RSD GLASS 3.4.4

Multi-Tenant Edition
Governance Apps

Installation Guide

English

RSD-000046-EN-d17a289

Trademarks and Registered Names

All brand and product names quoted in this publication are trademarks or registered trademarks of their respective holders.

Notices

RSD GLASS® is a software package property of RSD - Geneva, Switzerland that cannot be used without license.
RSD reserves the right to make any modifications to this product and to the corresponding documentation without prior notice or advice.

Manual: RSD GLASS® - Installation Guide version 3.4.4
RSD-000046-EN-d17a289

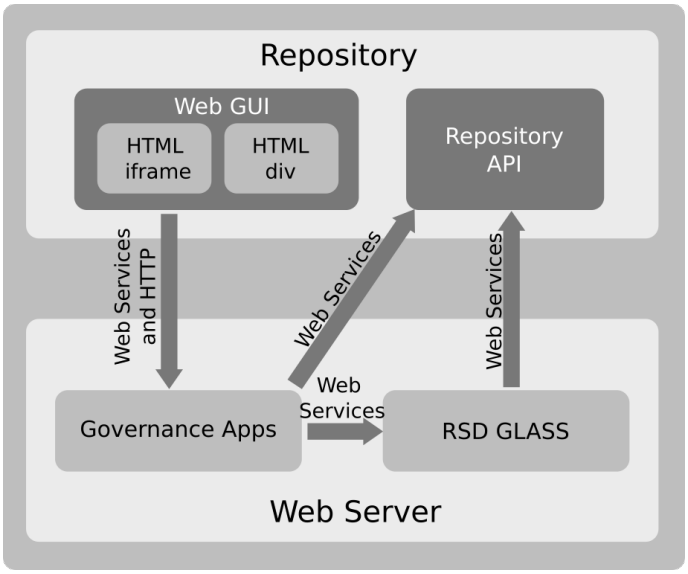
Contents

1. Introduction

RSD GLASS® Governance Apps is a set of embeddable web components that allow end users to access RSD GLASS® features directly from the graphical client of their repository; for example, the user can govern content by clicking a button in the repositories client.

The web components interact with the RSD GLASS® API on the one side and with a content repository API on the other from the repository client. Note that the connection is established by the respective RSD GLASS® connector, which connects RSD GLASS® and a third party repository.

Figure 1: Architecture overview



Governance Apps include the following web components:

Wizard

Vaadin-based web component that can be embedded or called from your repository front-end application.

It serves to catalog and govern your repository documents. Note that it uses the document ID provided by the repository connector.

Widgets

Widgets are HTML5 standalone components that can be integrated in any web application as an iframe or directly (refer to *Widgets* on page 19).

1.1. Requirements

The following web browsers are supported:

- Opera 28 and later
- FireFox 35 and later
- Chrome 40 and later
- Internet Explorer 10.x and 11.x

2. Setup

To set up Governance Apps, you need to do the following:

1. Create a configuration file with the application properties.
2. Deploy the WAR archive with Governance Apps to a web application container.
3. Integrate the required Widgets and Wizard into your web application.

2.1. Configuration and Installation

Governance Apps configuration is set in the `RSDGLASSGovernanceApps.war/WEB-INF/conf/govapps.properties` file. This file is intended as a template for your configuration. Do not modify the file: copy it and modify the copy to allow easy recovery of the default settings and product upgrade (refer to *Creating Configuration File* on page 5).

The `govapps.properties` file content differs depending on the setup:

- In a cloud RSD GLASS® instance, the file contains a ZooKeeper configuration properties. Any other properties are obtained from the ZooKeeper.
- In an on-premise setup, the file contains any applicable properties including properties needed to establish communication with the target repositories.

2.1.1. Creating Configuration File

To create the Governance Apps configuration file, do the following:

1. Copy the `RSDGLASSGovernanceApps.war/WEB-INF/classes/govapps.properties` to a location accessible to your web server.
2. Set the `govapps.config.folder` property to the file location as your server's `govapps.config.foldersystem` property, for example, –`Dgovapps.config.folder=/opt/tomcat/conf/govapps`).

