Custom Web Portal Developer's Guide

SecurID™

Contact Information

RSA Link at https://community.rsa.com contains a knowledgebase that answers common questions and provides solutions to known problems, product documentation, community discussions, and case management.

Trademarks

RSA Conference Logo, RSA, and other trademarks, are trademarks of RSA Security LLC or its affiliates ("RSA"). For a list of RSA trademarks, go to https://www.rsa.com/en-us/company/rsa-trademarks. Other trademarks are trademarks of their respective owners.

License Agreement

This software and the associated documentation are proprietary and confidential to RSA Security LLC or its affiliates are furnished under license, and may be used and copied only in accordance with the terms of such license and with the inclusion of the copyright notice below. This software and the documentation, and any copies thereof, may not be provided or otherwise made available to any other person.

No title to or ownership of the software or documentation or any intellectual property rights thereto is hereby transferred. Any unauthorized use or reproduction of this software and the documentation may be subject to civil and/or criminal liability.

This software is subject to change without notice and should not be construed as a commitment by RSA.

Third-Party Licenses

This product may include software developed by parties other than RSA. The text of the license agreements applicable to third-party software in this product may be viewed on the product documentation page on RSA Link. By using this product, a user of this product agrees to be fully bound by terms of the license agreements.

Note on Encryption Technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

Distribution

Use, copying, and distribution of any RSA Security LLC or its affiliates ("RSA") software described in this publication requires an applicable software license.

RSA believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." RSA MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

March 2021

Contents

Preface

About This Guide

This guide is for web developers who want to develop custom web portal pages that enable users to authenticate to protected SaaS applications using SecurID.

For a complete list of SecurID documentation, see "SecurID Documentation" on RSA Link at https://community.rsa.com/docs/DOC-60094.

SecurID Support and Service

You can access community and support information on RSA Link at https://community.rsa.com. RSA Link contains a knowledgebase that answers common questions and provides solutions to known problems, product documentation, community discussions, and case management.

Support for SecurID Authentication Manager

Before you call Customer Support for help with the SecurID Authentication Manager appliance, have the following information available:

- Access to the SecurID Authentication Manager appliance.
- Your license serial number. To find this number, do one of the following:
  - Look at the order confirmation e-mail that you received when your ordered the product. This e-mail contains the license serial number.
  - Log on to the Security Console, and click **License Status**. Click **View Installed License**.
- The appliance software version. This information is located in the top, right corner of the Quick Setup, or you can log on to the Security Console and click **Software Version Information**.

Support for the Cloud Authentication Service and Identity Routers

If your company has deployed identity routers and uses the Cloud Authentication Service, SecurID provides you with a unique identifier called the Customer Support ID. This is required when you register with SecurID Customer Support. To see your Customer Support ID, sign in to the Cloud Administration Console and click **My Account > Company Settings**.

RSA Ready Partner Program

The RSA Ready Partner Program website at www.rsaready.com provides information about third-party hardware and software products that have been certified to work with SecurID products. The website includes Implementation Guides with step-by-step instructions and other information on how SecurID products work with third-party products.

Chapter 1: Custom Portal Development

Custom Portal Development Overview

SecurID web portal pages let users authenticate to access protected web applications using a web browser. This guide explains how to set up a custom web portal.

SecurID includes a default web portal out-of-the-box that runs on the identity router with no required configuration changes. This standard portal provides basic functions.

If you want to change the look and feel of your portal, including custom error messages to control your user experience, you can develop a custom web portal on a separate web server.

This table compares the capabilities of the standard and custom portals.

Capability	Standard Portal	Custom Portal
Application icons associated with applications	Yes	Yes
Custom branding		Yes
Custom error messages		Yes
Additional authentication based on policy criteria	Yes	Yes
LDAP password reset	Yes	Yes
Set HFED application credentials	Yes	Yes

The custom portal may stand on its own or it may be integrated into an existing web portal. The custom portal SDK (software development kit) provides sample web pages to help you get started.

Development Task Summary

This guide describes how to perform the following high-level development tasks:

1. Meet all preliminary requirements such as setting up the identity router, configuring identity sources, and testing application interactivity with the identity router using the standard portal.
2. Prepare the portal server where programmers can deploy the custom portal pages.
3. Configure the identity router to interact with the custom portal. This step involves:
    a. Configuring the custom portal as a trusted headers application. This configuration must point to the web server where the custom portal pages are deployed. See the Help topic Add an Application Using Trusted Headers.
    b. Configuring the custom portal page using the Cloud Administration Console. See the Help topic Configure a Custom Portal Page for Web Applications.
    c. Editing the login/portal/error pages to replace the placeholder "%server" in the custom portal pages with the identity router hostname.
    d. Setting the trusted headers application to use HTTPS.
4. Develop and test your custom web pages using the example web pages provided as a starting point. You

can develop the pages on a test server, and publish your changes to the production server after everything works properly.

Chapter 2: Getting Started

Getting Started

This chapter provides the following information:

- Procedures to confirm that the identity router and the standard web application portal interacts correctly with identity sources, and with protected web applications.
- Information to help you deploy a server to host the custom portal pages.

Confirm the Environment Configuration

This involves testing that the identity router and the standard web application portal interact correctly with identity sources, and with protected web applications. You must be a Super Admin in the Cloud Administration Console to perform these tasks.

The tasks are:

- Test password authentication and additional authentication.
- Test LDAP directory server password reset.

After you complete these tasks you can develop the custom portal using the trusted headers API.

Test Password Authentication and Additional Authentication

This procedure verifies whether a user can authenticate to the SecurID Application Portal using an LDAP directory server password and use additional authentication to access a protected application.

The procedure assumes the SSO Agent is fully configured and that you have the required credentials to access protected applications.

Before you begin

Confirm that:

- The identity router is connected within your network environment and you can control its configuration using the Cloud Administration Console.
- The standard web portal is accessible and provides at least one link to a protected application that is configured for single sign-on (SSO) with an access policy that requires additional authentication.
- The identity router can access the protected applications. Use ping operations to confirm connectivity between all components that must communicate. This includes the browser, identity router, identity source, web portal, and protected application.
- An identity source is configured to work with the identity router.
- Your deployment has a user account and the user has credentials needed for additional authentication.

Procedure

1. Open a supported browser.
2. Enter the standard web portal URL into the browser.
3. Sign into the web portal.
4. Click a protected application that requires additional authentication.
5. Enter your credentials when prompted.

The test succeeds if you access the protected application. If you cannot access the application, verify that everything described in Before you begin is set up properly.

Test LDAP Directory Server Password Reset

If you allow users to change their LDAP directory server passwords when using the user application portal, this procedure lets you test that capability.

Before you begin

Ensure that:

- An administrator has configured the Cloud Authentication Service to allow users to change their LDAP directory server passwords using the application portal. See the Help topic "Add an Identity Source for the Cloud Authentication Service."
- An administrator has enabled the LDAP directory server to allow users to change their passwords.

