RADIUS Prerequisites

To enable SafeNet Authentication Service (SAS) to receive RADIUS requests from Apereo CAS, ensure the following:

- End users can authenticate from the Apereo CAS environment with a static password before configuring the Apereo CAS to use RADIUS authentication.
- Ports 1812/1813 are open to and from Apereo CAS.
- A shared secret key has been selected. A shared secret key provides an added layer of security by supplying an indirect reference to a shared secret key. It is used by a mutual agreement between the RADIUS server and RADIUS client for encryption, decryption, and digital signatures.

Third Party Prerequisites

To run and build the CAS server, ensure that the following software are installed on the CAS server machine:

- Java jre v1.7 or later
- Apache Maven v3.3 or later
- Tomcat v7.0 or later (with SSL enabled) must be up and running

Configuring SafeNet Authentication Service

The deployment of multi-factor authentication using SafeNet Authentication Service (SAS) with Apereo CAS using RADIUS protocol requires the following:

- Creating Users Stores in SafeNet Authentication Service, page 7
- Assigning an Authenticator in SafeNet Authentication Service, page 8
- Adding Apereo CAS as an Authentication Node in SafeNet Authentication Service, page 8
- Checking the SafeNet Authentication Service RADIUS Address, page 11

Creating Users Stores in SafeNet Authentication Service

Before SafeNet Authentication Service (SAS) can authenticate any user in your organization, you need to create a user store in SAS that reflects the users that would need to use multi-factor authentication. User records are created in the SAS user store using one of the following methods:

- Manually, one user at a time, using the **Create User** shortcut
- Manually, by importing one or more user records via a flat file
- Automatically, by synchronizing with your Active Directory / LDAP server using the SAS Synchronization Agent

For additional details on importing users to SafeNet Authentication Service, refer to “Creating Users” in the *SafeNet Authentication Service Subscriber Account Operator Guide*:

http://www.safenet-inc.com/resources/integration-guide/data-protection/Safenet_Authentication_Service/Safenet_Authentication_Service__Subscriber_Account_Operator_Guide/

All SafeNet Authentication Service documentation can be found on the SafeNet Knowledge Base site.

Assigning an Authenticator in SafeNet Authentication Service

SafeNet Authentication Service (SAS) supports a number of authentication methods that can be used as a second authentication factor for users who are authenticating through Apereo CAS.

The following authenticators are supported:

- eToken PASS
- RB-1 Keypad Token
- KT-4 Token
- SafeNet Gold
- SMS Token
- MP-1 Software Token
- MobilePASS
- GrIDSure Authentication

Authenticators can be assigned to users in two ways:

- **Manual provisioning**—Assign an authenticator to users one at a time.
- **Provisioning rules**—The administrator can set provisioning rules in SAS so that the rules will be triggered when group memberships and other user attributes change. An authenticator will be assigned automatically to the user.

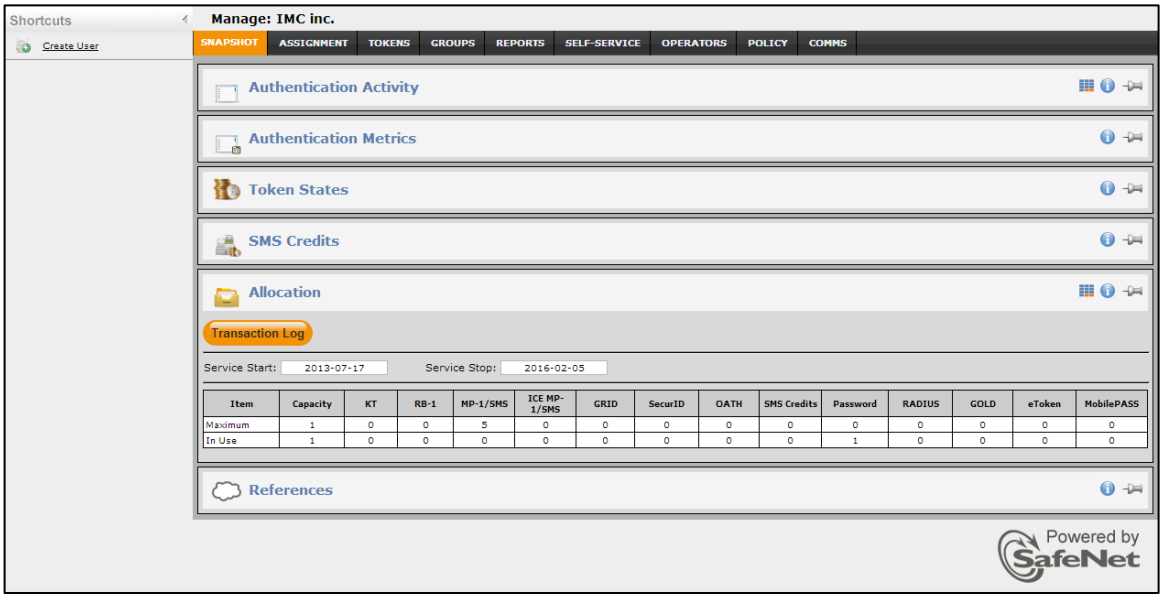
Refer to “Provisioning Rules” in the *SafeNet Authentication Service Subscriber Account Operator Guide* to learn how to provision the different authentication methods to the users in the SAS user store.

http://www.safenet-inc.com/resources/integration-guide/data-protection/Safenet_Authentication_Service/Safenet_Authentication_Service__Subscriber_Account_Operator_Guide/