2.1.2. Configuring On-Premise Instance

Governance Apps use a `govapps.properties` file to acquire its configuration. Multiple Governance Apps can use the same externalized `govapps.properties` file which allows you to change settings for all Governance Apps instances in a single configuration file.

Generally, you need to edit the following in the `govapps.properties` file for an on-premise setup:

1. Define the ID of the default content repository in the `repositories.default.crID` property.

   The ID is used by Governance Apps and must match the Unique code ID value from the Content Repositories configuration in RSD GLASS® Governance Manager.
2. Define the default virtual repository ID in the `repositories.default.virtualCrId` property.

   The ID is used by the Governance Apps and must match the Unique code ID value from the Virtual Content Repositories configuration of RSD GLASS® Governance Manager.
3. For each repository, define the connection properties so Governance Apps can access the repositories to acquire additional metadata.

   For fileNShare, the following properties are required:

   - `repositories.fns.userName`: user used to access the repository
   - `repositories.fns.password`: user password
   - `repositories.fns.domainUrl`: URL of the fileNshare REST service

- `repositories.fns.crId`: target content repository
- `repositories.fns.virtualCrId`: target virtual content repository

Figure 2: Example fileNshare configuration with fileNshare as the default repository

```
repositories.fns.userName=acme-crystal@acme.com
repositories.fns.password=acme4ever
repositories.fns.domainUrl=https://box.acme.com/fns-service
repositories.fns.crId=fns
repositories.fns.virtualCrId=fns_dev
```

4. For each repository, define the connection properties so Governance Apps can access the repositories to acquire additional metadata.
5. Configure the Governance Apps URL on the repository side for the respective tenant.

   The URL provides the Wizard the information necessary to identify the document to work with and its properties. For fileNshare repositories, the URL must be `https://<SERVER_NAME>:<PORT>/RSDGLASSGovernanceApps/?action=[declare/govern/status]&documentId=[documentID]`. Note that the URL-related properties in the `govapps.properties` for your repository need to be configured. For further information, refer to the repository documentation.

Now you can deploy the application, set up authentication, apply any white-labelling features, and provide localization resources.

2.1.3. Configuring Cloud Instance

To configure Governance Apps instance in a cloud environment, define the following ZooKeeper properties in the govapps.properties file:

- configuration.mode
- configuration.farm
- config.store.zk.host
- config.store.zk.retryPolicy.wait.ms
- config.store.zk.retryPolicy.count
- config.store.zk.connectionTimeout.ms
- config.store.zk.sessionTimeout.ms
- config.store.zk.app
- config.store.zk.farms

- config.store.zk.tenants

Figure 3: Example cloud instance configuration

```
#################
# Configuration #
#################
# Configuration mode cloud or local (=empty)
configuration.mode=
configuration.farm=
# ZK Server conf
config.store.zk.host=demo-glass4.rsd.com:2181
config.store.zk.retryPolicy.wait.ms=1000
config.store.zk.retryPolicy.count=3
config.store.zk.connectionTimeout.ms=10000
config.store.zk.sessionTimeout.ms=60000
# ZK Tree conf
config.store.zk.app=/rsd-cloud/DEV/rsd/app
config.store.zk.farms=/rsd-cloud/DEV/rsd/farms
config.store.zk.tenants=/rsd-cloud/DEV/rsd/tenants
```

2.1.4. Setting up SSL for Governance Manager

For RSD GLASS® Governance Manager connection through SSL, import the public certificate to the Governance Apps' JVM.

To import the certificate, run the following `keytool` command:

```
keytool -import -alias Glass -file Glass.cer -keystore <JAVA_CA_CERT_STORE>
```

2.2. Deployment

To deploy Governance Apps, extract the `RSDGLASSGovernanceApps.war` from the product ZIP archive and deploy it to your server (for example, for Tomcat, copy the war file to `$CATALINA_HOME/webapps/` folder).

Note that once Governance Apps has been deployed, a security directory is created that stores the Spring Security context xml files. Hence any configuration is performed in govapps.properties or ZooKeeper.