Procedure

1. Enter the standard web portal URL into a supported browser.
2. Sign into the web portal.
3. Navigate to the user account drop-down menu in the top-right corner of the portal home page and click **Change Password**.
4. Follow the prompts to change the password.

The test succeeds if you reset your password. If you cannot reset your password, verify that everything described in Before you begin is set up properly.

Trusted Headers API and Identity Router Deployment

The trusted headers API operates within these conditions:

- If unauthenticated users will be accessing the portal to access applications that do not require authentication, the web server hosting the custom portal pages with links to unprotected applications must be directly accessible to users. That is, unauthenticated users will not access these pages indirectly through the identity router.

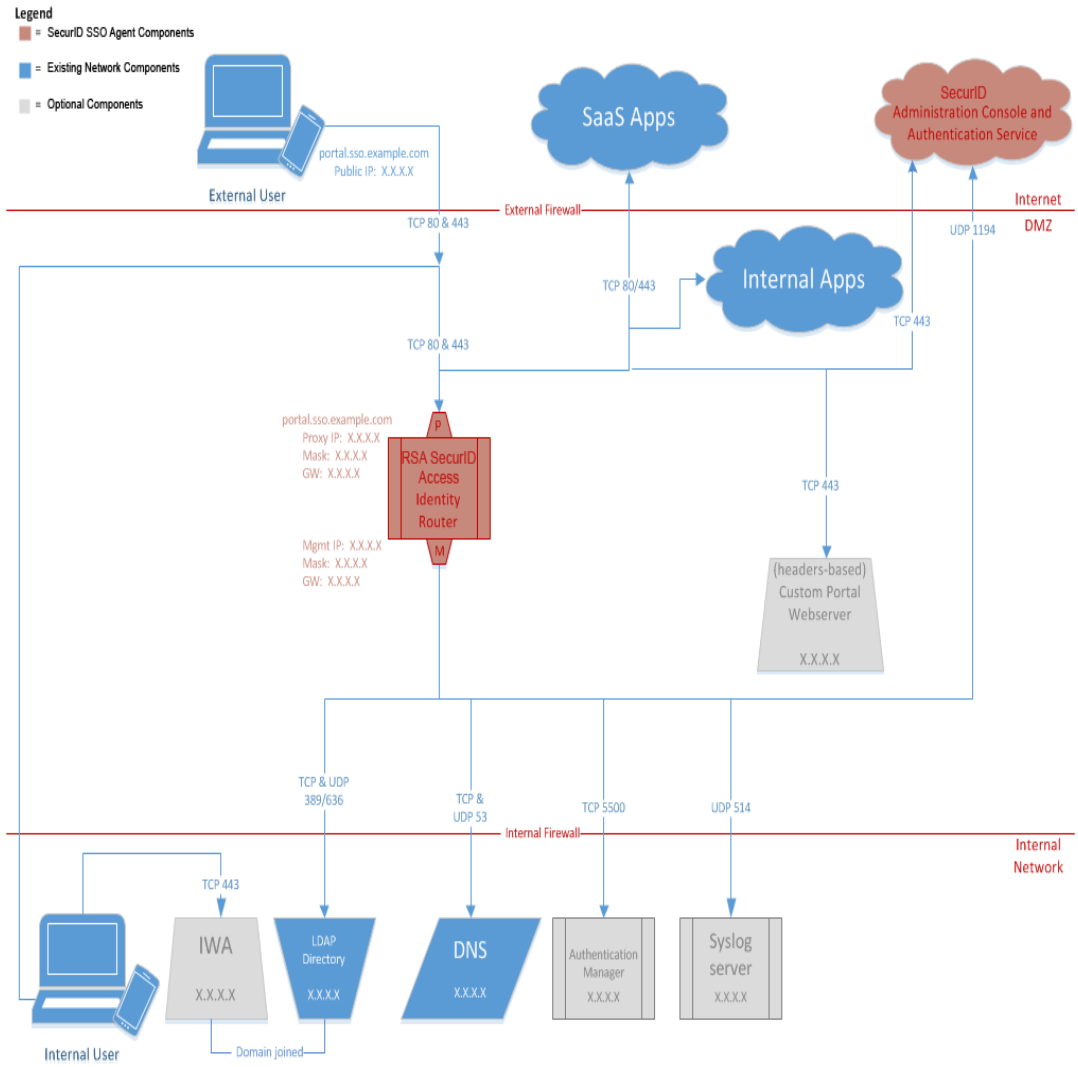
Protected portal and error pages can be hosted on the same server or a separate server but these pages present useful portal information when accessed indirectly, through the identity router.

- For on-premises identity routers, the web server hosting the custom portal pages cannot reach the identity router management interface. The identity router proxy interface must be able to access the web server.
- The web server hosting the custom portal pages can be in a different domain than the identity router protected domain name.
- With trusted headers, the custom portal web server does not need to use the singlepoint cookie for anything, and the identity router can provide access to the pages through a hostname within the protected domain name space.
- The web server hosting the custom portal pages is not required to reliably use a network time protocol (NTP) source to synchronize its time.

Custom Portal Network Configuration

The following diagram shows how the custom portal server fits in the network with the identity router. The custom portal may not be listed in the DNS but it must be accessible to the identity router.

You can develop the custom web pages directly on the custom portal server in a separate directory. To develop and test these pages on a separate web server, you must have a separate identity router. You must be able to publish the completed pages to the production portal.


Legend
= SecurID SSO Agent Components
= Existing Network Components
= Optional Components
External User
portal.sso.example.com
Public IP: X.X.X.X
SaaS Apps
SecurID
Administration Console and
Authentication Service
External Firewall
Internet
DMZ
TCP 80 & 443
UDP 1194
TCP 80/443
Internal Apps
TCP 443
TCP 80 & 443
portal.sso.example.com
Proxy IP: X.X.X.X
Mask: X.X.X.X
GW: X.X.X.X
P
RSA SecurID
Access
Identity
Router
M
Mgmt IP: X.X.X.X
Mask: X.X.X.X
GW: X.X.X.X
TCP 443
(headers-based)
Custom Portal
Webserver
X.X.X.X
TCP & UDP
389/636
TCP &
UDP 53
TCP 5500
UDP 514
Internal Firewall
Internal
Network
TCP 443
IWA
X.X.X.X
LDAP
Directory
X.X.X.X
DNS
X.X.X.X
Authentication
Manager
X.X.X.X
Syslog
server
X.X.X.X
Internal User
Domain joined

Chapter 3: Using the Trusted Headers API

Trusted Headers API and Identity Router Deployment

The trusted headers API operates within these conditions:

- If unauthenticated users will be accessing the portal to access applications that do not require authentication, the web server hosting the custom portal pages with links to unprotected applications must be directly accessible to users. That is, unauthenticated users will not access these pages indirectly through the identity router.

  Protected portal and error pages can be hosted on the same server or a separate server but these pages present useful portal information when accessed indirectly, through the identity router.