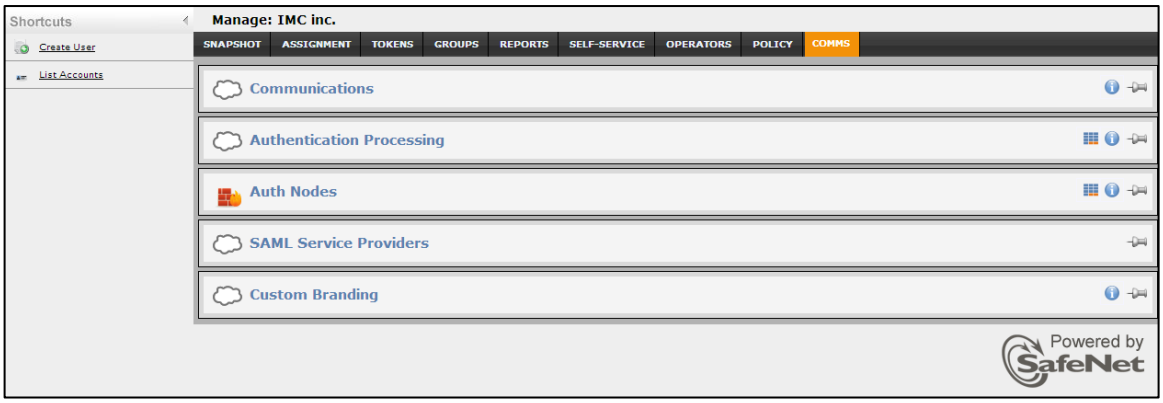
Adding Apereo CAS as an Authentication Node in SafeNet Authentication Service

Add a RADIUS entry in the SafeNet Authentication Service (SAS) **Auth Nodes** module to prepare it to receive RADIUS authentication requests from Apereo CAS. You will need the IP address of Apereo CAS and the shared secret to be used by both SAS and Apereo CAS.

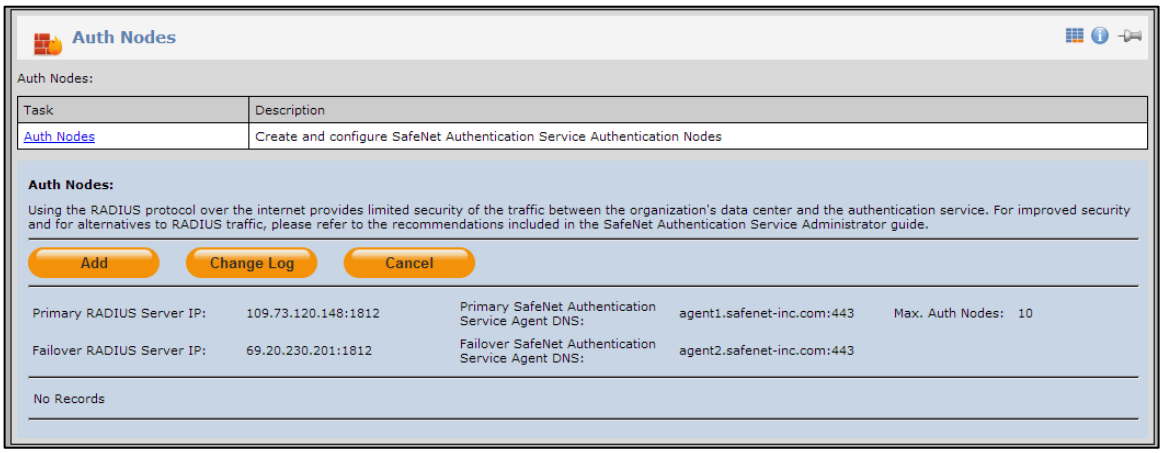
1. Log in to the SAS console with an Operator account.