> **Note:**
>
> It is recommended to configure the WAR file externally; create a folder, for example, /opt/tomcat/conf/govapps/ and copy the RSDGLASSGovernanceApps.war/WEB-INF/conf/govapps.properties configuration file to this folder.
>
> Add the property `-Dgovapps.config.folder=/opt/tomcat/conf/govapps` to JAVA_OPTS in your server start script (in Tomcat the setenv.sh).

2.3. Authentication

> **Note:**
>
> The authentication settings must adhere to the authentication settings of your RSD GLASS® instance.

In Governance Apps, you can set the following authentication modes:

BASIC

The user needs to provide a user name and password as defined in RSD GLASS® which is used to authenticate.

SAMLTOKEN

When the user authenticates to Governance Apps, the authentication uses a SAML authentication token. The relevant token data is then delegated to RSD GLASS®.

SAMLOAUTH

When the user authenticates to Governance Apps, the authentication uses a SAML authentication token. The SAML token is transformed into an OAuth token and used to authenticate to RSD GLASS®.

2.3.1. Configuring BASIC Authentication

To configure Governance Apps to use BASIC authentication, set the `security.securityMode` property to `BASIC` in the govapps.properties or in the ZooKeeper.

2.3.2. Configuring SAMLTOKEN Authentication

To configure SAMLTOKEN authentication, define the following in the `govapp.properties` file:

1. Set the security.securityMode property or the respective ZooKeeper node to `SAMLTOKEN`.
2. Set the SAML related properties:

   security.saml.spEnityId
   URL of the Service Provider metadata

   security.saml.signMetadata
   flag indicating whether this service signs generated metadata

   security.saml.spBaseURL
   base URL to construct SAML endpoints

   The URL must be provided with the correct protocol, server, port and context path.

   security.saml.idpMetadataXmlFile
   path to the XML file with the IdP configuration metadata

   The file can be acquired from the IdP metadata endpoint.

   security.saml.keystoreFile
   Path to the keystore .jks file with security certificates

   security.saml.keystoreKey and security.saml.keystorePassword
   Keystore key and password for the samlKeystoreFile certificate

   security.saml.spBindingSSO
   Set the binding SSO binding order in a comma-separated list. The default value is `post,artifact,paos`.

3. Set the delegationProperty value (refer to *Password Delegation* on page 9).

2.3.2.1. Password Delegation

Alternatively, the user ID and a delegated password can be used to authenticate to the RSD GLASS® WS API instead of a SAML delegation.

Before you can use password delegation, you need to copy the delegated generic password into the `${govapps.config.folder}/govapps.properties` file.

To authenticate to the RSD GLASS® WS API, the following properties must be set in the `govapps.properties` file:

```
delegationProperty=<ENCRYPTED_PASSWORD>
samlUserIdattributeName=<REQUEST_ATTRIBUTE>
```

REQUEST_ATTRIBUTE depends on the identity provider setting and can be, for example, an email, user name, token, UID, NameID.

2.3.2.1.1. Encrypting Password

To generate an encrypted password and prevent using plain text password in the properties values, do the following:

1. On your command prompt, change to the `$CATALINA_HOME/webapps/RSDGLASSGovernanceApps/WEB-INF/` directory.
2. Copy the delegationProperty.jar library from the Governance App ZIP archive to the location.
3. Generate the encrypted version of your password:
   - On Windows run the following command:

     ```
     java -cp "delegationProperty.jar;classes;lib/*"
      com.rsd.glass.connectors.integration.DelegationPropertyHandler "<PASSWORD>"
     ```

   - On Linux and Mac OSX run the following command:

     ```
     java -cp "delegationProperty.jar:classes:lib/*" \
     com.rsd.glass.connectors.integration.DelegationPropertyHandler '<PASSWORD>'
     ```

   Figure 4: Example command run

   ```
   /opt/tomcat/webapps/RSDGLASSGovernanceApps/WEB-INF$ java -cp \
   "delegationProperty.jar:classes:lib/*" \
   com.rsd.glass.connectors.integration.DelegationPropertyHandler 'SECRET'
   Result: WLQBJwiHbvxJWahZCGi1Mw==
   ```