- For on-premises identity routers, the web server hosting the custom portal pages cannot reach the identity router management interface. The identity router proxy interface must be able to access the web server.

- The web server hosting the custom portal pages can be in a different domain than the identity router protected domain name.

- With trusted headers, the custom portal web server does not need to use the singlepoint cookie for anything, and the identity router can provide access to the pages through a hostname within the protected domain name space.

- The web server hosting the custom portal pages is not required to reliably use a network time protocol (NTP) source to synchronize its time.

Identity Router Reverse Proxy Role

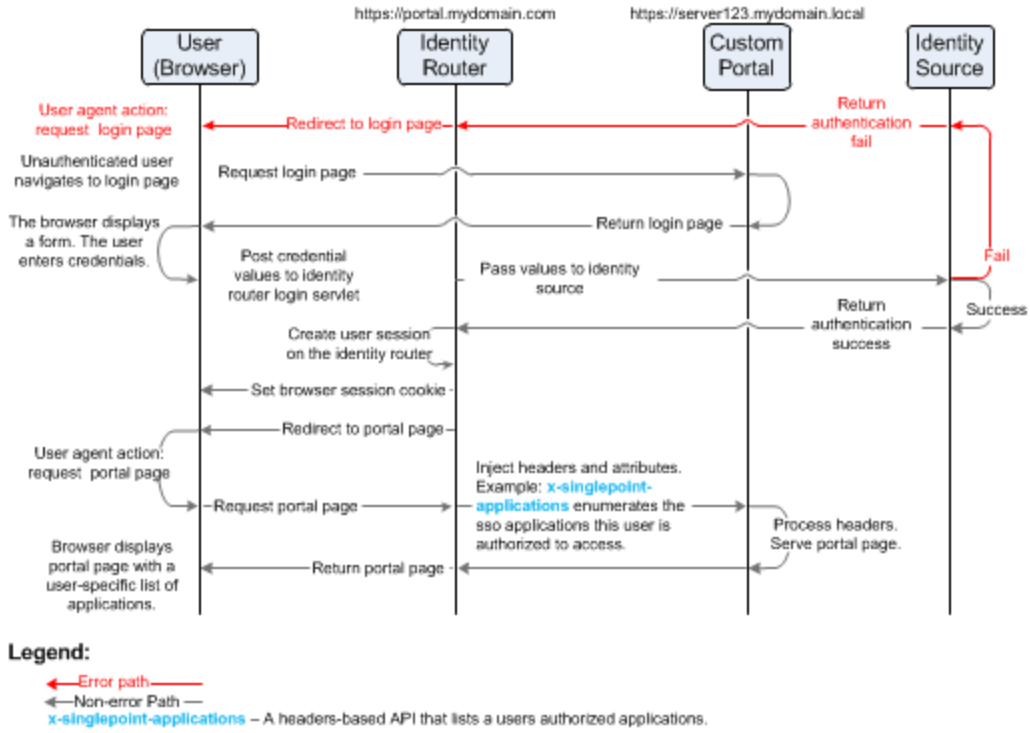
The identity router is a reverse proxy server placed between the user and other servers including the portal and protected application servers. The identity router intercepts all user requests, to rewrite and, when needed, inject headers into the requests before forwarding them to the intended server.

The identity router also intercepts all server responses, acting on them as needed to control the end-user browser experience. When a user successfully authenticates, the identity router establishes a user session and sets a cookie in the browser to manage the session.

The following graphics illustrate major interactions between the identity router, web browser, portal server, authentication server, and protected application server.

Initial Authentication Flow

The following example shows the authentication when a user signs in. Successful authentication returns a list of the applications the user is authorized to access. The browser handles user agent actions on the browser and does not require any further input or response. The terminating and resuming paths at the identity router and the URLs at the top of the diagram indicate the identity router proxy role between the portal and the browser.

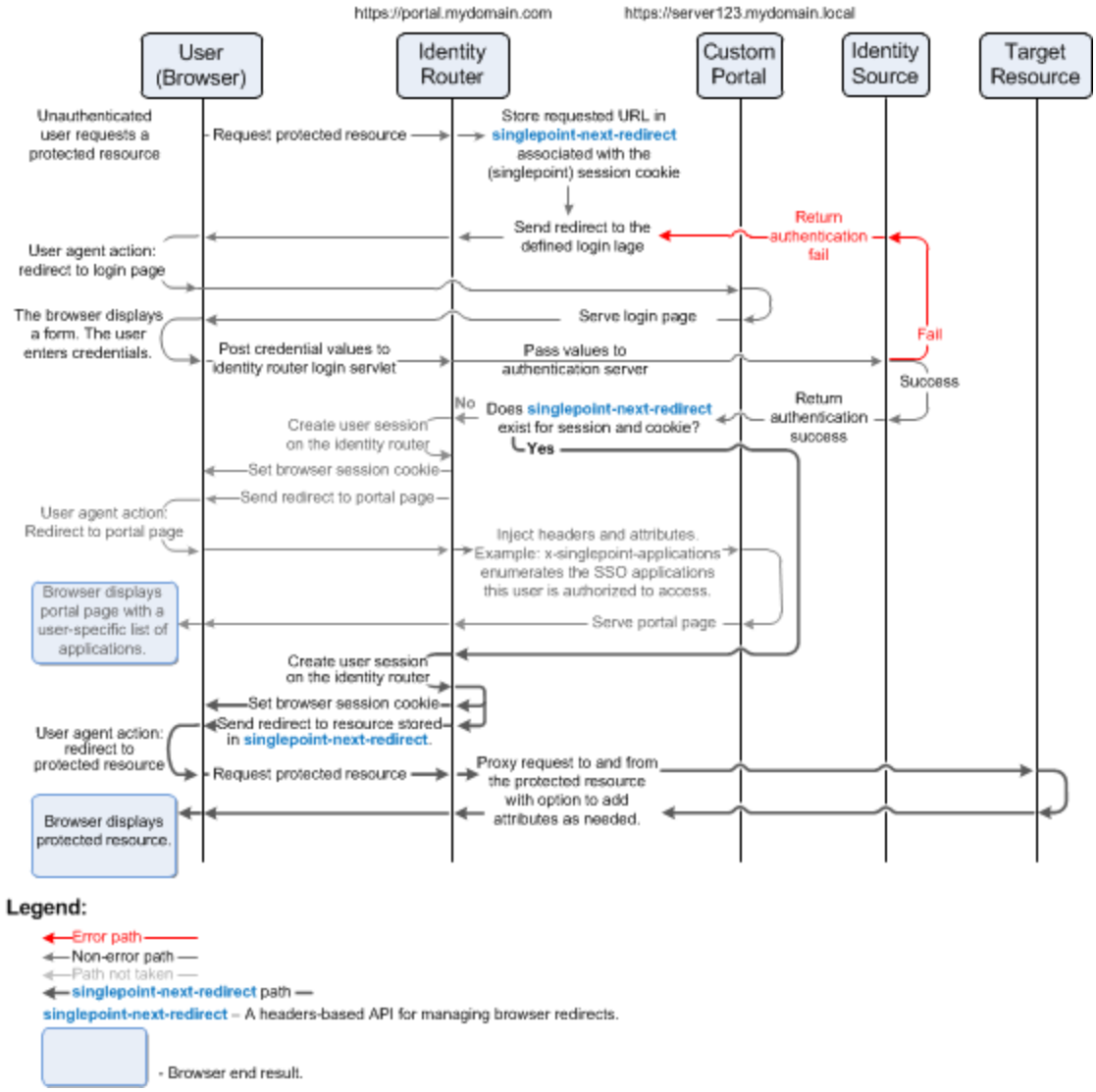


Authentication Flow Handling Possible Conditions

The following example shows a flow where a user authenticates and requests a non-proxied application (such as a SAML application). Some alternate paths handle errors and conditions where redirection depends on an authenticated state that may or may not be set.