2. Click the **COMMS** tab and then select **Auth Nodes**.

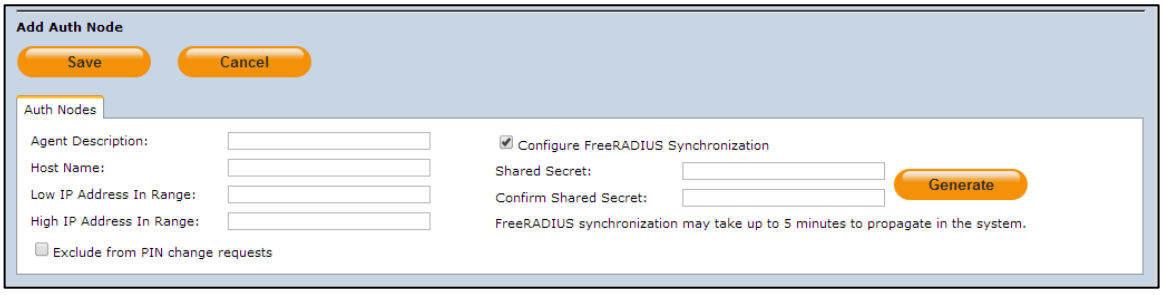


3. In the **Auth Nodes** module, click the **Auth Nodes** link.

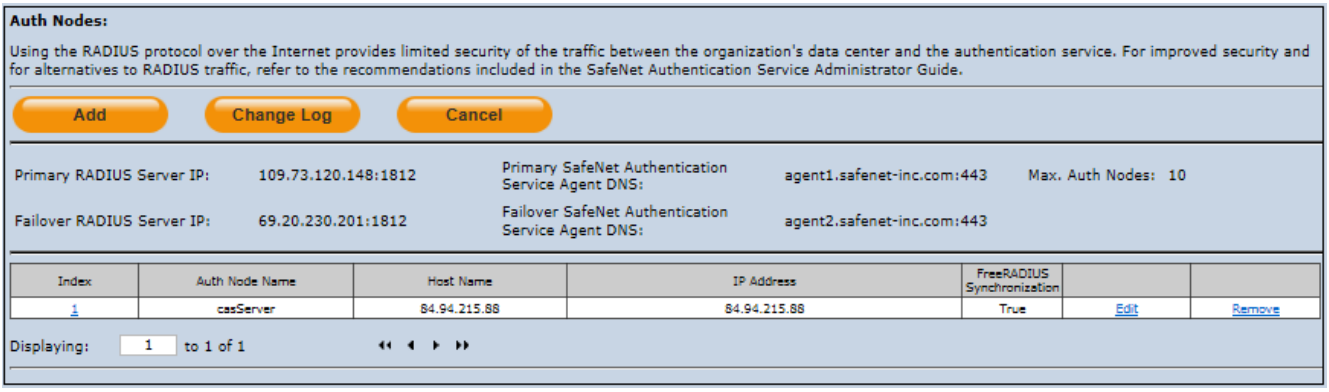


4. Under **Auth Nodes**, click **Add**.
5. In the **Add Auth Nodes** section, complete the following fields, and then click **Save**:

Agent Description	Enter a host description.
Host Name	Enter the name of the host that will authenticate with SAS.
Low IP Address In Range	Enter the IP address of the host or the lowest IP address in a range of addresses that will authenticate with SAS (in this case, a range of IP addresses is being used).
High IP Address In Range	Enter the highest IP address in a range of IP addresses that will authenticate with SAS (in this case, a range of IP addresses is being used).
Configure FreeRADIUS Synchronization	Select this option.
Shared Secret	Enter the shared secret key.
Confirm Shared Secret	Re-enter the shared secret key.



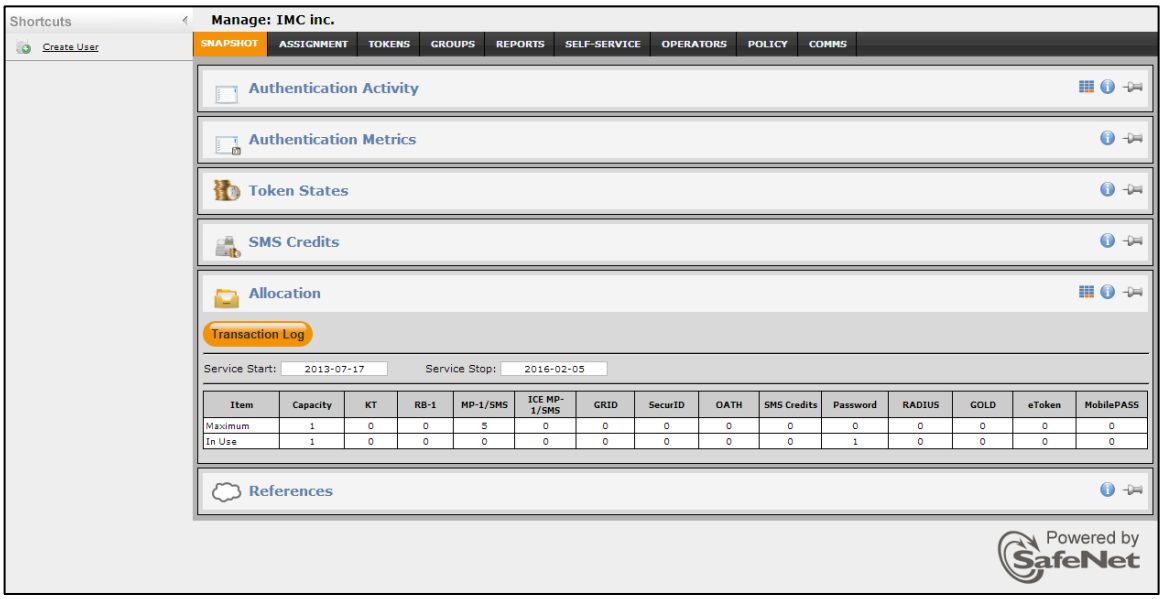
The authentication node is added to the system.



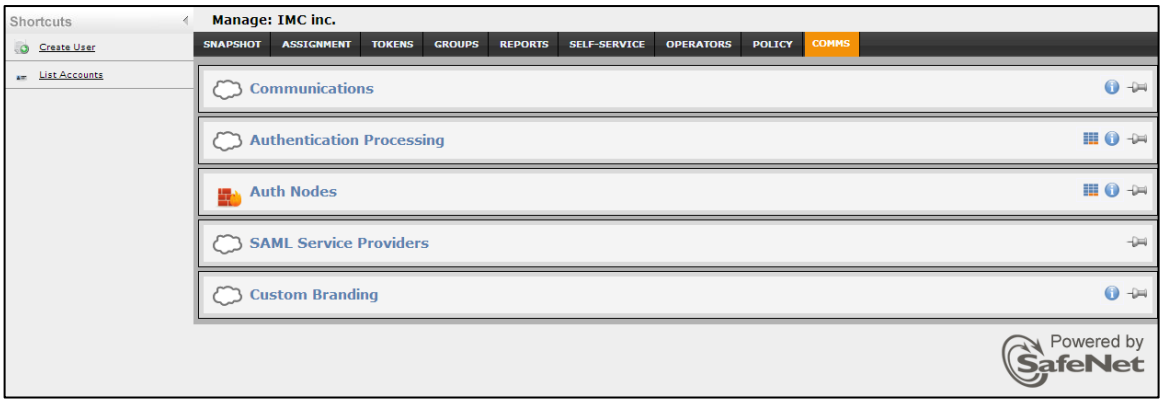
Checking the SafeNet Authentication Service RADIUS Address

Before adding SafeNet Authentication Service (SAS) as a RADIUS server in Apereo CAS, check its IP address. The IP address will be added to Apereo CAS as a RADIUS server at a later stage.