4. Enter the result into `delegationProperty` in the `govapp.properties` file.

2.3.3. Configuring SAMLOAUTH Authentication

To configure SAMLOAUTH authentication for Governance Apps, do the following:

1. Set the security.securityMode property or the respective ZooKeeper node to `SAMLOAUTH`.
2. Configure SAML properties:
   a) Set the SAML related properties:

      security.saml.spEnityId
      URL of the Service Provider metadata

      security.saml.signMetadata
      flag indicating whether this service signs generated metadata

security.saml.spBaseURL

base URL to construct SAML endpoints

The URL must be provided with the correct protocol, server, port and context path.

security.saml.idpMetadataXmlFile

path to the XML file with the IdP configuration metadata

The file can be acquired from the IdP metadata endpoint.

security.saml.keystoreFile

Path to the keystore .jks file with security certificates

security.saml.keystoreKey and security.saml.keystorePassword

Keystore key and password for the samlKeystoreFile certificate

security.saml.spBindingSSO

Set the binding SSO binding order in a comma-separated list. The default value is `post,artifact,paos`.

3. Configure identity provider (refer to *Configuring OpenAM Identity Provider* on page 12).
4. Configure OAuth properties:

security.oauth2.userInfoUrl

userInfoUrl (for example, for OpenAM: http://<IP>:<PORT>/openam/oauth2/tokeninfo)

security.oauth2.accessTokenUri

access-token-uri (for example, for the OpenAM: http://<IP>:<PORT>/openam/oauth2/access_token)

security.oauth2.userAuthorizationUri

user-authorization-uri (for example, for the OpenAM: http://<IP>:<PORT>/openam/oauth2/authorize)

security.oauth2.grantType

grant type for obtaining an access token for the resource

security.oauth2.authId

unique identifier for the Oauth authorization details

security.oauth2.clientId

client identifier for the Oauth authorization details

security.oauth2.secretId

client secret specified for the Oauth authorization details

security.oauth2.userIdResponseName

client identification in the token info response

security.oauth2.tenantIdResponseName

tenant identification in the token info response

Figure 5: Summary of properties with value formats

```
# oauth properties
security.oauth2.grantType=authorization_code
security.oauth2.authId=api
security.oauth2.clientId=ClientID
security.oauth2.secretId=SecretID
security.oauth2.userInfoUrl=http://<ip>:<port>/OpenAM/oauth2/tokeninfo
security.oauth2.accessTokenUri=http://<ip>:<port>/OpenAM/oauth2/access_token
security.oauth2.userAuthorizationUri=http://<ip>:<port>/OpenAM/oauth2/authorize
security.oauth2.userIdResponseName=uid
security.oauth2.tenantIdResponseName=description
```

2.3.3.1. Configuring Oauth2

To configure integration with Oauth2, define the following properties in the `govapps.properties` file:

security.oauth2.userInfoUrl

userInfoUrl (for example, for OpenAM: http://<IP>:<PORT>/openam/oauth2/tokeninfo)

security.oauth2.accessTokenUri

access-token-uri (for example, for the OpenAM: http://<IP>:<PORT>/openam/oauth2/access_token)

security.oauth2.userAuthorizationUri

user-authorization-uri (for example, for the OpenAM: http://<IP>:<PORT>/openam/oauth2/authorize)

security.oauth2.grantType

grant type for obtaining an access token for the resource

security.oauth2.authId

unique identifier for the Oauth authorization details

security.oauth2.clientId

client identifier for the Oauth authorization details

security.oauth2.secretId

client secret specified for the Oauth authorization details

security.oauth2.userIdResponseName

client identification in the token info response

security.oauth2.tenantIdResponseName

tenant identification in the token info response

Figure 6: Summary of properties with value formats

```
# oauth properties
security.oauth2.grantType=authorization_code
security.oauth2.authId=api
security.oauth2.clientId=ClientID
security.oauth2.secretId=SecretID
security.oauth2.userInfoUrl=http://<ip>:<port>/OpenAM/oauth2/tokeninfo
security.oauth2.accessTokenUri=http://<ip>:<port>/OpenAM/oauth2/access_token
security.oauth2.userAuthorizationUri=http://<ip>:<port>/OpenAM/oauth2/authorize
security.oauth2.userIdResponseName=uid
security.oauth2.tenantIdResponseName=description
```