If the user's initial request is for a non-proxied application (such as a SAML application), the final redirect to the protected resource sends the user to that application directly rather than to a proxied page.

Note: The darker flow paths in the diagram show the path when the user accesses the application using a deep link, triggering the singlepoint-next-redirect API to control the flow.



Identity Router Functionality Accessed Using Servlets

Java servlets on the identity router provide its browser-facing functionality.

- The **LoginServlet** handles sign-on requests and requests for protected resources.
- The **LogoutServlet** handles termination of the user's session and redirection to the sign-on Page.
- The **passwordMgr servlet** handles LDAP directory server password reset operations when password reset is enabled.
- The **KeychainRetrievalServlet** provides the portal with information about the applications that have been assigned to all signed-in users. For instructions to use this servlet, see Application ID Retrieval on page 35.
- The **KeychainUpdateServlet** allows the user to update HFED application credentials.

- The **IdPRetrievalServlet** provides the portal with information about the identity providers that have been made available to all users. For instructions to use this servlet, see Identity Provider ID Retrieval on page 38.

Note: Additional authentication, when dictated by policy, is handled by other SecurID functionality.

Custom Portal Web Pages Connect Users and Browsers to Identity Router Functions

The following custom portal sample web pages provided with the software developer's kit (SDK) contain the links to the servlets on the identity router.

- **login.*typ***- Login page that displays errors and posts to **LoginServlet** on submit.
- **portal.*typ***- Portal page that displays available applications, provides an option to update HFED application credentials, and includes a sign-out link to the **LogoutServlet**.
- **Strongauth.*typ***- (obsolete) Additional authentication page. This functionality is now handled by other SecurID functionality.
- **error.*typ***- Error page that displays customized authorization errors and includes a sign-out link to **LogoutServlet**.
- **keychain.*typ***- Keychain update page that manages HFED application credential updates using **KeychainUpdateServlet** and includes a sign-out link to **LogoutServlet**.
- **locationcollection.html** – Sample code page containing JavaScript code for HTML5 location collection. Use this page to understand HTML5 location collection, but do not use it as the login page. Use login.*typ* as the login page. See the code comments in login.typ for instructions on how to use location.html.

typ may be .asp, .php, or .jsp. SecurID provides starting web pages for each of these scripting languages. You must modify these web pages to access the appropriate identity router functions and install them on your custom portal.

Note: The SDK provides .asp, .php, and .jsp sample pages but you can write the custom portal in most server side scripting language that can provide a login form and POST to the **LoginServlet**, consume http headers to display a list of applications, as well as other server side functionality.
Except for the login page, all custom portal pages must be part of the custom portal trusted headers application, and must be accessed through the proxy hostname you specify when configuring the application.

Development Tasks

This section describes the main tasks for developing custom web portal pages.

Before you begin

- Confirm that the identity router and identity sources are configured, and that your application is interacting correctly with the identity router. See Getting Started on page 11.
- A portal server is deployed in the network and is ready to configure for custom portal web pages.

Procedure

1. Install an appropriate scripting language for the custom portal.
2. Configure the identity router to interact with the custom portal. This step involves:
    a. Configuring the custom portal as a trusted headers application. This configuration must point to the web server where the custom portal pages are deployed. See the Help topic Add an Application Using Trusted Headers.
    b. Configuring the custom portal page using the Cloud Administration Console. See the Help topic Configure a Custom Portal Page for Web Applications.
    c. Editing the login/portal/error pages to replace the placeholder "%server" in the custom portal pages with the identity router hostname.
    d. Setting the trusted headers application to use HTTPS.
3. Configure development tools.
4. Edit the custom portal files to point to the identity router.
5. Add custom branding and images to your web pages.
6. Customize error messages.
7. (Optional) Configure HTML5 location collection.
8. Configure user attributes needed by the protected application.
9. (Optional) Configure integration attributes for a customized user experience.
10. (Optional) Configure LDAP directory server password reset self-service.
11. Handle other errors.

Install a Scripting Language for the Custom Portal

Decide which scripting language pages to use. SecurID provides sample server pages with jsp, .php, and .asp, extensions. You can use one of these languages or another scripting language that your server platform supports.

To use the sample pages, download and install one of the following scripting language interpreters on your custom portal web server.

- JSP 2.1 or higher
- PHP 5.3 or higher
- ASP.NET and Web Tools 2013 or higher

Java Portal

The java sample web portal pages are provided in a web application archive (.war file) named **portal_jsp.war** in the SSO Web Services Software Development Kit (SWS SDK). The file is ready for deployment on a Java servlet container.

- It is built using Java 6 and tested on Tomcat 6.0.16. It should work with little or no adaptation on other containers.
- It is implemented using JavaServer Pages (JSP) without any backing beans, so that no compilation is required.

You can modify and re-assemble the JSPs and configuration. The **portal_jsp.war** file includes the following:

File	Description
error.jsp	Error page that displays authorization errors and a sign out link to **LogoutServlet**.
json-20070829.jar	Used for simple JSON parsing.
lib	Subdirectory that contains library files for the web application.
login.jsp	Sign-in page that displays errors and post to **LoginServlet** on submit.
Strongauth.jsp - (obsolete)	You can ignore this page. This functionality is now handled by other SecurID components.
WEB-INF	Subdirectory that contains: • Configuration information for the web application. • Class files for servlets. • Classes that are called by the JSP methods. • web.xml file for web application configuration.

PHP Portal

The PHP sample web portal pages are provided in the SWS SDK directory **examples/PortalAPICalls/php**. The pages comprise a collection of PHP scripts ready for deployment on a PHP-capable web server. The following server pages are included:

File	Description
error.php	Error page that displays authorization errors and includes a sign-out link to **LogoutServlet**.
keychain.php	Keychain update page that manages HFED application credential updates using **KeychainUpdateServlet** and includes a sign-out link to **LogoutServlet**.
login.php	Sign-in page that displays errors and posts to **LoginServlet** on submit.
portal.php	Portal page that displays available applications, provides an option to update HFED application credentials, and includes a sign-out link to the **LogoutServlet**.
Strongauth.php (obsolete)	You can ignore this page. This functionality is now handled by other SecurID components.