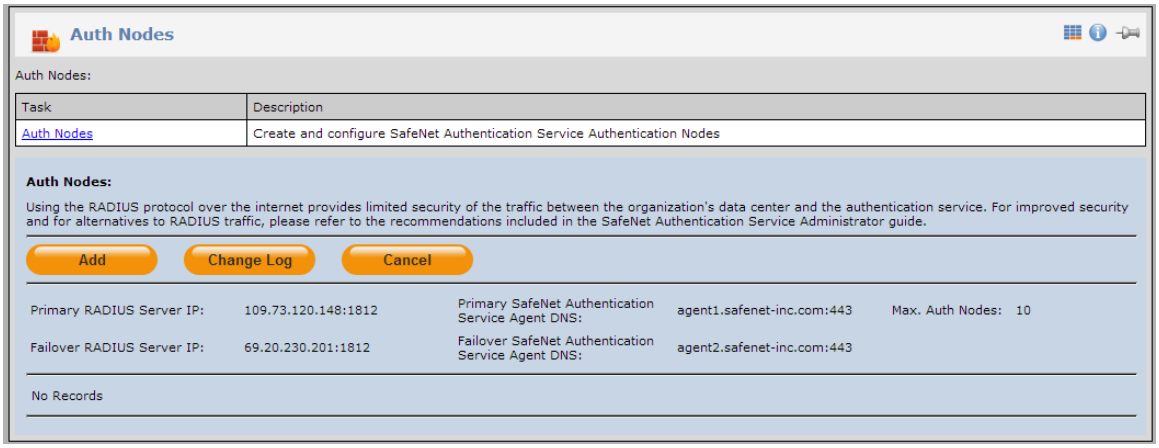
1. Log in to the SAS console with an Operator account.



2. Click the **COMMS** tab, and then select **Auth Nodes**.



3. In the **Auth Nodes** module, click the **Auth Nodes** link. The SAS RADIUS server details are displayed.



Configuring Apereo CAS v4.2

Perform the following steps to configure the Apereo CAS v4.2 server to use the RADIUS authentication:

1. On the CAS server machine, open the **pom.xml** file (for example, **…\casProject\pom.xml**) to enable the RADIUS protocol.
2. In the file, under the **<dependences>** tag, add the following dependency:

   <dependency>

   <groupId>org.jasig.cas</groupId>

   <artifactId>cas-server-support-radius</artifactId>

   <version>${cas.version}</version>

   </dependency>
3. Save the file.
4. In the **cas.properties** file (for example, **…\casProject\etc\cas\cas.properties**), perform the following steps:
   a. Under # **RADIUS Authentication Server**, enable the following properties by removing the # sign, and then set the following RADIUS configuration.

cas.radius.client.inetaddr	Enter SAS RADIUS server IP address.
cas.radius.client.port.acct	Enter the port number (for example, **1812**).
cas.radius.client.sharedsecret	Enter the shared secret.
cas.radius.server.protocol	Enter **PAP**.

   b. Locate the line starting with **accept.authn.users**.
   c. Comment out this line by adding # at the starting of the following line:

      #accept.authn.users=casuser::Mellon
   d. Save the file.

5. Build the CAS server (refer to “Appendix A: Building and Setting up the CAS Server” on page 17), and then deploy the build output file, **cas.war** (for example, **…\target\cas.war**) on the tomcat **webapps** folder (for example,**…\Tomcat 7.0\webapps**).
6. In the **deployerConfigContext.xml** file (for example, **…\Tomcat 7.0\webapps\cas\WEB-INF\ deployerConfigContext.xml**), locate the following line:

   alias="primaryAuthenticationHandler"
7. Replace the line with the following:
   <alias name="radiusAuthenticationHandler" alias="primaryAuthenticationHandler" />
8. Restart the tomcat server and then run CAS on the browser.

Customizing the Apereo CAS Login Page to Use the GrIDSure Token

Perform the following steps to customize the **casLoginView.jsp** and **messages.properties** file to use the GrIDSure token.

1. On CAS server machine, back up the following file:

   cas/WEB-INF/view/jsp/default/ui/casLoginView.jsp
2. In the **casLoginView.jsp** file, locate the following line:

   ** **
3. Add the following code after the line located in the previous step:

```
<script>
function showGrid()
{
var gridMakerURL =
"https://10.164.44.214/blackshieldss/O/O1LL7DKH5L/index.aspx?getChallengeImage=true&userNam
e=";
        var uname = document.getElementById('username');
        if(uname)
{
                if (uname.value == "")
{
alert("<spring:message code="screen.welcome.alert.userNameNotDefined" javaScriptEscape='true'
/>");}
                else{
                        var obj = document.getElementById('fm1');
obj.innerHTML += '<br><br><img border="1" src="' + gridMakerURL + uname.value + '">';
                document.getElementById("username").value = uname.value;
                document.getElementById("username").readOnly=true;
                document.getElementById("username").style.backgroundColor='gray'
                document.getElementById('password').labels[0].innerHTML = "<spring:message
code="screen.welcome.label.otp" javaScriptEscape='true' />";} }
```

</script>

4. In the code added in the previous step, replace the highlighted content (**https://10.164.44.214/blackshieldss/O/O1LL7DKH5L/index.aspx**) with the **SAS Self-Service** URL.
5. After the line starting with **<input class="btn-reset**, add the following:

   <input class="btn-submit" type="button" name="getOTP" accesskey="g" value="<spring:message code="screen.welcome.button.getGrid" />" onclick="showGrid()" tabindex="8">
6. Save the file.
7. In the **messages.properties** file (for example, **…\Tomcat 7.0\webapps \cas\WEB-INF\classes\messages.proterties**), add the following at the end:

   ## To support SAS grid token##

   screen.welcome.label.otp=Otp:

   screen.welcome.button.getGrid=GetGrid

   screen.welcome.alert.userNameNotDefined = User name required

> **NOTE:** The **messages.properties** file is used for the default language (English). For any other language, customize the language-specific file.