2.3.3.2. Configuring OpenAM Identity Provider

To configure SSO authentication with the OpenAM server as Identity Provider, do the following:

1. Download the SP metadata from the URL `http://<GOVAPPS_SERVER>:<PORT>/RSDGLASSGovernanceApps/saml/metadata/alias/defaultAlias`
2. Log on to OpenAM as Administrator.
3. On the **OpenAM Common Tasks** page, click **Register Remote Service Provider**.
4. Upload the SP metadata file.
5. Verify that the SP has been registered on the Federation tab.
6. Navigate to **Agents** > **OAuth 2.0**.
7. Add the OAuth2 redirection URI `http://<GOVAPPS_SERVER>:<PORT>/RSDGLASSGovernanceApps/oauth/login` to the list.
8. Store the idp metadata from `http://<OPENAM_ADDRESS>:<OPENAM_PORT>/OpenAM/saml2/jsp/exportmetadata.jsp` to the `idp.xml` in the configuration directory.

3. White-Label Support

White label support for the Wizard includes support for custom images and custom CSS settings using a SCSS file:

Images

You can substitute the default images with your own images directly on the server in the *$CATALINA_HOME*/webapps/RSDGLASSGovernanceApps/VAADIN/themes/govappsTheme/img/ directory. The new images are applied instantly: refresh your browser to see the changes. You might need to clear your browser cache as well.

> **Important:**
>
> Ensure that the new images use the same resolution as the default images.

You can change the following images:

Main logo (resolution: 245 x 50 pixels)

logo.png

Notification icons (resolution: 33 x 33 pixels)

- error.png
- ok.png
- warning.png

Treeview icons (resolution: 24 x 24 pixels)

- fileplan.png
- folder.png
- record.png
- recordClass.png

Button-left background (resolution: 6 x 36 pixels)

- left.png
- right.png

Button-right background (resolution: 2000 x 36 pixels)

- left-pressed.png
- right-pressed.png

CSS

You can modify the following theme parameters of the visual theme in the *$CATALINA_HOME*/webapps/RSDGLASSGovernanceApps/VAADIN/themes/govappsTheme/govappsTheme.scss file:

$font-stack
: global font stack used on the page

$page-background
: main background color header background excluded

$selection-color
: color of a highlighted item in the Treeview

$header-text-color

color of the text in the header on the right side

$header-background

background color for the header

$content-border-color

global border color

$button-text-color

color of the button label

$button-text-color-hover

color of a button label when holding mouse pointer over it

$button-height

button height

When changing button height, consider changing the background images to this height to achieve a consistent look.

$button-font-size

size of the button label font

After you have modified the theme variables in the SCSS file, you need to compile the CSS theme.

To compile the CSS theme, run the following command:

- On Windows:

```
$CATALINA_HOME/webapps/RSDGLASSGovernanceApps/VAADIN/themes/govappsTheme/
compileTheme.bat
```

- On Linux or Mac OSX:

```
$ java -cp "../../../WEB-INF/lib/*" com.vaadin.sass.SassCompiler
 styles.scss styles.css
```

White-label support for Widgets includes support for custom CSS files:

External CSS

The URL to external CCS resources is defined by the `presentation.externalResources.css` property in the `govapps.properties` file.

In the `presentation.externalResources.css` location, create your CSS files for individual widgets. Make sure the files are named `<WIDGET_NAME>.custom.css`, for example, `publicView.custom.css`, `dispositionApproval.custom.css`.

> **Note:**
>
> Once you define the `presentation.externalResources.css` property, it is only the referenced CSS resources that are used: the default CSS files in the WAR are ignored.

4. Internalization and Localization

> **Important:** Localization properties have been refactored so as to use parameters. Make sure to adapt any localization property files created for the previous version to use the new localization properties.

Localization strings are defined in the `messages_<LANGUAGE_CODE>.properties` files. Every locale has it own properties file.