The PHP sample components have been tested using PHP 5.2.6 and Apache 2.2.9. The files should work with little or no adaptation on other servers.

ASP Portal

The ASP sample web portal pages are provided in the SWS SDK directory **examples/PortalAPICalls/asp/**. The pages comprise a collection of ASP scripts ready for deployment on a ASP-capable web server. The following server pages are included:

File	Description
error.asp	Error page that displays authorization errors and a sign-out link to **LogoutServlet**.
login.asp	Sign-in page that displays errors and posts to **LoginServlet** on submit.

File	Description
portal.asp	Portal page that displays available applications and a sign-out link to the **LogoutServlet**.

The ASP sample components have been tested using IIS 5.0 and Windows Server 2003 SP2, but should work with little or no adaptation on other versions.

Configure the Identity Router to Interact with the Custom Portal

Use the Cloud Administration Console to configure the identity router to support the custom portal web pages and the trusted headers API.

Before you begin

You must be a Super Admin to perform this task.

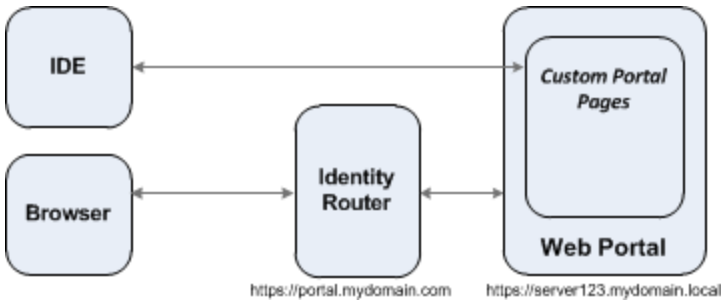
Procedure

1. Configuring the custom portal as a trusted headers application. This configuration must point to the web server where the custom portal pages are deployed. See the Help topic Add an Application Using Trusted Headers.
2. Configuring the custom portal page using the Cloud Administration Console. See the Help topic Configure a Custom Portal Page for Web Applications.
3. Editing the login/portal/error pages to replace the placeholder "%server" in the custom portal pages with identity router hostname.
4. Setting the trusted headers application to use HTTPS.

Configure Development Tools

Configure the development tools so you can edit the custom portal pages. You can use a text editor or an integrated development environment (IDE).

You also need a browser to test the customized portal web pages. The graphic below shows a typical development environment.



Edit the Custom Portal Files to Point to the Identity Router

Each web page file contains a **%server** pattern which you must change to the hostname of the identity router, or the load balancer DNS name in a high availability environment. For example. **login.php** contains the line:

```
<form method="post" action="https://%server/LoginServlet">
```

Change **%server** to the DNS name of the identity router. For example:

```
<form method="post" action="https://portal.mydomain.com/LoginServlet">
```

Add Custom Branding and Images to your Web Pages

Add customized branding and images as needed using standard HTML web page templates and HTML coding. The following steps show one way to customize your web pages:

Procedure

1. Copy a web page template to your HTML home directory.
2. Rename **login.*typ*** to **login.*typ*.orig**.
3. Rename the web page template to **login.*typ***.
4. Use your IDE to:
    a. Add images as needed to **login.*typ***.
    b. Paste in the relevant lines from **login.*typ*.orig** to **login.*typ***.
5. Perform similar operations on the portal error pages.
6. Retrieve application images from the **PortalImageServlet**. For instructions see Portal Image Service on page 35.
7. If you are using password reset functions, include password reset logic in the **portal.*typ*** page. or separate password reset page (**password_reset.*typ***).
8. Publish the modified pages to the production portal and view them in a browser to test them.

Customize Error Messages

Generic error messages help users recover from failed authentication, session expiration, and password expiration conditions. Error messages exist in the error, login, and portal web pages. Modify these error messages as needed to make them more relevant to your environment. For example, when a password needs to be updated, you may add a URL for a user to click to find more information.

Users who attempt to access the application portal and are not authorized to access any applications must be redirected a page that displays an appropriate error message. Use the https://<*IDR-HOST-NAME*>/WebPortal/error.html?singlepoint-portal-event=no_authorized_apps URL for this purpose.

Configure HTML5 Location Collection (Optional)

Your company can require different additional authentication methods for users in different countries by using the Country attribute in access policies. SecurID uses HTML5 to collect users' locations when they access the application portal. If you plan to collect users' locations when they access your custom portal, you must configure the HTML5 on the login page. Use the sample code file **locationcollection.html** to learn how to use HTML5 location collection in a browser. Make sure you pass the latitude as "latitude" and the longitude as "longitude" request variables to LoginServlet.

Note: In the sample code, location collection times out after three seconds. You can increase it as if necessary, but three seconds is the minimum recommended time.

User Attribute Headers

Administrators can designate user attributes from the identity source to be passed to the protected application

(in this case, the custom portal) on behalf of the signed-in user.

For example, the memberOf attribute can be passed to the portal. This attribute contains a list of groups to which the user belongs. The portal makes decisions about the user based on the information in the memberOf list.

User Attribute Flow

User attribute headers and the values to be passed to the custom portal are determined when the user signs in to the identity router. This information is saved for the duration of the session and is passed to each protected application that has been designated to receive the user attributes. The process flow is as follows:

1. The user logs on.
2. The designated attributes are saved to the user's session.
3. The user requests access to a protected application.
4. The request is evaluated to determine if attributes should be passed.
5. The attributes are added to the request and subsequently passed to the protected application.

Note: The user never sees the attributes on the request headers in the web browser. These attributes are only passed from the identity router to the protected application. If the attributes need to be passed to the user for some reason, this must be done by the protected application.

User Attribute Header Syntax

Attribute headers use the following syntax:

<dynamic-value> = Determined at runtime.

User attributes from identity sources are passed to the protected application in the following form:

header name = x-singlepoint-attr-*attribute-name*

header value = [*"some-data"*] or [*"some-data-1"*,*"some-data-2"*]

The name of the attribute is incorporated into the name of the request header. For example, a header name might be x-singlepoint-attr-memberOf. The value portion of the header is always presented in the following form:

[*"data"*] or [*"data1"*,*"data2"*]

This syntax permits support of single-value or multi-value user attributes. For example:

["CN=Group Policy Creator Owners,CN=Users,DC=myco,DC=org","CN=Domain

Admins,CN=Users,DC=myco,DC=org","CN=Enterprise

Admins,CN=Users,DC=myco,DC=org","CN=Schema

Admins,CN=Users,DC=myco,DC=org","CN=Administrators,CN=Builtin,DC=myco,DC=org"]

Note: Empty attributes are not passed. For example, if the user attribute phoneNumber is designated for passing using headers and a user signs in with no value assigned to phoneNumber, the header is not passed.