8. Save the file.

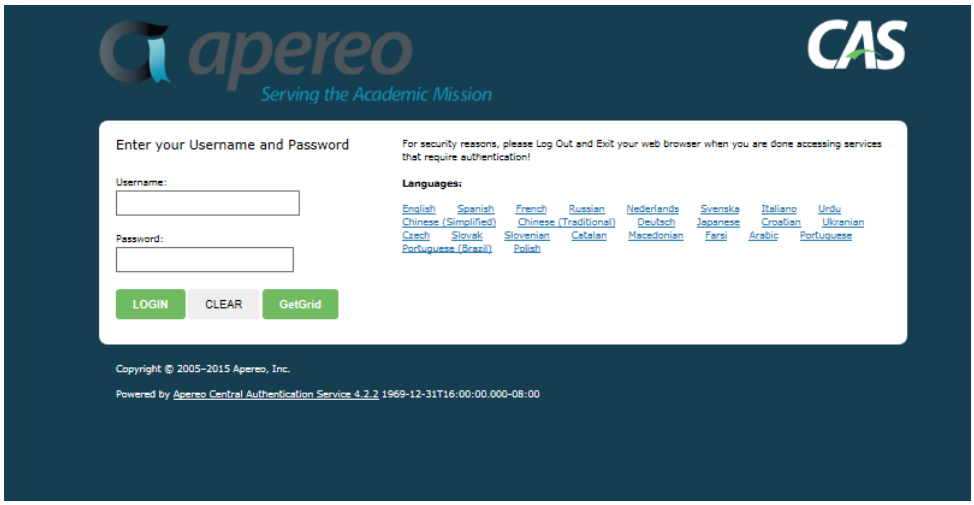
Running the Solution

For this integration, the SAS GrIDsure token is used as the enrolled token.

Accessing the CAS Management UI

1. In a web browser, open the following CAS management UI URL:

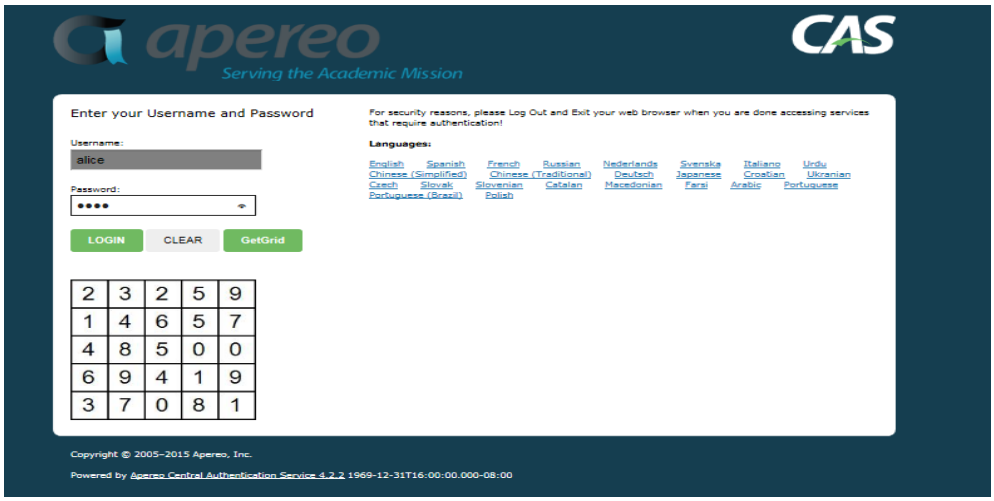
   https://myDomain.com:8443/cas-services
2. You will be redirected to the customized CAS login page. In the **Username** field, enter your user name, and then click **GetGrid**.



(The screen image above is from apereo,Inc. Trademarks are the property of their respective owners.)

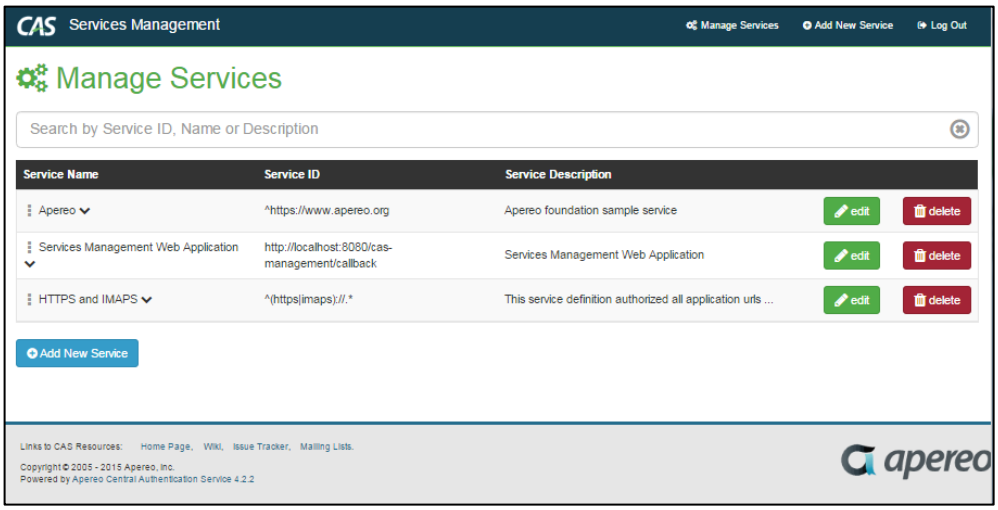
NOTE: Ensure that the user is authorized to use the CAS Management UI. Add user details in the **user-details.properties** file. For more details refer to "Appendix B: Configuring the CAS Management UI" on page 18.