The system attempts to identify the user's localization based on the browser or repository locale and then searches for the respective localization file. The localization file for a locale is identified by the language code in the file name. If no file for the locale is found the default localization file `messages.properties` is used.

To use custom localization files, you need to create a location with your messages.properties files and enter the path to the`presentation.externalResources.translations` property in the `govapps.properties` file. Note that if the messages.properties in the `presentation.externalResources.translations` location is not found, the messages.properties files in the RSDGLASSGovernanceApps.war file are used.

4.1. Creating Custom Localization File

To create a custom localization file, do the following:

1. Copy the `RSDGLASSGovernanceApps.war/WEB-INF/classes/messages.properties` file to a location accessible to your server.
2. Rename the copied `message.properties` file to `messages_<LANGUAGE_CODE>.properties` file and localize the property values in the file.
3. Set the location path to your message.properties file in the `presentation.externalResources.translations` property in the `govapps.properties` file.
4. Rename the `message.properties` file to `messages_<LANGUAGE_CODE>.properties` file.
5. Optionally, set the `presentation.useRepositoryLocale` property in the `govapps.properties` file to `true` so the system uses the repository's locale.

   If set to false, the system uses the language settings of the user's web browser.
6. Restart your Tomcat server to load the new messages.properties file.

5. Additional Properties

5.1. Cataloging and Governing Multiple Documents

The following properties in the govapps.properties file apply to bulk operations:

`functionality.bulk.skipCataloged`

boolean

if true, cataloged content is skipped

`functionality.bulk.executor.corePoolSize`

integer

minimum count of threads for parallel processing

`functionality.bulk.executor.maximumPoolSize`

integer

maximum count of threads for parallel processing

`functionality.bulk.executor.keepAliveTimeInSeconds`

integer

maximum idle time for a thread before termination in seconds

`functionality.bulk.executor.queueCapacity`

integer

when the number of items in the queue is higher than the defined integer, a new thread is created

5.2. Using Repository Locale

To display all fields, button names and messages in the language of the content repository or in the language specified in your browser settings, set `presentation.useRepositoryLocale` to `true`.

5.3. Configuring Reclassification

Reclassification can be applied only on Records that are assigned a dynamic Metadata Group Type: if the user wants to reclassify, a metadata group needs to be set as dynamic in RSD GLASS® Policy Manager. You can do so as follows:

1. Navigate to RSD GLASS Policy Manager and set up a new Metadata Group Type with desired name (for example, Contract_MGT).
2. To define the metadata group as dynamic, add the 'Unique code' suffix `-Dynamic` (for example, `Contract_MGT-Dynamic`).

To such Records, the following reclassification properties apply:

silentReclassify

boolean

If true, when cataloging or governing an already cataloged Component, the Component and its Record are reclassified automatically if necessary without any notification.

Note that the original Component and Record location is neither visible in the File Plan tree or its detail.

hideFqcList

comma-separated list of nodes to be ignored (child nodes are ignored as well).

For example, with the following value node /0000000001/0007/0000000001 and /0000000003/0002/0000000011 and their children are ignored:

```
functionality.reclassify.hideFqcList=
/0000000001/0007/0000000001,/0000000003/0002/0000000011
```

5.4. Configuring Correlation

Correlation allows the system to connect a repository document with an existing Record based on a correlation metadata pair, the repository correlation ID and correlation GLASS Query.

The following properties apply to the correlation feature:

`functionality.correlation.addComponentsToExistingRecord`

boolean

When set to `true`, RSD GLASS® attempts to correlate a correlation ID sent from Governance Apps with an already existing Record.

`functionality.correlation.addComponentsToExistingRecordOnly`

boolean

When set to `true`, only documents with an existing Record can be governed: if Records with a correlation ID do not exist, no new Record is created, and the system returns a warning.

If set to `false` and no correlated Record exists for the requested Record ID, a new Record is created. However, the Record is not declared in RSD GLASS® and therefore it is not governed.

`functionality.correlation.declareAddedRecords`

boolean

If the `functionality.correlation.addComponentsToExistingRecordOnly` property is set to `false` and this property is set to `true` and no correlated Record exists for the requested Record ID, a new Record is created and declared in RSD GLASS®.