Integration API Header Attributes and Parameters

SecurID includes header attributes and parameters that you can enable for use in user sessions. Using attributes and parameters lets you integrate the portal more tightly with identity sources and applications to provide a better user experience.

For example you can define a header attribute to return a user's givenName field from a identity source so the portal returns "Hi Joe" instead of "Hi Smith.Joseph" on web pages.

If you specify attributes for use in user sessions, you might also need to develop code to handle the attributes when they are returned to the portal.

Keep these guidelines in mind when using header attributes and parameters:

- HTTP request parameters are used for simple strings (login username, authentication error, and redirect URL).
- HTTP headers are used for session data, formatted either as strings or JSON objects. (JSON is a generic string format for complex objects that has support in many programming languages. See www.json.orgfor detailed information and toolkits.)
- The **LoginServlet** and **LogoutServlet** can take a singlepoint-next-redirect request parameter to control where users land after logging in and out, as described in HTTP Request Parameters below.
- Parameters that start with an x- prefix can be passed through headers. Any other attribute can be passed through as a header, but these are formatted as described in User Attribute Headers on page 23.

HTTP Request Parameters

HTTP Request parameters are appended to the URL that is passed to the application or user's browser. The user's browser will display the full URL, which might include some HTTP request parameters that the identity router or custom portal pages can parse.

singlepoint-auth-error

Type	Request Parameter
Data	String
Description	Authentication or authorization error
User	Provided on redirect to Login Page or Portal Page to indicate that an error occurred during login or while accessing a specific application.

See Handling singlepoint-auth-error Strings on page 31 for descriptions of error strings.

singlepoint-next-redirect

Type	Request Parameter
Data	String
Description	URL where a user is sent after successful login.
User	(Optional) Used by **LoginServlet** and **LogoutServlet**. The identity router allows redirects only to hostnames that are associated with your Protected Domain Name, configured applications, configured identity providers, or custom portal URLs (Login Page, Portal Page, and Error Handling Page) specified in the Cloud Administration Console.

Type	Request Parameter
	If the parameter is not provided, or if the provided parameter specifies a URL that is not allowed, the **LoginServlet** redirects to the configured portal page, and the **LogoutServlet** redirects to the configured sign-in page. To view a list of the allowed redirects for your deployment: 1. Set the identity router logging level to debug. 2. Publish changes. 3. Either view the system log messages using the Cloud Administration Console, or generate a log bundle on the identity router and view the **symplified.log** file.

singlepoint-password

Type	Request Parameter
Data	String
Description	User's LDAP password.
User	Expected by **LoginServlet**. **Note:** This sensitive parameter is always POSTed to the **LoginServlet**. It is never sent as a URL parameter.

singlepoint-username

Type	Request Parameter
Data	String.
Description	User's LDAP username.
User	Expected by **LoginServlet**.

HTTP Header Attributes

HTTP header attributes are additional headers passed to or received from the identity source, or application, or user’s browser.

The dynamic request header described below lets you define custom header attributes to handle specific data elements in your environment.

x-singlepoint-applications

Type	Request Header
Data	JSON.
Description	List of applications the user is allowed to access, each containing a text and URL property.
User	Provided on forward to all portal pages and/or selected applications to indicate which applications the user is allowed to access. This header is injected when you configure a Trusted Headers application and choose **pass headers** For instructions, see the Help topic "Add an Application Using Trusted Headers."

x-singlepoint-application-name

Type	Request Header
Data	String.
Description	Name of the associated application.

Type	Request Header
User	Provided on forward to all portal pages and/or selected applications to indicate which applications the user is allowed to access.

x-singlepoint-attr-<attribute-name>

Type	Dynamic Request Header
Data	Custom: [“*some-data*”] or [“*some-data-1*”,”*some-data-2*”].
Description	User attributes passed from the identity source to the application.
User	Provided on forward to the application to indicate which parts of the application the user can access.

x-singlepoint-failed-keychain-application

Type	Request Header
Data	JSON.
Description	A single JSON string representing the application for which user authentication failed.
User	Provided on forward to the error page.

x-singlepoint-username

Type	Request Header
Data	String
Description	User's LDAP username
User	Provided on forward to all portal pages and/or selected applications to indicate which user is authenticated.

LDAP Directory Server Password Reset

When this feature is configured, users can change their own LDAP directory server passwords from the portal. The portal can also interact with the LDAP directory server to allow users to reset their LDAP directory server passwords from the portal.

Verify the following settings:

- The LDAP directory server's certificate service is installed, running, and configured to accept SSL connections.
- The identity router connects to the LDAP directory server using port 636.
- You added an LDAP directory server identity source in the Cloud Administration Console with the following options:
    - Select the **Allow Users to Change Passwords** checkbox. This adds a **Change Your Login Password** link to the **user account drop-down menu** of the portal (displayed by clicking the **Settings** icon). A user can click this link to change the LDAP directory server password at any time.
    - Select the **Use SSL** checkbox.

- The administrator account specified by the username and password fields must have adequate privileges to modify other user's passwords. For Active Directory, the administrator must have permissions equal or greater than those given to the Domain Users group. For LDAP, the administrator must have root privileges on the directory server.

A user who tries to sign in to the portal with an expired LDAP directory server password is prompted to create a new password, which the identity router passes to the LDAP directory server.

See the Help topic "Add an Identity Source for the Cloud Authentication Service" for instructions to set up an LDAP directory server identity source for the Cloud Authentication Service.

Add Password Reset Logic to the Portal Server Page

If you want users to access password reset functions, you must add password reset logic to a custom portal page. In most cases the **portal.*typ*** page is a convenient place to include this logic rather than creating a separate **password_reset.*typ*** page. The logic you add interacts with the **passwordMGR** servlet (**https://<*your identity router*>/passwordMgr/resetpw**) to handle user password reset operations.

Procedure

1. Open the main portal page (called portal.typ in this example) in the IDE.
2. Keep or remove any branding or customizations as appropriate from the source page.
3. Add password reset functions using LDAP directory server password reset parameters described in LDAP Directory Server Password Reset Parameters below.
4. Develop the logic to handle JSON responses and do form posts to specific **passwordMGR** URLs to carry out password reset operations.

LDAP Directory Server Password Reset Parameters

When the user tries to reset the password by entering the old password and the new password, the identity router validates the old password and passes the new password to the LDAP directory server.

The API uses four parameters to allow LDAP directory server password reset:

```
https://<your identity router>/passwordMgr/resetpw
```

The form post uses the following data field:

user_name

user_store_name

old_password

new_password

The resetpw API can be used in the following situations:

- User password expired in the identity source.
- User is required to change the password at the next logon.
- User-initiated password reset request.

The API requires the old password. You can obtain the old password from the session or the user can be asked to enter the old password.

Note: The resetpw API enforces the LDAP Password policy and meets the security requirement of providing the old password while resetting the password.