3. In the **Password** field, enter your preferred grid pattern, and then click **LOGIN**.



(The screen image above is from apereo,Inc. Trademarks are the property of their respective owners.)

After successful authentication, you will logged on to the CAS management UI.



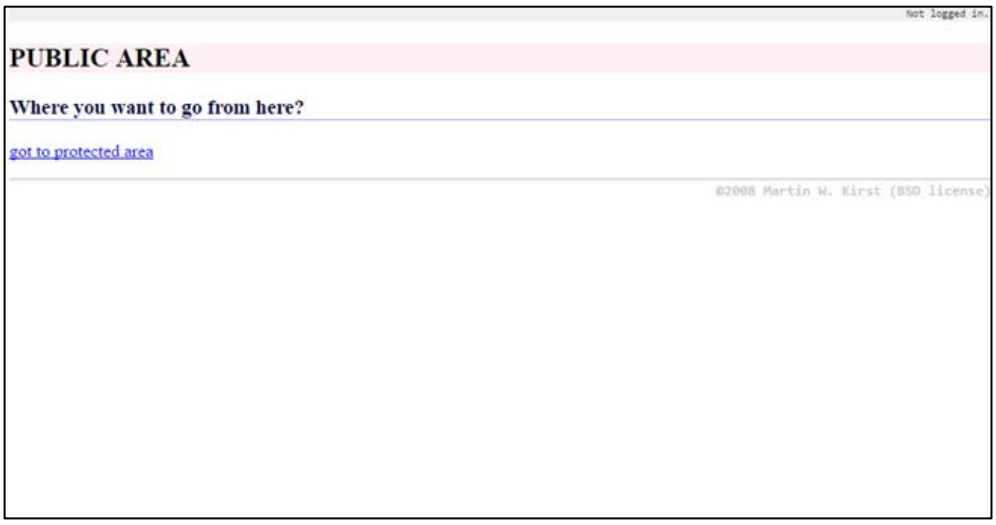
(The screen image above is from apereo,Inc. Trademarks are the property of their respective owners.)