`functionality.correlation.declareAddedComponents`

boolean

If set to `true`, the correlated Components are declared.

`functionality.correlation.glassQuery=nodeType:record AND record.uid`

string

A glassQuery which defines which Record is searched (in this example uid in record node type)

`functionality.correlation.repositoryCorrelationId`

string

Metadata element in the role of correlation ID (ID is taken from the content repository)

5.5. Configuring Download Link Behavior

The Download Link-related properties, configure the download links of the components displayed in the widgets, such as the component download link in the **Search and Legal Hold** widget.

The following properties apply to the download link:

`functionality.widgets.directDownloadLink`

- When set to `true`, the download link is a direct link to the repository file.
- When set to `false`, the download link calls a RSD GLASS® web service to acquire the component.

6. Widgets

Governance Apps provides widgets, which are HTML5 components you can incorporate into your web or browserapplication.

For more information on available widgets, refer to *Governance Apps Guide*.

6.1. Integrating Widgets

To integrate Widgets with your application, embed the Widgets directly or in an iframe:

Embedding Widgets

- To embed a Widget as an iframe, add the following code with the respective Widget name to your web application:

```
<iframe src="http://[GLASS_URL]/[WIDGET_NAME].html"></iframe>
```

- To embed a Widget directly, add the following code with the respective Widget name to your web application:

```
<div class="rsd-w-container rsd-w-container-[WIDGET_NAME]" data-
widget="widgetZero"></div><script src="http://[GLASS_URL]/rsd.min.js"></
script>
```

Figure 7: Example page with Widgets embedded

```
<!DOCTYPE html>
<html>
<head>
  <title>My Page</title>
  <!--CSS FILES USED ON THE EMBEDDED WIDGETS-->
  <link href="https://stgw.rsd-cloud.com/RSDGLASSGovernanceApps/VAADIN/widgets/dist/
css/bootstrap.min.css" rel="stylesheet">
  <link href="https://stgw.rsd-cloud.com/RSDGLASSGovernanceApps/VAADIN/widgets/dist/
css/bootstrap-theme.min.css" rel="stylesheet">
  <link href="https://stgw.rsd-cloud.com/RSDGLASSGovernanceApps/VAADIN/widgets/dist/
css/theme.css" rel="stylesheet">
</head>
<body>
  <!-- JavaScript that loads the widgets-->
  <script src="https://stgw.rsd-cloud.com/RSDGLASSGovernanceApps/VAADIN/widgets/dist/
rsd.min.js"></script>
  <p>Embedded widgets:</p>
  <div class="rsd-w-container rsd-w-container-publicView" data-widget="publicView"></
div>
  <div class="rsd-w-container rsd-w-container-dispositionApproval" data-
widget="dispositionApproval"></div>
  <div class="rsd-w-container rsd-w-container-activityStream" data-
widget="activityStream"></div>
  </body>
</html>
```

Figure 8: Example page with Widgets and Wizard in iframes

```
<!DOCTYPE html>
<html>
<head>
  <title>My Page</title>
</head>
<body>
    <iframe width="1800" height="900" src="https://stgw.rsd-cloud.com/
RSDGLASSGovernanceApps/VAADIN/widgets/widget1.html" border="0" frameborder="0"
 id="viewerFrame" allowfullscreen="true" webkitallowfullscreen="true"
 mozallowfullscreen="true"></iframe>
  <iframe width="1800" height="900" src="https://stgw.rsd-cloud.com/
RSDGLASSGovernanceApps/VAADIN/widgets/widget2.html" border="0" frameborder="0"
 id="viewerFrame" allowfullscreen="true" webkitallowfullscreen="true"
 mozallowfullscreen="true"></iframe>
  <iframe width="1800" height="900" src="https://stgw.rsd-cloud.com/
RSDGLASSGovernanceApps/VAADIN/widgets/widget4.html" border="0" frameborder="0"
 id="viewerFrame" allowfullscreen="true" webkitallowfullscreen="true"
 mozallowfullscreen="true"></iframe>
  </body>
</html>
```