Using the Parameters for LDAP Directory Server Password Resets

Your custom portal page must handle the following situations to successfully handle password resets:

- Verify that the password change feature is enabled for the LDAP directory server identity source connected to the Cloud Authentication Service.
- Respond when the user password has expired in the LDAP Directory Server or the user has been set to change the password on next logon.
- Respond to a user-initiated password reset request (change password).
- Handle errors.

Verifying That LDAP Directory Server Password Change Is Enabled

To verify that **Allow Users to Change Passwords** is set for the LDAP directory server identity source, do a get operation against the following URL:

```
https://<your identity router>/passwordMgr/allow
```

The JSON response resembles the following:

```
{"isPasswordChangeSupported":true,"userStoreName":"testAD","userName":"tes
ter","errorCode":"SUCCESS"}
```

Note: The above call requires a valid identity router session, that is, that the user is already logged in to the identity router.

Verifying That LDAP Directory Server Password Reset Is Enabled

To verify whether the **user password has expired** in the LDAP directory server or the user has been set to **change his password on next logon** in the LDAP directory server identity source, do a get operation against the following URL:

```
https://<your identity router>/WebPortal/api/v1/session/
```

The JSON response resembles the following:

```
{"Session":{"UserName":"tester","LastActivityDate":"2020-07-
02T11:50:47+0000","PasswordResetRequired":true}}
```

PasswordResetRequired is set to true when the user password has expired or user must change password on next logon is enabled.

If **PasswordResetRequired** is true, then use the /passwordMgr/resetpw API to reset the password. This API relies on the old password and validates it.

If **PasswordResetRequired** is false, then use the /passwordMgr/changepw API to change the password. This API does not require the old password.

Note: The above call requires a valid identity router session, that is, that the user is already logged in to the identity router. Even when the user password is expired or the user must change the password on next logon, a valid user session is created with the old password.

Add Password Change Logic to the Portal Server Page

If you want users to access password change functions, you must add password change logic to a custom portal page. In most cases the **portal.typ** page is a convenient place to include this logic rather than creating a separate **password_reset.typ** page. The logic you add interacts with the **passwordMGR** servlet (**https://<*your identity router*>/passwordMgr/changepw**) to handle user password change operations.

Procedure

1. Open the main portal page (called portal.typ in this example) in the IDE.
2. Keep or remove any branding or customizations as appropriate from the source page.
3. Add password change functions using LDAP directory server password change parameters described in Changing the Password Based on User Request below.
4. Develop the logic to handle JSON responses and do form posts to specific **passwordMgr** URLs to carry out password change operations.

Changing the Password Based on User Request

Users must be able to change their LDAP directory server passwords from the **user account drop-down** menu in the top-right corner of the portal home page. The following requirements must be met:

- The user has successfully signed in to the portal.
- The identity source is enabled for LDAP directory server password changes. To verify this, see Verifying That LDAP Directory Server Password Change Is Enabled on the previous page.)

To change the current user's LDAP directory server password, do a form post against the following servlet URL:

```
https://<your IDR>/passwordMgr/changepw
```

The form post uses the following data field:

```
new_password
```

```
old_password
```

Note: The changepw API does not enforce the LDAP Password policy while resetting the password. Use the resetpw API to enforce the LDAP Password policy.

Handling LDAP Directory Server Password Reset Errors

See LDAP Directory Server Password Reset Error Codes on the next page for a list of singlepoint-auth-error strings used for LDAP directory server password resets.

Set Application Credentials

When this feature is enabled, users can change their application-specific HFED credential information from the portal.

After the user clicks **Set Credentials** a form containing fields for the required application credentials appears. After the user submits the form, **KeychainUpdateServlet** is invoked.

Note: To enable this feature for each application, you must select **Allow Users to Change Credentials** on the Portal Display tab when configuring the application from the **Applications > My Applications** page of the Cloud Administration Console.

Update Keychain Credentials

KeychainUpdateServlet updates application credentials set by a signed-in user.

The user enters the required application credentials in a form and submits the form to be handled by **KeychainUpdateServlet**.

Users can update their credentials under the following circumstances:

- The user accesses an application for which the stored credentials are invalid. This triggers the KEYCHAIN_AUTHENTICATION_ERROR and prompts the user for new credentials.
- The user chooses to set or update credentials for an application.

KeychainUpdateServlet validates the request, checks whether allowKeychainUse is enabled for the application, and sets useKeychain to true if available as part of the form.

Note: The servlet works only if allowKeychainUse is enabled for the application and useKeychain is available as part of the form.

After updating the application credentials, **KeychainUpdateServlet** redirects the user to the configured portal page by default. You can override this behavior by appending the redirectUrl request parameter to the form submit.

If the user clicks **Cancel**, the previous page appears.

Handling singlepoint-auth-error Strings

This section lists subsets of singlepoint-auth-error strings that the identity router may generate:

- LDAP directory server password reset errors
- Authentication errors
- Authorization errors
- Other codes are not common and should be treated as generic errors.

LDAP Directory Server Password Reset Error Codes

The identity router generates error codes described in the table below in response to password reset errors received from the identity source. These default errors are returned to **error.php** for handling.You can write these errors as needed on portal pages for users. Notes in descriptions clarify the error message.

Error Code	Description
INTERNAL_ SERVICE_ ERROR	Internal server error. Contact your administrator.
INVALIDINPUT	Make sure that the new password meets your organization requirements. **Note:** You entered an invalid password.
INVALIDINPUT ORPASS	Make sure that the new password meets your organization requirements and that the old password is correct. **Note:** One or both passwords (new and old passwords) entered are invalid.
NO_ SUFFICIENT_ RIGHT	The administrative user configured for the identity source does not have sufficient privileges.

Error Code	Description
TIMEOUT	Timed out while trying to connect to LDAP directory server. Check your connectivity or contact your administrator. **Note:** The change request timed out.
USER_STORE_ ERROR	Error or errors occurred in the identity source. Check your connectivity or contact your administrator. **Note:** Error connecting to the identity source.
USER_STORE_ SERVICE_ NOT_ READY	A problem may exist in the LDAP directory server. Contact your administrator.
USER_NAME_ NOT_FOUND	The user name does not exist in the identity source. Check your connectivity or contact your administrator. **Note:** User name not found due to a probable connectivity error.
UNAUTHORIZED	The user is not authorized. **Note:** The admin user account does not have write permissions to the LDAP directory server.

Authentication Error Codes

The identity router generates error codes described in the table below in response to authentication errors received from the identity source. These errors are returned to **error.php**. You can write these errors as needed on portal pages for users. Notes in descriptions clarify the error message.

Note: Strong authentication errors are handled by internal identity router pages.

Error Code	Description
CONCURRENT_ SESSION_ LIMIT	The concurrent session limit has been reached; the user must log out of other sessions and try again.
IDENTITY_ NOT_UNIQUE	The user identity is associated with multiple users. **Note:** This is a legacy error message.
INTERNAL_ SERVER_ ERROR	The server encountered an unexpected condition that prevented it from fulfilling the request. **Note:** This is an identity router error condition.
NOT_ AUTHENTICATED	Unable to authenticate the user with the provided credentials.
UNAUTHORIZED	This user is not authorized to access the application.