Accessing the CAS JAVA Client

1. In a web browser, open the following CAS JAVA Client URL:

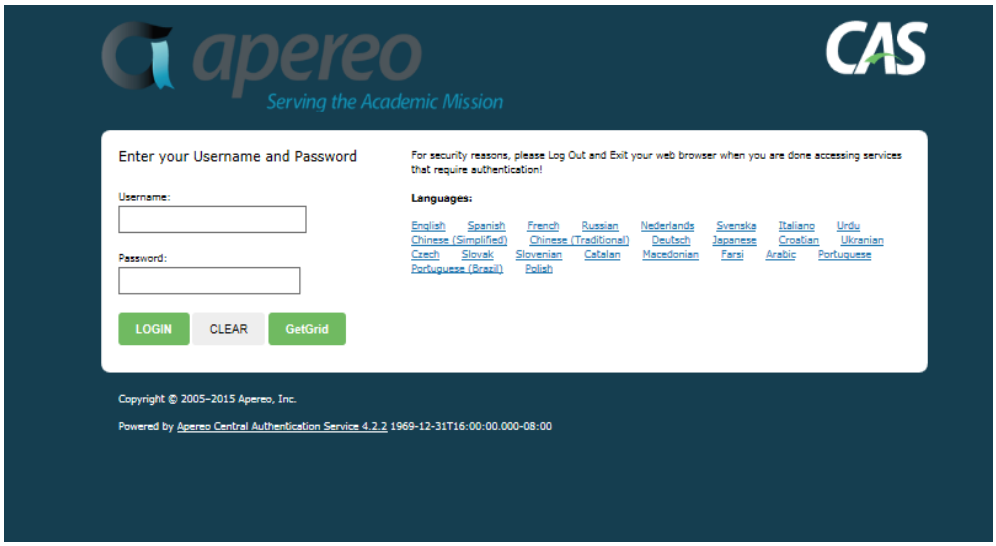
   https://myDomain.com:8443/mywebapp

2. The **PUBLIC AREA** page is displayed. Click **go to protected area**.



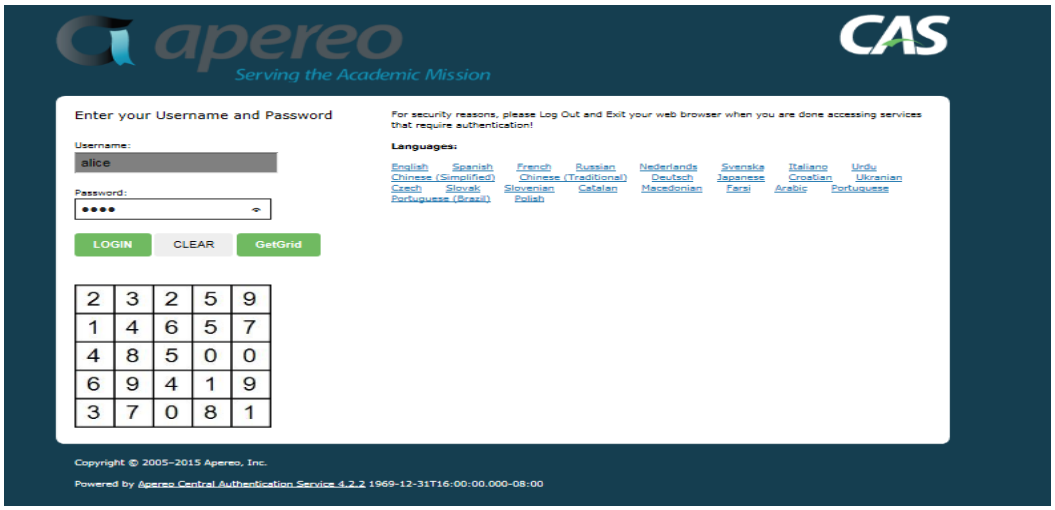
(The screen image above is from apereo,Inc. Trademarks are the property of their respective owners.)

3. You will be redirected to the customized CAS Login page. In the **Username** field, enter your user name, and then click **GetGrid**.



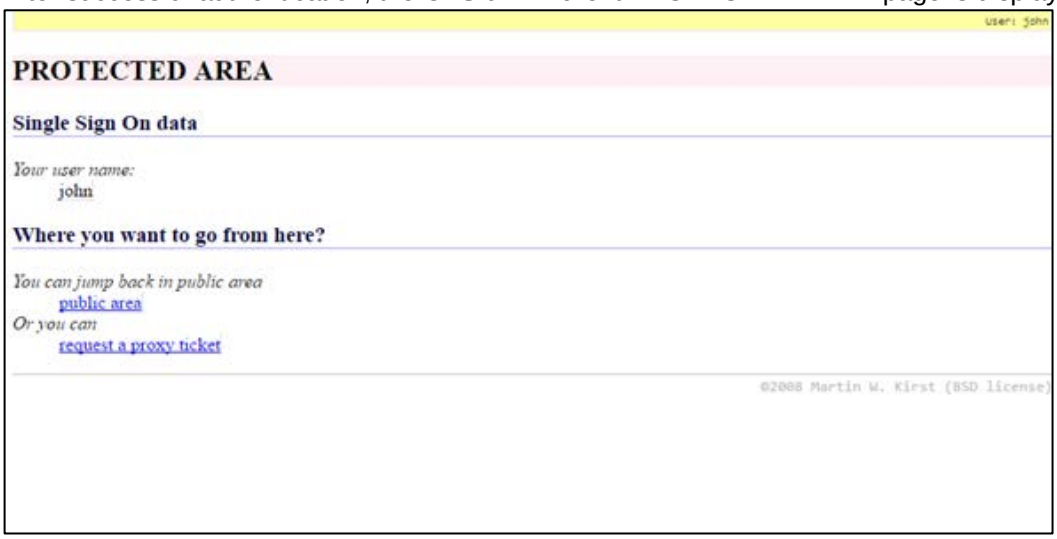
(The screen image above is from apereo,Inc. Trademarks are the property of their respective owners.)

4. In the **Password** field, enter your preferred grid pattern, and then click **LOGIN**.



(The screen image above is from apereo,Inc. Trademarks are the property of their respective owners.)

After successful authentication, the CAS JAVA client **PROTECTED AREA** page is displayed.



(The screen image above is from apereo,Inc. Trademarks are the property of their respective owners.)

Appendix A: Building and Setting up the CAS Server

Build and setup the CAS Server v4.2.2 (download the CAS Server sample war overlay project from the **https://github.com/apereo/cas-overlay-template** URL).

1. Open the command line interface, and go to the CAS server path (for example, **…\casProject\cas-overlay-template-master**).
2. Run the following command:

   mvn clean package
3. After a successful build, the **target** directory is created at the CAS server path.
4. Deploy the **cas.war** file (for example, **…\casProject\cas-overlay-template-master\target\cas.war**), created earlier in the **target** folder in the servlet container (for example, **Tomcat**).

Appendix B: Configuring the CAS Management UI

1. Download the war overlay services management web application from the **https://github.com/apereo/cas-services-management-overlay** URL.

   NOTE: Both the CAS server and the services management web application share the same configuration for the CAS services.

2. In the Services Registry JSON file (for example, **…\cas-services-management-overlay-master\etc\services\ ServicesManagementWebApplication-52497044623301.json**), locate **serviceId** in the file, and then update the **serviceId** URL.

   For example,

   "serviceId" : "https://<localhost>:8443/cas-services.*",

   Where, **<localhost>** is your system domain name or IP address of the CAS server.

3. Perform the following changes in the **cas-management.properties** file (for example, **…\cas-services-management-overlay-master\etc\cas-management.properties**):

   a. Locate line starting with **cas.host**.

   b. Set the CAS server URL:

      For example,

      cas.host= https://<localhost>::8443

      Where, **<localhost>** is your system domain name or IP address of the CAS server.

   c. Locate the following line:

      cas.prefix=${cas.host},

   d. Replace the line with the following:

      cas.prefix=${cas.host}/cas

   e. Locate the following line:

      cas-management.host=https://localhost:8443

   f. Replace the line with the following:

      cas-management.host=${cas.host}/cas

4. Copy the **services** folder (for example, **…\cas-services-management-overlay-master\etc\services**) from the cas-management application to the common path (for example, **…\casProject\etc\cas**) to share the CAS services with the CAS server.