Authorization Error Codes

The identity router generates error codes described in the table below in response to authorization errors returned to **error.jsp**. You can write these errors as needed on portal pages for users. Notes in descriptions clarify the error message.

Error Code	Description
ALLOW	A requested resource is not associated with a protected application.

Error Code	Description
ALLOW	The user is authorized for the requested resource. **Note:** The user is authorized to access the requested protected application.
APP_ CONNECTION_ ERROR	The identity router is unable to establish a connection with the application server. This is typically due to a mismatch between the cipher suites or protocols used by the identity router and those supported by the application server.
APPLICATION_ AUTHENTICATION_ ERROR	Authentication of the underlying application failed, typically because a username and/or password is out of sync with the authentication identity source.
COOKIE_EXPIRED	The cookie has expired due to a session expiration or inactivity timeout.
DENY	The user is not authorized for any applications. **Note:** The user is not authorized to access any applications.
DENY	The user is not authorized to access the requested resource.
INVALID_COOKIE	The user's **access** policy cookie is invalid because it originated from a different IP address than the address from which it was created.
INVALID_COOKIE	The user's **access** policy cookie is invalid.
KEYCHAIN_ AUTHENTICATION_ ERROR	Authentication of the protected application failed. This is typically due to typically due to the user credentials stored in the keychain being out of sync with the application's user database.
NO_COOKIE	The user must be authenticated. Note: The user tried to access an application without authenticating. For example, the user reused a URL from a previous session.

Appendix A: Programmatic Access to Application Icon

Portal Image Service

The identity router is responsible for managing the user portal, delivery and presentation of application connector metadata. The identity router portal is comprised of data from several distinct services, including keychain, images and identity providers among others. This technical note describes how to use the portal image service to retrieve a 50x50 pixel PNG image that represents an application to which a user has access. The service is exposed as a servlet with the name **PortalImageServlet**. For example:

```
https://home.symplified.net/PortalImageServlet?arg1=123
```

The **PortalImageServlet** takes one of two arguments; the first 'application_uuid' is used to retrieve an image for the requested application identifier. The second, 'idp_id', is used to retrieve an image for an identity provider that is allowed for the given portal. The arguments may not be combined.

Options

Application Identifier Parameter – application_uuid

When presented with the URL query parameter 'application_uuid' the portalImageServlet will attempt to locate the image resource associated with that application. The application identifier is a Universally Unique Identifier (UUID) also sometimes referred to as a Globally Unique Identifier (GUID). This is a 128-bit identifier presented as a UTF-8 string in the form of "00000000-0000-0000-0000-000000000000".

Example

```
https://portal-url/PortalImageServlet?application_uuid=00000000-0000-0000-0000-000000000000
```

Identity Provider (IdP) Identifier Parameter – idp_id

When presented with the URL query parameter idp_id the PortalImageServlet will attempt to locate the image resource associated with that IdP. The IdP identifier is a UTF-8 encoded string representing the friendly name of the IdP. For example, to access the image for the Integrated Windows Authentication (IWA) IdP, you would use the following: https://portal-url/PortalImageServlet?idp_id=IWA

Response

The PortalImageServlet will return an image in PNG format with the HTTP "Content-Type" header set to "image/png". In case of error, an appropriate HTTP status code along with information about the nature of the error condition. The response image is suitable for direct usage in an HTML img element of the form:

```
<img src="https://portal-url/PortalImageServlet?application_uuid=<uuid>"/>
```

Application ID Retrieval

The **KeychainRetrievalServlet** is responsible for providing the portal with information about the applications that have been assigned to the logged-in user. The servlet will return a list of applications in JSON format. The icon image URL for a given application may be found in the keychainApps array under the iconSrc field. This is a relative path by which the application image may be retrieved.

Example: JSON Response


```
{
"username": "user",
"keychainApps": [{
 "iconSrc": "/PortalImageServlet?application_uuid=<uuid>",
"id": "bd1a7ccc-fbcd-46b6-b63b-7470bf8843ef",
"useKeychain": true,
"username": "user@symplified.com",
"credentials": [{
"id": "username",
"name": "emailAddress",
"displayName": "Username",
"value": "user@symplified.com",
"obfuscate": false
}, {
"id": "password",
"name": "password",
"displayName": "Password",
"value": "******",
"obfuscate": true
}
],
"requiresAuthentication": true,
"allowKeychainUse": true,
"webOpenInNewTab": true,
"allowSameTab": false,
"mobileOpenInNewTab": true,
"keychainEditDisabled": false,
"isHttpFedDirect": false,
"url": "https://deem-expense-deem-com.symplified.net/expense/?
```

```
status\u003dOK\u0026singlepoint-force-sso\u003dtrue",
"name": "Deem Expense",
S"text": "Deem Expense"
}]
}
```


Example: HTTP Header Data

In addition to direct retrieval via the **KeychainRetrievalServlet**, custom portals will receive the keychain data from a custom HTTP header named HTTP_X_SINGLEPOINT_APPLICATIONS. The data returned in this header is the same JSON format as that returned directly from the servlet.


```
[
{
"Allowkeychainuse": true,
"allowSameTab": true,
"credentials": [
{
"displayName": "Username",
"id": "username",
"name": "login",
"obfuscate": false,
"value": "ahatest2_aha"
},
{
"displayName": "Password",
"id": "password",
"name": "password",
"obfuscate": true,
"value": "******"
}
],
"id": "1ee2edc5-07c6-44db-8f8a-0d53449c7d4d",
"isHttpFedDirect": false,
```

```
"keychainEditDisabled": false,
"mobileOpenInNewTab": true,
"name": "AMS-NetForum",
"requiresAuthentication": true,
"text": "AMS-HFed-Proxy",
"url":https://ams-aha-org.singlepoint48.com/
iweb?singlepoint-forcesso=true",
"useKeychain": true,
"username": "ahatest2_aha",
"webOpenInNewTab": true
}
]
```


Identity Provider ID Retrieval

The IdPRetrievalServlet is responsible for providing the portal with information about the identity providers that have been made available to all users. The servlet - which is available even when logged out - will return a list of providers in JSON format. The icon image URL for a given provider may be found in the returned item array under the iconSrc field. This is a relative path by which the provider image may be retrieved.

Example: JSON Response


```
[
{
"iconSrc": "/PortalImageServlet?idp_id=IWA",
"loginUrl": "https://login_url/",
"name": "IWA",
"portalIdPType": "IDP"
}
]
```


Frequently Asked Questions

Q: Is this a stable API that I may incorporate into my custom portal?

Yes, the **PortalImageServlet** is available in all production environments and is used in the default identity router portal for the same purpose.

Q: Is there an alternate method to obtain either this image or the (non-scaled) original image uploaded?

No, at this time there is no means to retrieve the original image used during application creation. The identity router managed images on the portals' behalf.