5. In the **cas.properties** file (for example, **…\casProject\etc\cas.properties**), locate the line starting with **service.registry**, and then set the services folder path. For example,

   service.registry.config.location=file:C:/casProject/etc/cas/services

6. In the **cas-management.properties** file (for example, **…\cas-services-management-overlay-master\etc\ cas-management.properties**), locate the line starting with **service.registry**, and then set the services folder path. For example,

   service.registry.config.location=file:C:/casProject/etc/cas/services

7. In the **propertyFileConfigurer.xml file** (for example, **…\webapps\cas-services\WEB-INF\ spring-configuration\ propertyFileConfigurer.xml**), locate **casManagementProperties**, and then update the **cas-management.properties** file path. For example,

<util:properties id="casManagementProperties" location="file:c:/cas-services-management-overlay-master/etc/cas/cas-management.properties" />

8. Add users' details in the **user-details.properties** file (for example, **…\casProject\etc\user-details.properties**) to authorize users' management UI access. For example,

   alice=notused,ROLE_ADMIN,enabled

9. In the **POM.xml** file, locate **cas.version**, and then update the CAS server version. For example,

   <cas.version>4.2.2</cas.version>

10. Go to the cas-management path (for example, **…/cas-services-management-overlay-master**) and then build the CAS management application using the following command:

    mvn clean package

11. After successfully building the CAS management application, the **target** directory is created in management UI path. Copy the **cas-services.war** file **(**for example, **…\cas-services-management-overlay-master\target\cas-services.war**) created in the **target** folder into the tomcat webapps folder (for example, **…\Tomcat 7.0\webapps**)

12. Ensure that the **user-details.properties** file path is correctly defined in the **managementConfigContext.xml** file (for example, **…\Tomcat 7.0\webapps\cas-services\WEB-INF\cas-services\managementConfigContext.xml**).

Appendix C: Configuring CAS JAVA Client Test Application

1. Download the demo application from the following URL: **https://wiki.jasig.org/download/attachments/13569483/mywebapp.war?version=1&modificationDate=1212097848936&api=v2**).
2. Unzip the **mywebapp.war** file that you downloaded earlier and then save the file into a temporary folder (for example, **...\casClient\mywebapp**).
3. Download the following **.jar** files and copy them into the **mywebapp lib** folder (for example, **…\Tomcat 7.0\webapps \mywebapp\WEB-INF\lib**):
   - **cas-client-core-3.3.3.jar**

     (URL to download the file, **http://java2s.com/Code/Jar/c/Downloadcasclientcore333jar.htm**)
   - **slf4j-api-1.7.1.jar**

     (URL to download the file, **http://www.java2s.com/Code/Jar/s/Downloadslf4japi171jar.htm**)
   - **xercesImpl-2.10.0.jar**

     (URL to download the file, **http://www.java2s.com/Code/Jar/x/Downloadxercesimpl2100jar.htm**)
   - **xml-apis-1.4.01.jar**

     (URL to download the file, **http://www.java2s.com/Code/Jar/x/Downloadxmlapis1401jar.htm**)
   - **xmlsec-1.4.4.jar**

     (URL to download the file, **http://www.java2s.com/Code/Jar/x/Downloadxmlsec144jar.htm**)
4. Perform the following changes in the **web.xml** file (for example, **…\Tomcat 7.0\webapps\mywebapp\WEB-INF\web.xml**):
   a. Locate the following lines:

      <param-value>https://localhost/cas/login</param-value>

<param-value>https://localhost/cas/</param-value>

In the lines, replace **localhost** with the CAS server domain name or IP address.

b. Locate the following line (all occurrences):

<param-value>https://localhost:8443</param-value>

In the line, replace **localhost** with the CAS client domain name or IP address.

c. Locate and remove the following lines:

```
<init-param>
    <param-name>proxyCallbackUrl</param-name>
    <param-value>https://localhost:8443/mywebapp/proxyCallback</param-value>
 </init-param>
 <init-param>
                                <param-name>proxyReceptorUrl</param-name>
    <param-value>/mywebapp/proxyCallback</param-value>
 </init-param>
```

NOTE: The lines are not required for CAS v4.2 or later.

---

d. Locate and remove the following lines:

```
<init-param>
        <param-name>renew</param-name>
        <param-value>false</param-value>
                </init-param>
                <init-param>
        <param-name>gateway</param-name>
        <param-value>false</param-value>
</init-param>
```

NOTE: The lines are not required for CAS v4.2 or later.

---

5. Copy the **HTTPSandIMAPS-10000001.json** file (for example, **…/cas-overlay-template-master\target\cas\WEB-INF\classes\services**) into the services folder (for example, **…\etc\cas\services**) that is shared between the management UI and the CAS server.
6. Copy the **mywebapp.war** file and the **mywebapp** folder to the tomcat server (for example, **…\Tomcat 7.0\webapps**).
7. Restart the tomcat server and then run the mywebapp on a browser.

Support Contacts

If you encounter a problem while installing, registering, or operating this product, please make sure that you have read the documentation. If you cannot resolve the issue, contact your supplier or Gemalto Customer Support. Gemalto Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Gemalto and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Contact Method	Contact Information	
Address	Gemalto 4690 Millennium Drive Belcamp, Maryland 21017 USA	
Phone	United States	1-800-545-6608
	International	1-410-931-7520
Technical Support Customer Portal	https://supportportal.gemalto.com Existing customers with a Technical Support Customer Portal account can log in to manage incidents, get the latest software upgrades, and access the Gemalto Knowledge Base.	